EGOTECHWORLD • SINHALA LEARNING SERIES • 2026

Java Banking App Development

ජාවා භාවිතයෙන් සරල බැංකු යෙදුමක් හදමු

Java ඇයි? • Java basics • Windows Desktop • Deposit • Withdraw • Transfer • Testing • QA

LITTLE STAR BANK

Rs **10,000** → Rs **9,850**

JAVA 25 LTS | SWING UI | BIGDECIMAL

අවුරුදු 10+ නවකයන් සඳහා පියවරෙන් පියවර සිංහල වැඩපොත • Demo money only

White pages · VS Code-style code · Desktop previews · Fully tested example project

START HERE • 2026

පොත භාවිතා කරන්නේ කොහොමද?

මෙය theory පමණක් නොවේ. එක් එක් lesson එක කියවා, අදාල code එක run කර, mini challenge එක විසඳා ඔබේ සටහන ලියන්න.

01	Java කියන්නේ මොකක්ද සහ බැංකු සඳහා එය වැදගත් වන්නේ ඇයි?
02	JDK + VS Code + Windows CMD setup
03	Java classes, methods, conditions, BigDecimal
04	Little Star Bank desktop app එක build කර run කිරීම
05	SDLC, test cases, automated QA සහ documentation

අතිශය වැදගත් ආරක්ෂක සටහන

මෙය ළමුන්ට Java coding ඉගැන්වීම සඳහා පමණක් නිර්මාණය කළ banking simulator එකකි. සැබෑ මුදල් ගනුදෙනු, බැංකු ගිණුම්, passwords හෝ customer data මෙහි භාවිතා නොකරන්න.

ඔබට අවශ්‍ය දේ:

Windows PC • JDK 21+ (JDK 25 LTS නිර්දේශිතයි) • VS Code (විකල්ප) • පහත ZIP example project එක • ප්‍රශ්න අසන හොඳ සිතක්!

READING MAP

අධ්‍යයන මාර්ග සිතියම



01	Java භාවිතා කරන්නේ ඇයි?	10 පාඩම්
02	Java ඉගෙනීමේ පළමු පියවර	18 පාඩම්
03	ව්‍යාපෘතිය සැලසුම් කිරීම	10 පාඩම්
04	ගණනය හා Business Logic	10 පාඩම්
05	Data Save කිරීම සහ History	8 පාඩම්
06	Windows Desktop UI නිර්මාණය	10 පාඩම්
07	App එක Run කරමු	9 පාඩම්
08	Testing, QA සහ Documentation	10 පාඩම්
09	Banking Security and Ethics	10 පාඩම්
10	Publish නොව Portfolio Demo	10 පාඩම්

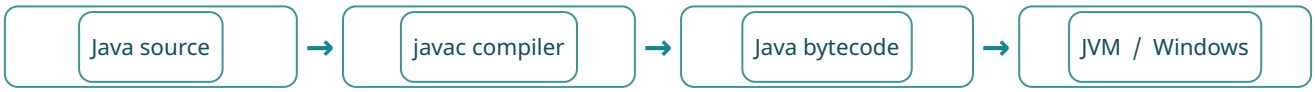
මීළඟ කොටසේ source code appendix, test cases, quick reference, learning checklist සහ official links ඇත.

CHAPTER 01 / WHY JAVA IN BANKING

Java භාවිතා කරන්නේ ඇයි?

01

මෙම කොටසේ පාඩම් 10ක් ඉගෙනගෙන අවසානයේ කුඩා වැඩක් කරමු.



මෙම කොටසේ ඉගෙනගන්නා මාතෘකා

Java මොකක්ද? • බැංකු සඳහා Java ජනප්‍රිය වන්නේ ඇයි? • JVM එකේ මැජික් එක • Java සහ Python වෙනස • Java සහ JavaScript එකමද? • Java 2026 අනුව version තේරීම

කේතය කටපාඩම් කරන්න එපා. එක් එක් පියවර මොකක් සඳහාද කියලා සොයා බලන්න.

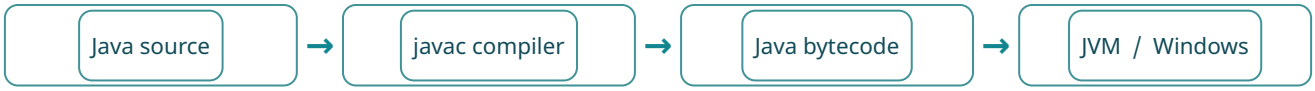
LESSON 001 • CHAPTER 01

Java මොකක්ද?

Java කියන්නේ programming language එකක්. අපි ලියන code එක JDK මඟින් class files බවට compile කර JVM එක තුළ ක්‍රියාත්මක කරනවා.

අපි උදාහරණයකින් බලමු

අද අපි හදන්නේ Windows හි ධාවනය වන කල්පිත බැංකු වැඩසටහනක්.



ඔබේ වැඩේ / TRY IT

ඔබ දන්නා software තුනක් ලියන්න.

_____ _____

මතක තබාගන්න: Java code එක browser එකට පමණක් සීමා නැත.

සැලකිල්ල: Java යනු නමකින් පමණක් ආරක්ෂිත banking system එකක් නොවේ.

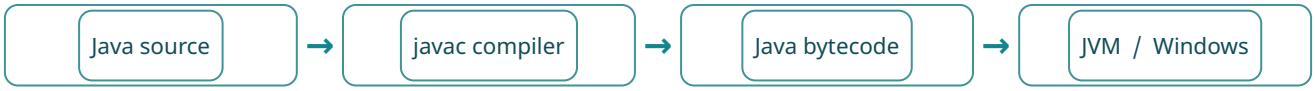
LESSON 002 • CHAPTER 01

බැංකු සඳහා Java ජනප්‍රිය වන්නේ ඇයි?

ස්ථාපිත tools, JVM පරිසරය, තද type checking සහ විශාල enterprise ecosystem එක නිසා Java server-side පද්ධතිවල බහුලයි.

අපි උදාහරණයකින් බලමු

ගනුදෙනු ගණනක් එකවර ලැබෙන app එකක code නඩත්තු කිරීම වැදගත්.



ඔබේ වැඩේ / TRY IT

එකම app එක සෑම දිනකම වැඩ කිරීම වැදගත් වන්නේ ඇයි?

_____ _____

මතක තබාගන්න: නඩත්තු කළ හැකි, test කළ හැකි software වටිනාකමක්.

සැලකිල්ල: Java භාවිතා කළ පමණින් app එකට නීතිමය අනුකූලතාව ලැබෙන්නේ නැහැ.

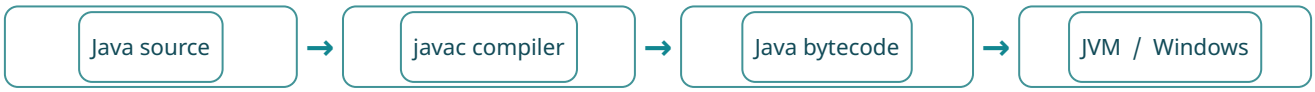
LESSON 003 • CHAPTER 01

JVM එකේ මැජික් එක

JVM කියන්නේ compiled Java bytecode ධාවනය කරන පරිසරය. එකම source code එකට ගැළපෙන platform එකක JDK/JRE අවශ්‍යයි.

අපි උදාහරණයකින් බලමු

Windows පරිගණකයකත් Linux server එකකත් JVM තිබුණොත් බොහෝ Java code ධාවනය කළ හැක.



ඔබේ වැඩේ / TRY IT

Source -> javac -> .class -> JVM චිත්‍රය අඳින්න.

_____ _____

මතක තබාගන්න: Write once, run on compatible JVMs.

සැලකිල්ල: Desktop UI appearance එක OS අනුව වෙනස් විය හැක.

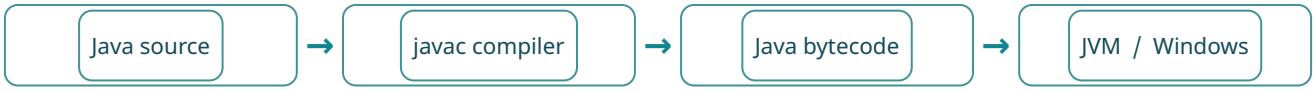
LESSON 004 • CHAPTER 01

Java සහ Python වෙනස

Python ඉක්මනින් code අත්හදා බැලීමට පහසුයි. Java වල types සහ class structure මුලදී වැඩි නමුත් විශාල projects නඩත්තු කිරීමට උදව් වෙනවා.

අපි උදාහරණයකින් බලමු

BankAccount class එකක balance එක BigDecimal ලෙස type කරන්න පුළුවන්.



ඔබේ වැඩේ / TRY IT

Python/Java අතර ඔබ කැමති දෙකේ ලක්ෂණ දෙකක් ලියන්න.

_____ _____

මතක තබාගන්න: හොඳ භාෂාව තෝරන්නේ ව්‍යාපෘතිය අනුවයි.

සැලකිල්ල: කිසිම language එකක් සියලු වැඩ සඳහා හොඳම එක නොවේ.

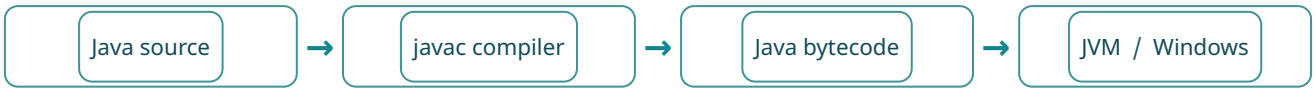
LESSON 005 • CHAPTER 01

Java සහ JavaScript එකමද?

නැහැ. නම් සමාන වුණත් Java, JavaScript වෙන වෙනම languages. Java Swing desktop UI එකක් නිර්මාණය කළ හැකියි.

අපි උදාහරණයකින් බලමු

JavaScript බොහෝවිට web browser logic සඳහා; Java මෙහි Windows app එක සඳහා.



ඔබේ වැඩේ / TRY IT

Java සහ JavaScript වෙනස් වාක්‍ය දෙකකින් පැහැදිලි කරන්න.

_____ _____

මතක තබාගන්න: නම සමාන වීමෙන් technology එකම නොවේ.

සැලකිල්ල: Java web app එකක් server එකේද ධාවනය විය හැක.

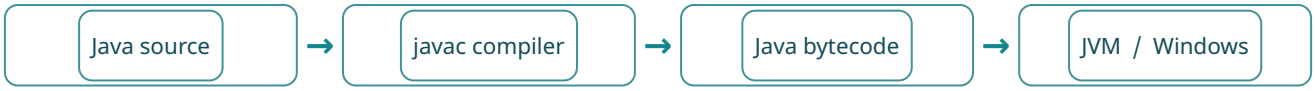
LESSON 006 • CHAPTER 01

Java 2026 අනුව version තේරීම

2026 ඔක්තෝබර් වන විට Java 27 නවතම feature release; Java 25 latest LTS. Java 21 ද LTS සහ beginners සඳහා භාවිත කළ හැක.

අපි උදාහරණයකින් බලමු

මෙම project එක JDK 21+ සමඟ compile වනවා; JDK 25 LTS හොඳ install තේරීමක්.



ඔබේ වැඩේ / TRY IT

`java -version` කියවා ඔබේ version එක සටහන් කරන්න.

_____ _____

මතක තබාගන්න: LTS කියන්නේ දීර්ඝ සහාය නියමිත release එකක්.

සැලකිල්ල: Download කිරීමේදී official JDK distributor එකක් භාවිතා කරන්න.

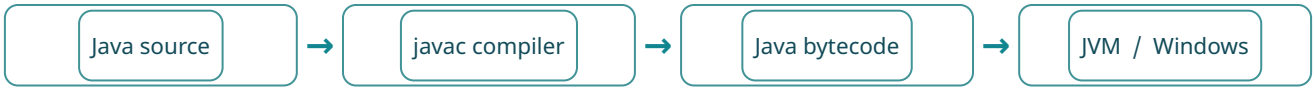
LESSON 007 • CHAPTER 01

JDK, JRE, JVM කවුද?

JDK තුළ compiler සහ development tools ඇත. JVM code ධාවනය කරයි. JRE යන සංකල්පය runtime අවශ්‍යතා සඳහා භාවිතා වේ.

අපි උදාහරණයකින් බලමු

අපට `javac` සහ `java` commands දෙකම අවශ්‍යයි.



ඔබේ වැඩේ / TRY IT

Compiler සහ runner වෙන වෙනම තෝරන්න.

_____ _____

මතක තබාගන්න: JDK = code ලියලා run කිරීමේ මෙවලම්.

සැලකිල්ල: Modern JDK packages සාමාන්‍යයෙන් runtime එකත් ඇතුළත් කරයි.

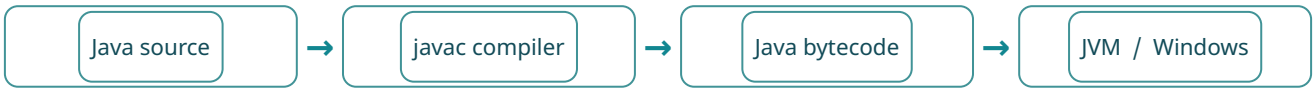
LESSON 008 • CHAPTER 01

Java වල BigDecimal

මුදල් ගණිතයේදී double floating point rounding errors ඇති කළ හැක. BigDecimal decimal values නිවැරදිව පාලනයට උපකාරී.

අපි උදාහරණයකින් බලමු

`new BigDecimal("0.10")` සහ `new BigDecimal("0.20")` එකතු කරමු.



ඔබේ වැඩේ / TRY IT

රුපියල් 125.50 + 74.25 ගණනය කරන්න.

_____ _____

මතක තබාගන්න: Money = BigDecimal, precision rules සහ validation.

සැලකිල්ල: BigDecimal නිවැරදි වුණත් business rules නිවැරදි විය යුතුයි.

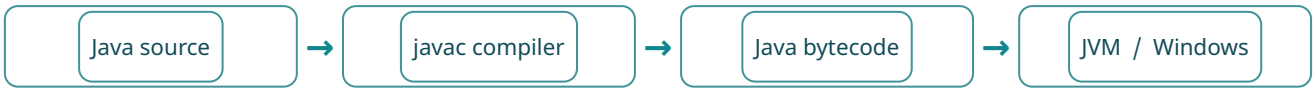
LESSON 009 • CHAPTER 01

Bank එකක Java කරන වැඩ

Backend services, APIs, transaction workflows, reporting සහ integration වැනි කොටස් Java භාවිතයෙන් තැනිය හැක.

අපි උදාහරණයකින් බලමු

Mobile banking app එකෙන් එන request එක Java server එකක process විය හැක.



ඔබේ වැඩේ / TRY IT

Mobile -> API -> Service -> Database diagram එක අඳින්න.

_____ _____

මතක තබාගන්න: දර්ශනය වන්නේ app එකේ UI කොටස පමණක්.

සැලකිල්ල: අපේ simulator එකට public API එකක් හෝ real bank connection එකක් නැහැ.

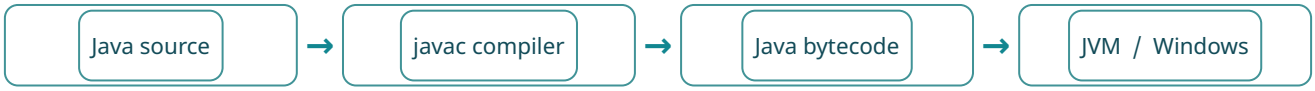
LESSON 010 • CHAPTER 01

මෙය simulator එකක් පමණයි

අපේ Little Star Bank app එකේ Amaya සහ Nimal කියන කල්පිත ගිණුම් දෙක තියෙනවා. සැබෑ customer data භාවිතා නොකරන්න.

අපි උදාහරණයකින් බලමු

A100 = Rs 10,000.00; A200 = Rs 5,000.00.



ඔබේ වැඩේ / TRY IT

මේවා සැබෑ මුදල් නොවන හේතුව ලියන්න.

_____ _____

මතක තබාගන්න: Safe test data සමඟ coding ඉගෙනගැනීම.

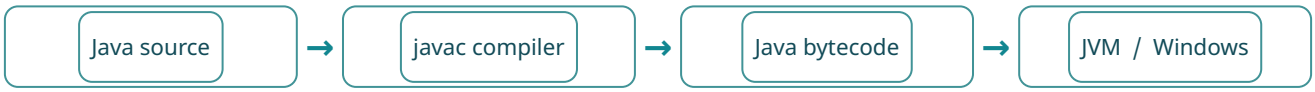
සැලකිල්ල: Authentication නැති app එකක් real bank එකක් ලෙස deploy නොකරන්න.

CHAPTER 02 / JAVA FOUNDATIONS

Java ඉගෙනීමේ පළමු පියවර

02

මෙම කොටසේ පාඩම් 18ක් ඉගෙනගෙන අවසානයේ කුඩා වැඩක් කරමු.



මෙම කොටසේ ඉගෙනගන්නා මාතෘකා

Java ස්ථාපනය • පළමු Hello Java program එක • Java file structure • Variables කියන්නේ පෙට්ටි • String සමඟ වැඩ කිරීම • Numbers: int සහ double

කේතය කටපාඩම් කරන්න එපා. එක් එක් පියවර මොකක් සඳහාද කියලා සොයා බලන්න.

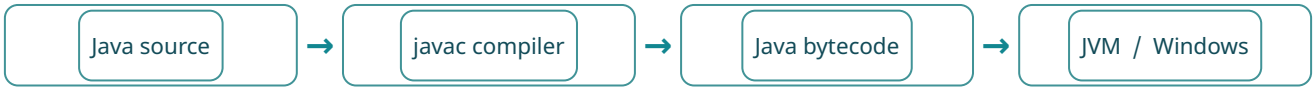
LESSON 011 • CHAPTER 02

Java ස්ථාපනය

JDK 21 හෝ JDK 25 LTS install කර PATH නිවැරදිද බලන්න. VS Code එකට Extension Pack for Java භාවිතා කළ හැක.

අපි උදාහරණයකින් බලමු

CMD: `java -version` සහ `javac -version`.



ඔබේ වැඩේ / TRY IT

Commands දෙකම run කර output ලියන්න.

_____ _____

මතක තබාගන්න: දෙකම හඳුනාගන්නා විට setup සූදානම්.

සැලකිල්ල: අනවශ්‍ය administrator privileges භාවිතා නොකරන්න.

LESSON 012 • CHAPTER 02

පළමු Hello Java program එක

Java program එක `class` එකක `main` method එකෙන් ආරම්භ කළ හැක. `System.out.println` කියන්නේ CMD එකට message පෙන්වීම.

අපි උදාහරණයකින් බලමු

HelloBank.java තුළ `Hello Little Star Bank!` print කරන්න.

○○○ VS CODE / TERMINAL — Java / Windows CMD

```
public class HelloBank {
    public static void main(String[] args) {
        // My very first Java message.
        System.out.println("Hello Little Star Bank!");
    }
}
```

ඔබේ වැඩේ / TRY IT

Message එක ඔබ කැමති නමකට වෙනස් කරන්න.

_____ _____

මතක තබාගන්න: Main method = program entry point.

සැලකිල්ල: File name සහ public class name එක සමානව තබන්න.

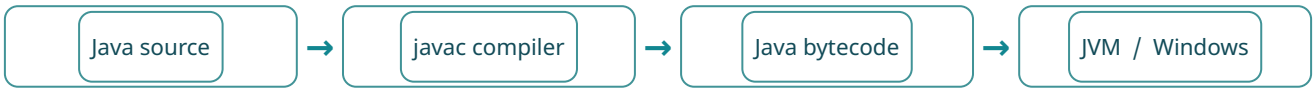
LESSON 013 • CHAPTER 02

Java file structure

`.java` file වල අපේ code; compiler නිර්මාණය කරන්නේ `.class` bytecode. `out/` folder එක compiled files තබන්න.

අපි උදාහරණයකින් බලමු

`src/egotech/bank/Main.java` -> `out/egotech/bank/Main.class`.



ඔබේ වැඩේ / TRY IT

Source file සහ compiled file අඳුනාගන්න.

_____ _____

මතක තබාගන්න: Source සහ build output වෙනස් දේවල්.

සැලකිල්ල: Compiled binary file එක අතින් edit නොකරන්න.

LESSON 014 • CHAPTER 02

Variables කියන්නේ පෙට්ටි

Variable එකක් value එකක් අල්ලා තබන නමක්. Java type එක මඟින් ඒ value එකේ ආකාරය කියනවා.

අපි උදාහරණයකින් බලමු

`String name = "Amaya";` සහ `int age = 10;`.

○○○ VS CODE / TERMINAL — Java / Windows CMD

```
String name = "Amaya";
int items = 5;
boolean active = true;
System.out.println(name + " has " + items + " items.");
```

ඔබේ වැඩේ / TRY IT

String සහ int වෙනස උදාහරණයකින් ලියන්න.

_____ _____

මතක තබාගන්න: විවිධ data සඳහා නිවැරදි types භාවිත කරන්න.

සැලකිල්ල: වැරදි type එකක් compiler හඳුනාගන්න පුළුවන්.

LESSON 015 • CHAPTER 02

String සමඟ වැඩ කිරීම

String එකක වචන සහ පෙළ තබයි. User display names වැනි දේවල් Strings. Secret passwords strings ලෙස plain text තබාගැනීම මෙම පාඩමේ නොකරමු.

අපි උදාහරණයකින් බලමු

`String greeting = "Hello " + name;.`

○○○ VS CODE / TERMINAL — Java / Windows CMD

```
String name = "Nimal";
String greeting = "Hello " + name;
System.out.println(greeting);
```

ඔබේ වැඩේ / TRY IT

නම වෙනස් කර දෙවන greeting එක හදන්න.

_____ _____

මතක තබාගන්න: String concat මඟින් වචන එකතු කළ හැක.

සැලකිල්ල: සැබෑ customer information examples වල භාවිතා කරන්න එපා.

LESSON 016 • CHAPTER 02

Numbers: int සහ double

`int` පූර්ණ සංඛ්‍යා සඳහා; `double` approximate decimal ගණනය සඳහා. Demo money සඳහා BigDecimal වඩා ගැළපේ.

අපි උදාහරණයකින් බලමු

5 items ගණන `int` එකක, banking amount `BigDecimal` එකක.

○○○ VS CODE / TERMINAL — Java / Windows CMD

```
int itemCount = 3;
double scienceExample = 0.1 + 0.2;
BigDecimal money = new BigDecimal("0.30");
```

ඔබේ වැඩේ / TRY IT

5 items සහ Rs 5.25 අතර type තෝරන්න.

_____ _____

මතක තබාගන්න: Money සඳහා floating-point avoid කරන්න.

සැලකිල්ල: `double` සමඟ සමානකම් direct compare කිරීමේදී සීමා ඇත.

LESSON 017 • CHAPTER 02

Boolean - ඇත්තද බොරුද?

Boolean value එක `true` හෝ `false`. Conditions සඳහා මෙය උපකාරී.

අපි උදාහරණයකින් බලමු

`boolean enough = balance.compareTo(amount) >= 0;.`

VS CODE / TERMINAL — Java / Windows CMD

```
BigDecimal balance = new BigDecimal("100.00");
BigDecimal amount = new BigDecimal("120.00");
boolean enough = balance.compareTo(amount) >= 0;
System.out.println(enough); // false
```

ඔබේ වැඩේ / TRY IT

Rs 100 balance, Rs 120 withdraw: enough කොපමණද?

_____ _____

> **මතක තබාගන්න:** Withdraw කිරීමට මුදල් ප්‍රමාණවත්ද පළමුව බලන්න.

සැලකිල්ල: Condition එක UI එකේ පමණක් නොව service එකේද තිබිය යුතුයි.

LESSON 018 • CHAPTER 02

if / else තීරණ

`if` condition එක true නම් code එකක්; නැත්නම් `else` code එක ක්‍රියාත්මක වෙයි.

අපි උදාහරණයකින් බලමු

Amount එක zero නම් error දක්වන්න.

○○○ VS CODE / TERMINAL — Java / Windows CMD

```
if (amount.signum() <= 0) {
    System.out.println("Invalid amount");
} else {
    System.out.println("Continue");
}
```

ඔබේ වැඩේ / TRY IT

Negative amount එකකට condition එකක් ලියන්න.

_____ _____

මතක තබාගන්න: Validation code නිවැරදි මාර්ග තෝරයි.

සැලකිල්ල: Input එක පිළිගන්නා කලින් නැවත පරීක්ෂා කරන්න.

LESSON 019 • CHAPTER 02

Comparisons

`> < ==` මඟින් numbers සසඳන්න. BigDecimal සමඟ `compareTo` භාවිතා කරන්න.

අපි උදාහරණයකින් බලමු

`balance.compareTo(amount) < 0` නම් balance ප්‍රමාණවත් නැහැ.

○○○ VS CODE / TERMINAL — Java / Windows CMD

```
int a = 10;
System.out.println(a > 5); // true
System.out.println(a == 10); // true
// BigDecimal objects use compareTo.
System.out.println(balance.compareTo(amount));
```

ඔබේ වැඩේ / TRY IT

Rs 250 ඉතිරි, Rs 200 withdraw — result?

_____ _____

මතක තබාගන්න: Numbers සහ objects compare කරන ක්‍රම වෙනස්.

සැලකිල්ල: BigDecimal objects `==` මඟින් compare නොකරන්න.

LESSON 020 • CHAPTER 02

Loops for

එකම පියවර කිහිපවරක් කිරීමට `for` loop. History entries පෙන්වීමට loop යොදාගන්න පුළුවන්.

අපි උදාහරණයකින් බලමු

Transaction entries 3ක් iterate කර print කරන්න.

○○○ VS CODE / TERMINAL — Java / Windows CMD

```
for (int i = 1; i <= 5; i++) {
    System.out.println("Step " + i);
}
```

ඔබේ වැඩේ / TRY IT

1 සිට 5 දක්වා අංක print කරන්න.

_____ _____

මතක තබාගන්න: Loop එක repeat work අඩු කරයි.

සැලකිල්ල: Loop අවසන් වන condition එක වැදගත්.

LESSON 021 • CHAPTER 02

Loops while

Condition එක true පවතින තාක් `while` loop ක්‍රියාත්මක වේ. CMD menus සඳහා භාවිතා කළ හැක.

අපි උදාහරණයකින් බලමු

`while (running)` menu එක නැවත පෙන්වයි.

○○○ VS CODE / TERMINAL — Java / Windows CMD

```
boolean running = true;
while (running) {
    System.out.println("Menu is active");
    running = false; // Stop the example loop.
}
```

ඔබේ වැඩේ / TRY IT

Exit choice එකක් පැමිණි විට loop නවත්වන්න.

_____ _____

> **මතක තබාගන්න:** Infinite loops වළක්වන්න.

සැලකිල්ල: Swing app එකේ main UI thread block කරන loop එකක් දාන්න එපා.

LESSON 022 • CHAPTER 02

Methods - කුඩා උපදෙස්

Method එකක් එක් වගකීමක් කරයි. `deposit()` කියන්නේ මුදල් එකතු කරන business operation එක.

අපි උදාහරණයකින් බලමු

`public void deposit(...)` method signature එක බලන්න.

○○○ VS CODE / TERMINAL — Java / Windows CMD

```
static int add(int a, int b) {
    return a + b;
}
System.out.println(add(2, 3)); // 5
```

ඔබේ වැඩේ / TRY IT

ගණන 2ක් එකතු කරන method එකක් සැලසුම් කරන්න.

_____ _____

මතක තබාගන්න: කුඩා methods test කිරීම පහසුයි.

සැලකිල්ල: UI button එක තුළම සියලු logic ලියන්න එපා.

LESSON 023 • CHAPTER 02

Parameters

Method එකට දෙන inputs parameters. `withdraw(id, amount)` තුළ id සහ amount දෙකක් ඇත.

අපි උදාහරණයකින් බලමු

`deposit("A100", "250.00").`

○○○ VS CODE / TERMINAL — Java / Windows CMD

```
// id and amount are parameters.
service.deposit("A100", "250.00");
```

ඔබේ වැඩේ / TRY IT

id A200 කර call එකක් ලියන්න.

_____ _____

මතක තබාගන්න: Names සහ types නිවැරදි order එකේ දිය යුතුයි.

සැලකිල්ල: User text validate කර පසුව operation කරන්න.

LESSON 024 • CHAPTER 02

Return values

Method එකක් ප්‍රතිඵලයක් ආපසු ලබාදිය හැක. `balance(id)` මුදල් ප්‍රමාණය return කරයි.

අපි උදාහරණයකින් බලමු

`BigDecimal current = bank.balance("A100");.`

VS CODE / TERMINAL — Java / Windows CMD

```
BigDecimal current = bank.balance("A100");
System.out.println(Money.format(current));
```

ඔබේ වැඩේ / TRY IT

Return කළ amount text ලෙස පෙන්වන්න.

_____ _____

මතක තබාගන්න: return සහ print එකම දෙයක් නොවේ.

සැලකිල්ල: Business service එක UI widgets භාවිත නොකර return data ලබාදීම හොඳයි.

LESSON 025 • CHAPTER 02

Classes සහ Objects

Class කියන්නේ සැලැස්මක්; object කියන්නේ එයින් සෑදුණු instance එකක්. BankService එක balances සහ transaction rules අල්ලාගනී.

අපි උදාහරණයකින් බලමු

`BankService service = new BankService(file);.`

VS CODE / TERMINAL — Java / Windows CMD

```
BankService service = new BankService(
    Path.of("data", "ledger.tsv")
);
```

ඔබේ වැඩේ / TRY IT

Class, object දෙකට උදාහරණ දෙන්න.

_____ _____

මතක තබාගන්න: OOP මඟින් වෙන වෙනම වගකීම් වෙන් කරයි.

සැලකිල්ල: Class එක විශාල වූ විට responsibilities බෙදන්න.

LESSON 026 • CHAPTER 02

Constructor

`new BankService(file)` කරන විට constructor එක මුල් values සකස් කරයි; saved data load කරයි.

අපි උදාහරණයකින් බලමු

පළමු run එකේ fictional opening balances නිර්මාණය වේ.

○○○ VS CODE / TERMINAL — Java / Windows CMD

```
public BankService(Path path) throws IOException {
    store = new LedgerStore(path);
    // Load the fictional account history.
}
```

ඔබේ වැඩේ / TRY IT

පළමු run එක සහ පසු run එක වෙනස් වන්නේ ඇයි?

_____ _____

> **මතක තබාගන්න:** Constructor initialized state එකක් තැබිය යුතුයි.

සැලකිල්ල: Constructor exceptions handle කළ යුතුයි.

LESSON 027 • CHAPTER 02

Arrays, List සහ Map

`List` වල transaction history අනුපිළිවෙළින්; `Map` වල account ID එකට balance සම්බන්ධ කර තබන්න පුළුවන්.

අපි උදාහරණයකින් බලමු

`Map<String, BigDecimal> balances`.

○○○ VS CODE / TERMINAL — Java / Windows CMD

```
List<LedgerEntry> items = service.history();
Map<String, String> labels = service.accounts();
System.out.println(labels.get("A100"));
```

ඔබේ වැඩේ / TRY IT

A100 -> 10000 සහ A200 -> 5000 map එක අඳින්න.

_____ _____

> **මතක තබාගන්න:** දත්ත තැබීමේ structure තෝරාගැනීම වැදගත්.

සැලකිල්ල: Collection තුළ සැබෑ customer secrets තබන්න එපා.

LESSON 028 • CHAPTER 02

Exceptions

වැරදි input එකක් ඉදිරියට නොයවලා `IllegalArgumentException` එවන්න. UI එකේදී දරුවාට තේරෙන message පෙන්වන්න.

අපි උදාහරණයකින් බලමු

`throw new IllegalArgumentException("Invalid amount");.`

○○○ VS CODE / TERMINAL — Java / Windows CMD

```
if (input.isBlank()) {
    throw new IllegalArgumentException(
        "Please enter an amount"
    );
}
```

ඔබේ වැඩේ / TRY IT

අකුරු ඇතුළත් කළොත් පෙන්විය යුතු message එක ලියන්න.

_____ _____

මතක තබාගන්න: Errors නොසඟවා හඳුනාගන්න.

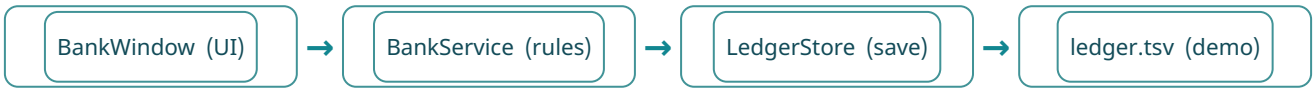
සැලකිල්ල: Exception එකක් අල්ලා නිශ්ශබ්දව අතහරින්න එපා.

CHAPTER 03 / PLANNING AND SDLC

ව්‍යාපෘතිය සැලසුම් කිරීම

03

මෙම කොටසේ පාඩම් 10ක් ඉගෙනගෙන අවසානයේ කුඩා වැඩක් කරමු.



මෙම කොටසේ ඉගෙනගන්නා මාතෘකා

Project goal එක ලියමු • SDLC - අදහසින් app එකට • User story • Functional requirements • Non-functional requirements • Out of scope

කේතය කටපාඩම් කරන්න එපා. එක් එක් පියවර මොකක් සඳහාද කියලා සොයා බලන්න.

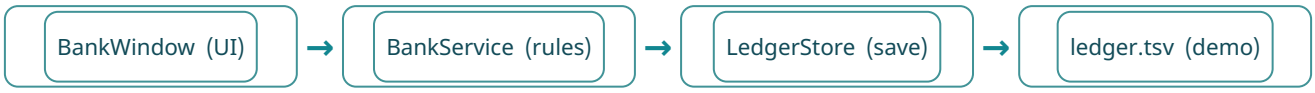
LESSON 029 • CHAPTER 03

Project goal එක ලියමු

Project එක පටන්ගන්න කලින් කාටද, මොකටද, කරන්නේ මොනවාද කියලා සටහන් කරගන්න.

අපි උදාහරණයකින් බලමු

Goal: demo money deposit, withdraw, transfer, transaction history.



ඔබේ වැඩේ / TRY IT

පාරිභෝගිකයා වෙනුවට මෙහි learners කවුද?

_____ _____

මතක තබාගන්න: අරමුණ සරලව ලියාගත් විට scope එක පැහැදිලියි.

සැලකිල්ල: සැබෑ ගෙවීම් සිදු නොවන බව goal එකේම කියන්න.

LESSON 030 • CHAPTER 03

SDLC - අදහසින් app එකට

SDLC: Requirements, Design, Development, Testing, Release, Maintenance. එක් එක් අදියරේ output එකක් තිබේ.

අපි උදාහරණයකින් බලමු

Requirement -> Sketch -> Java classes -> QA -> ZIP.



ඔබේ වැඩේ / TRY IT

අදියර හයක් අනුපිළිවෙළට ලියන්න.

_____ _____

> **මතක තබාගන්න:** වැඩ පටන් ගැනීමට පෙර සැලැස්මක් උපකාරී.

සැලකිල්ල: Testing එක අවසානයේ විතරක් නොව මුළු ගමනේම කරන්න.

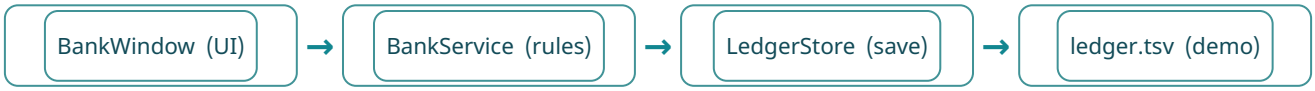
LESSON 031 • CHAPTER 03

User story

User story කියන්නේ කෙනෙක්ට කළ යුතු වැඩ පැහැදිලි කරන වාක්‍යයක්.

අපි උදාහරණයකින් බලමු

“As a learner, I want to deposit demo money so I can see a new balance.”



ඔබේ වැඩේ / TRY IT

Withdraw සඳහා user story එකක් ලියන්න.

_____ _____

මතක තබාගන්න: User action සහ එයින් ලැබෙන value දෙකම දක්වන්න.

සැලකිල්ල: Story එකකට acceptance criteria ද තබන්න.

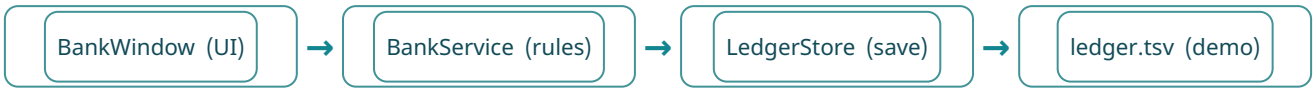
LESSON 032 • CHAPTER 03

Functional requirements

App එක කරන කාර්යයන් FR ලෙස ලියන්න. Deposit, withdrawal, transfer, report history ඒ අතර වේ.

අපි උදාහරණයකින් බලමු

FR03: Withdraw cannot go below zero.



ඔබේ වැඩේ / TRY IT

FR03 සඳහා acceptance criteria 2ක් ලියන්න.

_____ _____

මතක තබාගන්න: Requirement එකක් test එකකට සම්බන්ධ කළ හැක.

සැලකිල්ල: කල්පිත ගිණුම් දෙකෙන් ඔබ්බට accounts සෑදීම scope එකෙන් ඉවත් කළා.

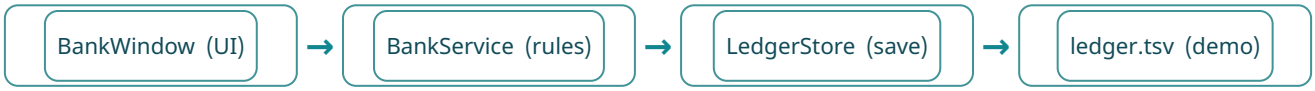
LESSON 033 • CHAPTER 03

Non-functional requirements

App වේගවත්ද, භාවිතය පහසුද, machine එකේ run වෙනවාද වැනි ගුණාංග NFR.

අපි උදාහරණයකින් බලමු

NFR: JDK 21+ සහ standard Java libraries පමණක් භාවිතා කරයි.



ඔබේ වැඩේ / TRY IT

Performance හෝ usability requirement එකක් ලියන්න.

_____ _____

මතක තබාගන්න: Quality එක කුඩා apps වලටත් වැදගත්.

සැලකිල්ල: Banking-grade security මෙම example scope එකට ඇතුළත් නැහැ.

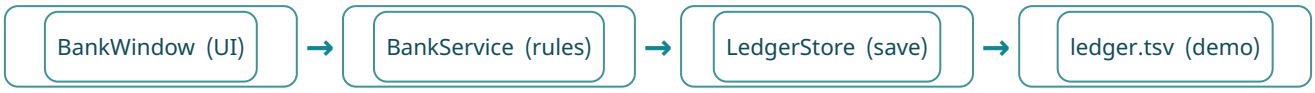
LESSON 034 • CHAPTER 03

Out of scope

App එකේ නොමැති features ලිවීමත් වැදගත්. එවිට simulator එක real bank එකක් කියා වැරදියට නොසිතෙයි.

අපි උදාහරණයකින් බලමු

Login, KYC, debit card, payments, interest calculation නැහැ.



ඔබේ වැඩේ / TRY IT

Out of scope features තුනක් හඳුනාගන්න.

_____ _____

මතක තබාගන්න: යමක් නොකිරීමටත් හේතුවක් තිබිය යුතුයි.

සැලකිල්ල: Real customer data කිසිවිටක භාවිතා නොකරන්න.

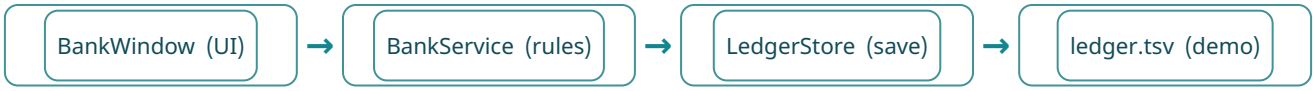
LESSON 035 • CHAPTER 03

UI wireframe

Code ලියන්න කලින් තිරයේ කොටස් ඇඳ බලමු: title, balance cards, amount input, buttons, history.

අපි උදාහරණයකින් බලමු

Paper එකක A100/A200 balances, Deposit/Withdraw/Transfer buttons සලකුණු කරන්න.



ඔබේ වැඩේ / TRY IT

පැන්සලකින් screen එක ඇඳන්න.

_____ _____

මතක තබාගන්න: Design first, code second.

සැලකිල්ල: User error message පෙන්වන්න තැනක්ද අවශ්‍යයි.

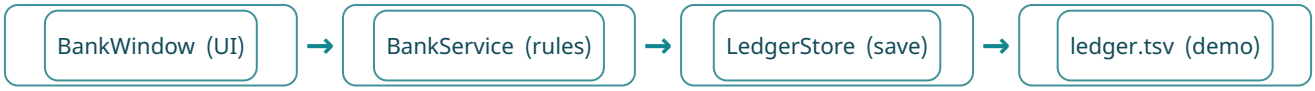
LESSON 036 • CHAPTER 03

System architecture

UI -> Service -> Store -> local file ලෙස layers වෙන් කරමු. එවිට business rules GUI එකෙන් වෙනම test කළ හැක.

අපි උදාහරණයකින් බලමු

BankWindow -> BankService -> LedgerStore -> ledger.tsv.



ඔබේ වැඩේ / TRY IT

Arrow diagram එක අඳින්න.

_____ _____

මතක තබාගන්න: Separation of concerns = වගකීම් වෙන් කිරීම.

සැලකිල්ල: Real bank architecture මෙයට වඩා සංකීර්ණයි.

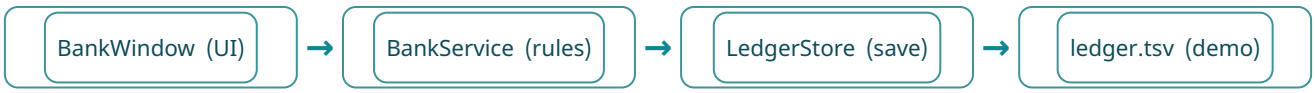
LESSON 037 • CHAPTER 03

File plan

Project එකේ source files සෑම එකකටම වගකීමක්: Money, BankService, BankWindow, LedgerStore, Tests.

අපි උදාහරණයකින් බලමු

`src/egotech/bank/` යටතේ Java files පිහිටයි.



ඔබේ වැඩේ / TRY IT

Source folder tree එක සටහන් කරන්න.

_____ _____

මතක තබාගන්න: Naming සහ folders හොඳින් තබාගන්න.

සැලකිල්ල: `.class` files source folder එකට දාන්න එපා.

LESSON 038 • CHAPTER 03

Data flow

Deposit click -> amount validate -> journal write -> balance update -> UI refresh. පියවර මාරුවී නොයෑම වැදගත්.

අපි උදාහරණයකින් බලමු

Valid amount Rs 250 එක A100 ට එකතු කරලා refresh වෙන ආකාරය.



ඔබේ වැඩේ / TRY IT

පියවර පහක් අංක කර ලියන්න.

_____ _____

මතක තබාගන්න: UI එකේ value වෙනස් වීමට business event එකක් අවශ්‍යයි.

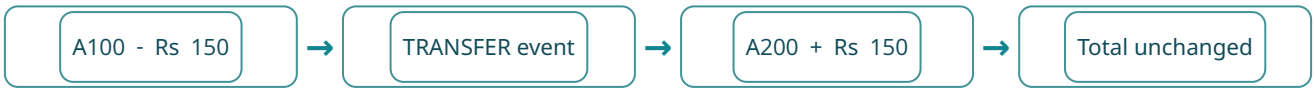
සැලකිල්ල: Error පැමිණි විට මුදල් නොවෙනස් විය යුතුයි.

CHAPTER 04 / MONEY RULES

ගණනය හා Business Logic

04

මෙම කොටසේ පාඩම් 10ක් ඉගෙනගෙන අවසානයේ කුඩා වැඩක් කරමු.



> **මෙම කොටසේ ඉගෙනගන්නා මාතෘකා**
>
> Money.parse හඳුනාගනිමු • Amount scale එක • Limits සහ ranges • Opening balances • Deposit algorithm • Withdraw algorithm

කේතය කටපාඩම් කරන්න එපා. එක් එක් පියවර මොකක් සඳහාද කියලා සොයා බලන්න.

LESSON 039 • CHAPTER 04

Money.parse හඳුනාගනිමු

Input field එකෙන් ලැබෙන්නේ String එකක්. එය පිළිගත හැකි money format එකකට පරිවර්තනය කරන්න.

අපි උදාහරණයකින් බලමු

`Money.parse("250.00")` -> BigDecimal(250.00).

○○○ VS CODE / TERMINAL — Java / Windows CMD

```
BigDecimal amount = Money.parse("250.00");
System.out.println(Money.format(amount));
// Result: Rs 250.00
```

ඔබේ වැඩේ / TRY IT

"2.345" වැරදි ඇයි කියන්න.

_____ _____

මතක තබාගන්න: Input parsing යනු validation පළමු පියවරයි.

සැලකිල්ල: Space සහ විශේෂ අක්ෂර අවසර ඇතිදැයි rules තීරණය කරයි.

LESSON 040 • CHAPTER 04

Amount scale එක

250, 250.5, 250.50 වැනි values 2-decimal money ලෙස තබන්න. තෙවන decimal digit එක ප්‍රතික්ෂේප කරමු.

අපි උදාහරණයකින් බලමු

`setScale(2, RoundingMode.UNNECESSARY).`

VS CODE / TERMINAL — Java / Windows CMD

```
BigDecimal m = new BigDecimal("250.5");
BigDecimal fixed = m.setScale(2);
System.out.println(fixed); // 250.50
```

ඔබේ වැඩේ / TRY IT

100.5 ගබඩා වන්නේ කෙසේද?

_____ _____

මතක තබාගන්න: Currency unit සඳහා decimal places නියමිතව තබාගන්න.

සැලකිල්ල: සියලු currencies වල 2 decimals නැහැ; මෙය LKR demo එකකි.

LESSON 041 • CHAPTER 04

Limits සහ ranges

Zero, negatives සහ Rs 1,000,000 ට වැඩි amount demo එකට නොපිළිගන්න.

අපි උදාහරණයකින් බලමු

`amount.signum() <= 0` නම් error.

○○○ VS CODE / TERMINAL — Java / Windows CMD

```
if (amount.signum() <= 0) {
    throw new IllegalArgumentException(
        "Amount must be positive"
   );
}
```

ඔබේ වැඩේ / TRY IT

-10, 0, 100, 1000001 ලෙස test cases හදන්න.

_____ _____

> **මතක තබාගන්න:** Validation නැතිව balance update කරන්න එපා.

සැලකිල්ල: Maximum limit එක product rule එකක්; සැබෑ බැංකු නීතියක් නොවේ.

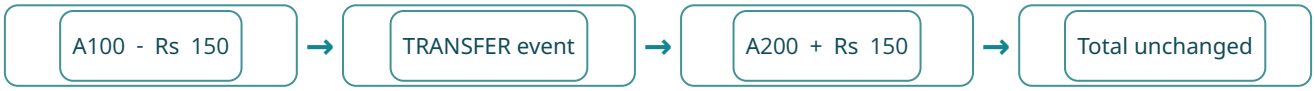
LESSON 042 • CHAPTER 04

Opening balances

App එක පළමු වර ධාවනය වන විට A100 සහ A200 සඳහා fictional opening entries ලියනවා.

අපි උදාහරණයකින් බලමු

A100 10000; A200 5000.



ඔබේ වැඩේ / TRY IT

මුල් මුදල එකතුව කොපමණද?

_____ _____

මතක තබාගන්න: Opening event එක history හි පෙනේ.

සැලකිල්ල: Real bank එකක opening balance සාක්ෂි සහ නියමිත process වලින් පාලනය වෙනවා.

LESSON 043 • CHAPTER 04

Deposit algorithm

Deposit සඳහා account හඳුනාගන්න, amount validate කරන්න, event save කරන්න, balance වැඩි කරන්න.

අපි උදාහරණයකින් බලමු

10000 + 250 = 10250.

○○○ VS CODE / TERMINAL — Java / Windows CMD

```
public synchronized void deposit(String id, String amount)
        throws IOException {
    requireAccount(id);
    post(new LedgerEntry(Instant.now(), "DEPOSIT",
         "-", id, Money.parse(amount)));
}
```

ඔබේ වැඩේ / TRY IT

A200 ට 40 deposit කලොත්?

_____ _____

මතක තබාගන්න: Valid deposit එකක් balance වැඩි කරයි.

සැලකිල්ල: Save failure එකක් ආවොත් screen එක වෙනස් නොවිය යුතුයි.

LESSON 044 • CHAPTER 04

Withdraw algorithm

Withdraw සඳහා sufficient funds check කරන්න. Balance negative නොවීමට rule එක service layer එකේ තියෙන්න ඕනේ.

අපි උදාහරණයකින් බලමු

A100 10000 - 150 = 9850.

VS CODE / TERMINAL — Java / Windows CMD

```
BigDecimal value = Money.parse(amount);
if (balance(id).compareTo(value) < 0) {
    throw new IllegalArgumentException(
        "Insufficient demo balance"
   );
}
```

ඔබේ වැඩේ / TRY IT

Rs 11000 ඉවත් කරන්න යන විට expected message?

_____ _____

මතක තබාගන්න: මුරපද/අවසර මෙහි නැත: සැබෑ system නොවේ.

සැලකිල්ල: Overdraft products වෙනම rules සහ contracts අවශ්‍ය විය හැක.

LESSON 045 • CHAPTER 04

Transfer algorithm

Transfer එකක ගිණුම් දෙකක balances එකවර වෙනස් වෙනවා: එකෙන් අඩුයි, අනෙක වැඩියි.

අපි උදාහරණයකින් බලමු

A100: 10000 - 500 = 9500; A200: 5000 + 500 = 5500.

○○○ VS CODE / TERMINAL — Java / Windows CMD

```
// One journal event contains both demo accounts.
post(new LedgerEntry(Instant.now(), "TRANSFER",
     from, to, value));
```

ඔබේ වැඩේ / TRY IT

Transfer පසු මුළු එකතුව බලන්න.

_____ _____

මතක තබාගන්න: Transfer මුදල නැතිවීම හෝ දෙවරක් යාම නොවිය යුතුයි.

සැලකිල්ල: Production transfer සඳහා ACID DB transactions අවශ්‍යයි.

LESSON 046 • CHAPTER 04

Same-account rule

එකම account එකෙන් එම account එකට transfer කිරීම demo එකේ reject කරනවා.

අපි උදාහරණයකින් බලමු

A100 -> A100 Rs 100 invalid.

○○○ VS CODE / TERMINAL — Java / Windows CMD

```
if (from.equals(to)) {
    throw new IllegalArgumentException(
        "Choose a different recipient"
    );
}
```

ඔබේ වැඩේ / TRY IT

එවැනි transfer එකක් UI එකෙන් try කරන්න.

_____ _____

මතක තබාගන්න: Business rules simple වුණත් පිළිවෙළට test කළ යුතුයි.

සැලකිල්ල: User හට හේතුව කියන message එකක් පෙන්වන්න.

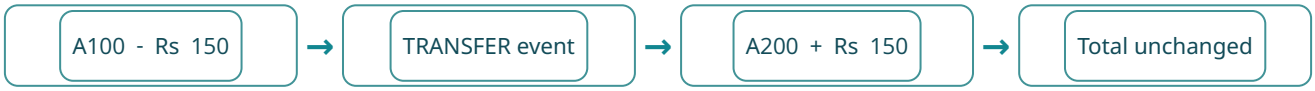
LESSON 047 • CHAPTER 04

Invariant - මුදල් එකතුව

Transfer කරන විට දෙගිණුම් එකතුව නොවෙනස් විය යුතුයි. මෙය invariant එකක්.

අපි උදාහරණයකින් බලමු

Before 15000; transfer Rs 800; after 15000.



ඔබේ වැඩේ / TRY IT

Transfer තුනකින් පසු මුළු demo මුදල පරීක්ෂා කරන්න.

_____ _____

මතක තබාගන්න: Invariant check මඟින් ගැටලු ඉක්මනින් සොයාගන්න.

සැලකිල්ල: Deposit/withdraw නම් මුළු එකතුව වෙනස් වෙනවා.

LESSON 048 • CHAPTER 04

Thread safety - synchronized

අපේ `synchronized` methods එකම Java process තුළ calls අනුපිළිවෙළට කරයි.

අපි උදාහරණයකින් බලමු

`public synchronized void transfer(...).`

○○○ VS CODE / TERMINAL — Java / Windows CMD

```
public synchronized void transfer(
    String from, String to, String amount
) throws IOException {
    // Validate, save, then update demo balances.
}
```

ඔබේ වැඩේ / TRY IT

Multiple windows එකවර භාවිතා කළොත් ගැටලු මොනවාද?

_____ _____

මතක තබාගන්න: Concurrency ප්‍රශ්න මුදල් යෙදුම්වල වැදගත්.

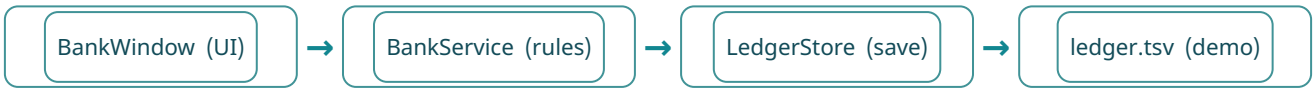
සැලකිල්ල: මේක multi-process හෝ network security විසඳුමක් නොවේ.

CHAPTER 05 / PERSISTENCE

Data Save කිරීම සහ History

05

මෙම කොටසේ පාඩම් 8ක් ඉගෙනගෙන අවසානයේ කුඩා වැඩක් කරමු.



මෙම කොටසේ ඉගෙනගන්නා මාතෘකා

Journal කියන්නේ මොකක්ද? • ledger.tsv ගොනුව • Instant සහ time • Reading saved records • Save-then-apply • Real database අවශ්‍ය ඇයි?

කේතය කටපාඩම් කරන්න එපා. එක් එක් පියවර මොකක් සඳහාද කියලා සොයා බලන්න.

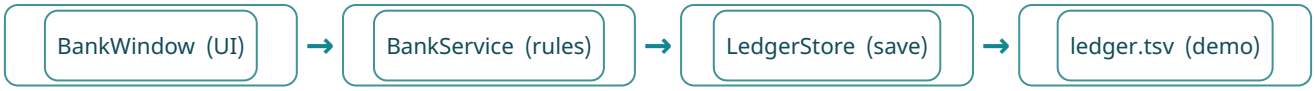
LESSON 049 • CHAPTER 05

Journal කියන්නේ මොකක්ද?

ඇති වූ වෙනස්කම් event ලෙස ලියා තබන log එකක්. Reload වන විට එම events කියවා balances නැවත ගණනය කරමු.

අපි උදාහරණයකින් බලමු

OPEN, DEPOSIT, WITHDRAW, TRANSFER events.



ඔබේ වැඩේ / TRY IT

Balance එක save කරනවාට වඩා events save කිරීමේ වාසිය ලියන්න.

_____ _____

මතක තබාගන්න: History මඟින් ක්‍රියාවලිය අනුගමනය කළ හැක.

සැලකිල්ල: Teaching journal එක tamper-proof නොවේ.

LESSON 050 • CHAPTER 05

ledger.tsv ගොනුව

Tab-separated fields: timestamp, type, from, to, amount. මෙය database එකක් නොව plain text file එකක්.

අපි උදාහරණයකින් බලමු

`2026-...\tTRANSFER\tA100\tA200\t150.00.`

○○○ VS CODE / TERMINAL — Java / Windows CMD

```
// Demo journal (tab-separated):
// time   type   from   to   amount
// 2026-10-31T...Z  TRANSFER  A100  A200  150.00
```

ඔබේ වැඩේ / TRY IT

Transaction එකක fields පහක් නම් කරන්න.

_____ _____

මතක තබාගන්න: Structured rows කියවීම පහසු කරයි.

සැලකිල්ල: File එක external edit කළොත් integrity අවදානම ඇත.

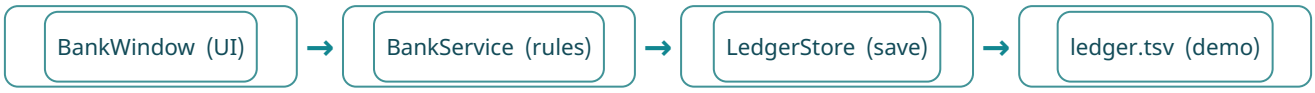
LESSON 051 • CHAPTER 05

Instant සහ time

Transaction එක සිදු වූ අවස්ථාව `Instant.now()` ලෙස save කරනවා. UI එකේ timestamp සරල ආකාරයෙන් පෙන්වයි.

අපි උදාහරණයකින් බලමු

`2026-10-31T...Z` UTC time format.



ඔබේ වැඩේ / TRY IT

UTC කියන්නේ මොකක්ද සොයන්න.

_____ _____

මතක තබාගන්න: Event chronology දත්තවල වැදගත්.

සැලකිල්ල: Real banks වල time zones, trace IDs සහ clock sync වැදගත්.

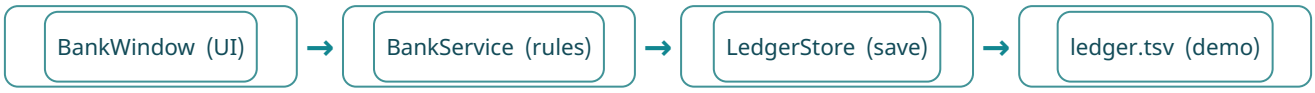
LESSON 052 • CHAPTER 05

Reading saved records

App නැවත අරින විට `LedgerStore.load()` file එකේ lines කියවා entries බවට පරිවර්තනය කරනවා.

අපි උදාහරණයකින් බලමු

5 events තිබුණොත් 5 entries restore වේ.



ඔබේ වැඩේ / TRY IT

Restart පසු balance තවම නිවැරදිද බලන්න.

_____ _____

මතක තබාගන්න: Persistence නැතිව restart එකේ data නැතිවෙයි.

සැලකිල්ල: Bad/corrupt file line තිබුණොත් demo fail-fast කරයි.

LESSON 053 • CHAPTER 05

Save-then-apply

අපේ demo එක record file එකට append කිරීමෙන් පසුව memory balance update කරනවා. Write error ආවොත් screen value නොවෙනස්.

අපි උදාහරණයකින් බලමු

`store.append(entry); apply(entry);.`

○○○ VS CODE / TERMINAL — Java / Windows CMD

```
private void post(LedgerEntry entry) throws IOException {
    store.append(entry); // Save the demo event.
    apply(entry);       // Update the in-memory balance.
    events.add(entry);
}
```

ඔබේ වැඩේ / TRY IT

Order එක පෙරළුවොත් අවදානම කුමක්ද?

_____ _____

> **මතක තබාගන්න:** Operation order consistency සඳහා වැදගත්.

සැලකිල්ල: Plain text storage crash-safe transactional solution එකක් නොවේ.

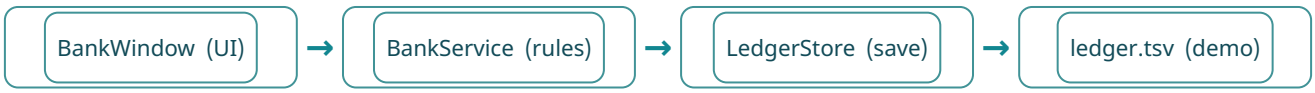
LESSON 054 • CHAPTER 05

Real database අවශ්‍ය ඇයි?

සැබෑ financial records සඳහා ACID transactions, concurrency controls, backup, audit, disaster recovery වැනි සැකසුම් අවශ්‍ය වේ.

අපි උදාහරණයකින් බලමු

PostgreSQL හෝ වෙනත් transaction DB එකක් production architecture එකක භාවිතා කළ හැක.



ඔබේ වැඩේ / TRY IT

Text file සහ DB අතර වෙනස්කම් 3ක් ලියන්න.

_____ _____

මතක තබාගන්න: Demo store සහ production DB එක සමාන නොවේ.

සැලකිල්ල: Database එකක් තිබුණත් authentication සහ authorization වෙනම අවශ්‍යයි.

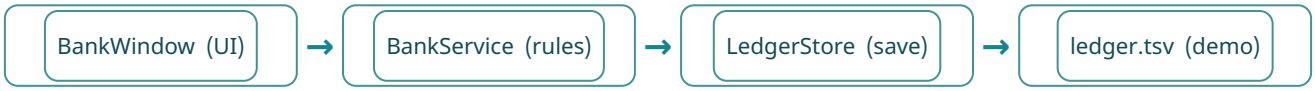
LESSON 055 • CHAPTER 05

History පෙන්වීම

History ලැයිස්තුවෙන් අවසන් event මුලින් පෙන්වන්න reverse order loop එකක් යොදාගත්තා.

අපි උදාහරණයකින් බලමු

UI හි TRANSFER / DEPOSIT rows බලන්න.



ඔබේ වැඩේ / TRY IT

ඔබ deposit එකක් කළ පසු top row එක කුමක්ද?

_____ _____

මතක තබාගන්න: Userට තමා කළ ක්‍රියාව confirm කිරීම වැදගත්.

සැලකිල්ල: Sensitive records ලොග් කිරීමේදී ප්‍රවේශය සීමා කරන්න.

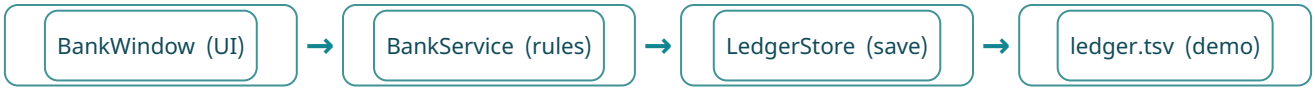
LESSON 056 • CHAPTER 05

Reset කිරීම සහ Backup

Demo app එක නවතා `data/ledger.tsv` ඉවත් කළොත් නව කල්පිත balances පටන්ගනී. Production වල මෙවැනි delete බරපතළයි.

අපි උදාහරණයකින් බලමු

Learning reset: close app -> backup file -> delete demo file.



ඔබේ වැඩේ / TRY IT

Reset කිරීමට පෙර copy එකක් තබා බලන්න.

_____ _____

මතක තබාගන්න: Backups නැතිව හදිසි හානි වලින් ආරක්ෂා විය නොහැක.

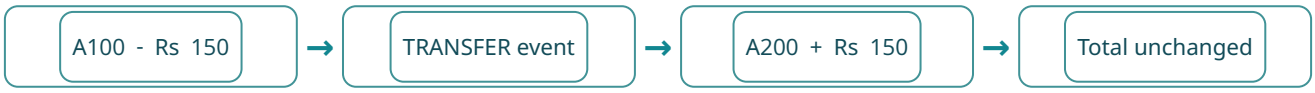
සැලකිල්ල: සැබෑ financial records අත්තනෝමතිකව මකන්න නොහැක.

CHAPTER 06 / JAVA SWING GUI

Windows Desktop UI නිර්මාණය

06

මෙම කොටසේ පාඩම් 10ක් ඉගෙනගෙන අවසානයේ කුඩා වැඩක් කරමු.



මෙම කොටසේ ඉගෙනගන්නා මාතෘකා

Swing කියන්නේ මොකක්ද? • JFrame - ප්‍රධාන කවුළුව • JLabel - කියවන්නට text • JTextField - input ගැනීම • JComboBox - ගිණුම් තෝරාගැනීම • JButton සහ click events

කේතය කටපාඩම් කරන්න එපා. එක් එක් පියවර මොකක් සඳහාද කියලා සොයා බලන්න.

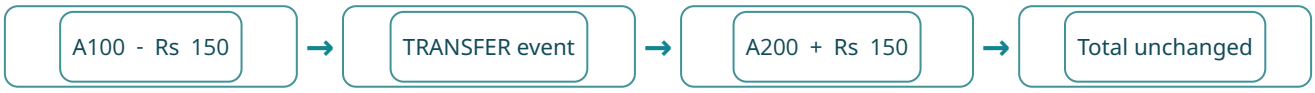
LESSON 057 • CHAPTER 06

Swing කියන්නේ මොකක්ද?

Swing යනු Java desktop interface library එකක්. JFrame, JLabel, JButton, JTextField වැනි widgets වලින් window හදමු.

අපි උදාහරණයකින් බලමු

```
new JFrame("Little Star Bank").
```



ඔබේ වැඩේ / TRY IT

Window එකේ controls පහක් නම් කරන්න.

_____ _____

මතක තබාගන්න: GUI යනු පරිශීලකයා දකින කොටස.

සැලකිල්ල: Swing පමණක් නොව වෙනත් UI frameworks ද ඇත.

LESSON 058 • CHAPTER 06

JFrame - ප්‍රධාන කවුළුව

JFrame මඟින් Windows desktop window එකක් පෙන්වයි. Title, size සහ close behaviour තබන්න.

අපි උදාහරණයකින් බලමු

```
setSize(800, 620);
```

○○○ VS CODE / TERMINAL — Java / Windows CMD

```
JFrame frame = new JFrame("Little Star Bank");
frame.setSize(800, 620);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setVisible(true);
```

ඔබේ වැඩේ / TRY IT

Window size එක 900x650 ලෙස වෙනස් කරන්න.

_____ _____

මතක තබාගන්න: Frame එක app එකේ outer shell එකයි.

සැලකිල්ල: කුඩා monitor වලට ගැළපෙන්න layout පරීක්ෂා කරන්න.

LESSON 059 • CHAPTER 06

JLabel - කියවන්නට text

Balance label දෙකෙන් A100, A200 හි fictional money පෙන්වයි.

අපි උදාහරණයකින් බලමු

`balanceA.setText("Amaya: " + Money.format(...)).`

VS CODE / TERMINAL — Java / Windows CMD

```
balanceA.setText(
    "Amaya: " + Money.format(service.balance("A100"))
);
```

ඔබේ වැඩේ / TRY IT

Display name එක වෙනස් කර screen එක බලන්න.

_____ _____

මතක තබාගන්න: Userට ප්‍රතිඵල පැහැදිලිව පෙනිය යුතුයි.

සැලකිල්ල: Text color එකම තොරතුරේ එකම සලකුණ නොවිය යුතුයි.

LESSON 060 • CHAPTER 06

JTextField - input ගැනීම

User enter කරන demo amount එක text field එකෙන් කියවයි; එය validation පසු BigDecimal බවට පත්වේ.

අපි උදාහරණයකින් බලමු

`amount.getText()` -> `Money.parse(...)`.

○○○ VS CODE / TERMINAL — Java / Windows CMD

```
JTextField amount = new JTextField("250.00");
String raw = amount.getText();
BigDecimal checked = Money.parse(raw);
```

ඔබේ වැඩේ / TRY IT

`abc` enter කර error message බලන්න.

_____ _____

> **මතක තබාගන්න:** UI input සැමවිටම අවිශ්වාස ලෙස සැලකිය යුතුයි.

සැලකිල්ල: Input field එකට live bank passwords දාන්න එපා.

LESSON 061 • CHAPTER 06

JComboBox - ගිණුම් තෝරාගැනීම

From හා To accounts තෝරාගන්න dropdown controls. Fixed fictional IDs පමණක් තිබෙනවා.

අපි උදාහරණයකින් බලමු

A100 -> A200 transfer.

○○○ VS CODE / TERMINAL — Java / Windows CMD

```
JComboBox<String> from = new JComboBox<>(
    new String[] {"A100", "A200"}
);
```

ඔබේ වැඩේ / TRY IT

From සහ To මාරු කර බලන්න.

_____ _____

මතක තබාගන්න: Dropdown එක typo වැළකීමට උපකාරී.

සැලකිල්ල: Server-side authorization substitute එකක් නොවේ.

LESSON 062 • CHAPTER 06

JButton සහ click events

Deposit button click කළ විට service.deposit method එක call වෙනවා; දත්ත පරීක්ෂා කර UI refresh වෙනවා.

අපි උදාහරණයකින් බලමු

`button.addActionListener(e -> execute("DEPOSIT"));.`

VS CODE / TERMINAL — Java / Windows CMD

```
JButton deposit = new JButton("Deposit");
deposit.addActionListener(event -> {
    // Call the service and refresh the screen.
});
```

ඔබේ වැඩේ / TRY IT

Button click එකෙන් method එකට ගමන් කරන arrow අඳින්න.

_____ _____

මතක තබාගන්න: Event-driven programming කියන්නේ user actions වලට ප්‍රතිචාර දීම.

සැලකිල්ල: Slow tasks UI thread මත නොකරන්න.

LESSON 063 • CHAPTER 06

Layout Managers

BorderLayout සහ GridLayout භාවිතා කළොත් controls හොඳින් පිළිවෙළට තබන්න පුළුවන්.

අපි උදාහරණයකින් බලමු

Balance cards දෙක columns දෙකක් ලෙස පෙන්වන්න.

VS CODE / TERMINAL — Java / Windows CMD

```
JPanel cards = new JPanel(new GridLayout(1, 2, 12, 4));
cards.add(balanceA);
cards.add(balanceB);
```

ඔබේ වැඩේ / TRY IT

Buttons තීරු තුනක් ලෙස ඇඳන්න.

_____ _____

> **මතක තබාගන්න:** Absolute pixel positions වලට වඩා layout managers හොඳයි.

සැලකිල්ල: Small window test එකක් කරන්න.

LESSON 064 • CHAPTER 06

Error dialog

වැරදි amount එකක් දුන් විට app crash නොවී JOptionPane message එකක් පෙන්වයි.

අපි උදාහරණයකින් බලමු

`JOptionPane.showMessageDialog(...)`.

○○○ VS CODE / TERMINAL — Java / Windows CMD

```
JOptionPane.showMessageDialog(this,
    "Please check the amount",
    "Input error",
    JOptionPane.WARNING_MESSAGE);
```

ඔබේ වැඩේ / TRY IT

Overdraft error එක intentionally trigger කරන්න.

_____ _____

මතක තබාගන්න: Human-readable error = හොඳ UX.

සැලකිල්ල: Technical exception details userට නොපෙන්වන්න.

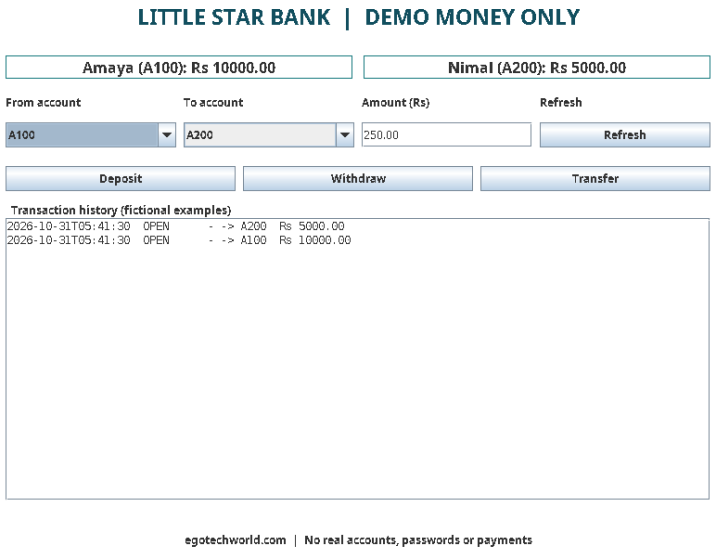
LESSON 065 • CHAPTER 06

Refresh screen

Deposit හෝ transfer සාර්ථක වුණ පසු balance labels හා history එක අලුත් කළ යුතුයි.

අපි උදාහරණයකින් බලමු

`refreshScreen()` තුළ labels set කරනවා.



Actual Java Swing window (fictional bank accounts, rendered during testing)

ඔබේ වැඩේ / TRY IT

Refresh button එක click කර history පරීක්ෂා කරන්න.

_____ _____

මතක තබාගන්න: UI state සහ service state ගැළපිය යුතුයි.

සැලකිල්ල: විශාල apps වල data binding patterns උදව් වෙයි.

LESSON 066 • CHAPTER 06

Swing event thread

Swing UI එක Event Dispatch Thread මත නිර්මාණය කරන්න. `SwingUtilities.invokeLater` උපකාරී.

අපි උදාහරණයකින් බලමු

Main.java තුළ BankWindow පෙන්වීම.

○○○ VS CODE / TERMINAL — Java / Windows CMD

```
SwingUtilities.invokeLater(() -> {
    new BankWindow(service).setVisible(true);
});
```

ඔබේ වැඩේ / TRY IT

UI thread මත ලොකු loop එකක් නොකරන්න හේතුව ලියන්න.

_____ _____

මතක තබාගන්න: Responsive UI සඳහා thread rules වැදගත්.

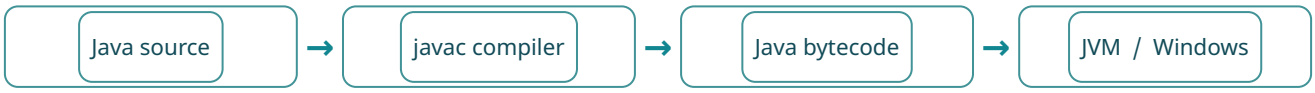
සැලකිල්ල: බාහිර සේවා calls background workers වලට යොදන්න.

CHAPTER 07 / PRACTICAL BUILD

App එක Run කරමු

07

මෙම කොටසේ පාඩම් 9ක් ඉගෙනගෙන අවසානයේ කුඩා වැඩක් කරමු.



> **මෙම කොටසේ ඉගෙනගන්නා මාතෘකා**
>
> ZIP extract කිරීම • CMD විවෘත කිරීම • Compile command • Run command • First deposit lab • Withdraw lab

කේතය කටපාඩම් කරන්න එපා. එක් එක් පියවර මොකක් සඳහාද කියලා සොයා බලන්න.

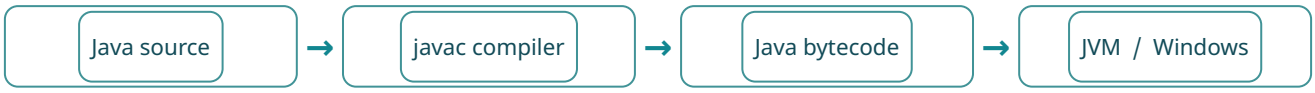
LESSON 067 • CHAPTER 07

ZIP extract කිරීම

Project folder එක ZIP එකෙන් extract කරන්න. Java files සහ Windows batch scripts ඇති බව පරීක්ෂා කරන්න.

අපි උදාහරණයකින් බලමු

README, src, docs, run_windows.bat, test_windows.bat.



ඔබේ වැඩේ / TRY IT

Folder structure එක VS Code Explorer හි බලන්න.

_____ _____

මතක තබාගන්න: Project root එක තුළින් commands run කරන්න.

සැලකිල්ල: අනිත් project files සමඟ overwrite නොකරන්න.

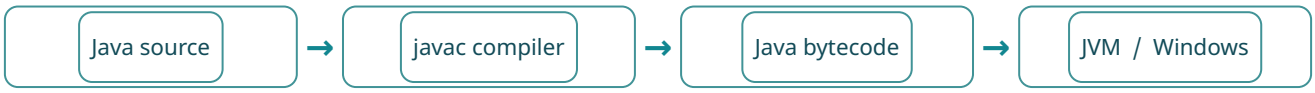
LESSON 068 • CHAPTER 07

CMD විවෘත කිරීම

Windows File Explorer path bar හි cmd ලියා Enter කරන්න; ඒ folder එකේ CMD එක අරිනවා.

අපි උදාහරණයකින් බලමු

`cd C:\\MyProjects\\LittleStarBank`.



ඔබේ වැඩේ / TRY IT

`dir` command එකෙන් Java files තහවුරු කරන්න.

_____ _____

මතක තබාගන්න: නිවැරදි working directory වැදගත්.

සැලකිල්ල: Commands lowercase/uppercase නිවැරදිව බලන්න.

LESSON 069 • CHAPTER 07

Compile command

`javac -encoding UTF-8 -d out src\\egotech\\bank\\*.java` මඟින් Java source compile කරයි.

අපි උදාහරණයකින් බලමු

Compiling කිරීමෙන් `out/` folder එක සෑදේ.

○○○ VS CODE / TERMINAL — Java / Windows CMD

```
javac -encoding UTF-8 -d out src\egotech\bank\*.java
```

ඔබේ වැඩේ / TRY IT

Compile error එකක් තිබුණොත් line number බලන්න.

_____ _____

මතක තබාගන්න: Compile error සහ run-time error වෙනස්.

සැලකිල්ල: Compiler error ignore කර run command දීමෙන් වැඩක් නැහැ.

LESSON 070 • CHAPTER 07

Run command

`java -cp out egotech.bank.Main` මඟින් compiled class එක run කරමු.

අපි උදාහරණයකින් බලමු

Desktop window එක විවෘත වන අතර opening balances දෙක පෙනේ.

○○○ VS CODE / TERMINAL — Java / Windows CMD

```
java -cp out egotech.bank.Main
```

ඔබේ වැඩේ / TRY IT

Window title සහ balance දෙක සටහන් කරන්න.

_____ _____

මතක තබාගන්න: Class path නිවැරදි නම් app එක run වෙයි.

සැලකිල්ල: JDK PATH දෝෂ තිබේ නම් `java -version` පරීක්ෂා කරන්න.

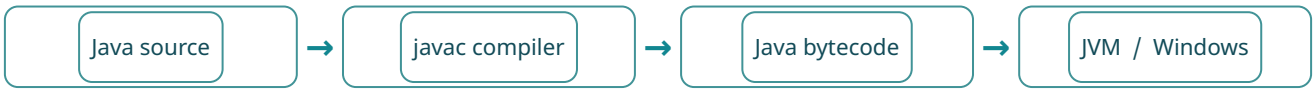
LESSON 071 • CHAPTER 07

First deposit lab

Amount Rs 250, From A100, Deposit click. A100 10250 වෙයි; history row එකත් එකතු වෙයි.

අපි උදාහරණයකින් බලමු

10000 + 250 = 10250.



ඔබේ වැඩේ / TRY IT

Result ලියා expected output සමඟ සසඳන්න.

_____ _____

මතක තබාගන්න: Expected vs actual ලියන එක manual testing.

සැලකිල්ල: වැරදි data history එකට නොයා යුතුයි.

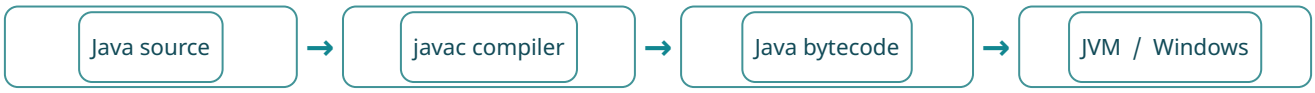
LESSON 072 • CHAPTER 07

Withdraw lab

Deposit පසු A100 10250. Rs 100 withdraw කළොත් 10150 වේ.

අපි උදාහරණයකින් බලමු

10250 - 100 = 10150.



ඔබේ වැඩේ / TRY IT

Withdraw Rs 20000 try කර error බලන්න.

_____ _____

මතක තබාගන්න: Sufficient balance check ප්‍රායෝගිකව බලන්න.

සැලකිල්ල: Real accounts වල withdrawals permissions සහ risk checks ඇත.

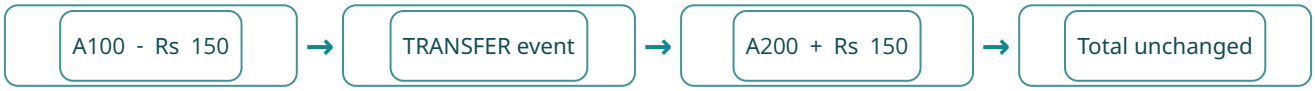
LESSON 073 • CHAPTER 07

Transfer lab

A100 10150 සිට A200 5000 ට Rs 150 යවමු. A100 10000, A200 5150.

අපි උදාහරණයකින් බලමු

Before total 15150, after 15150.



ඔබේ වැඩේ / TRY IT

එකතුව සමානද බලන්න.

_____ _____

මතක තබාගන්න: Transfer දෙපස balance සම්බන්ධයි.

සැලකිල්ල: Journal එකේ single TRANSFER event එකක් පෙනේ.

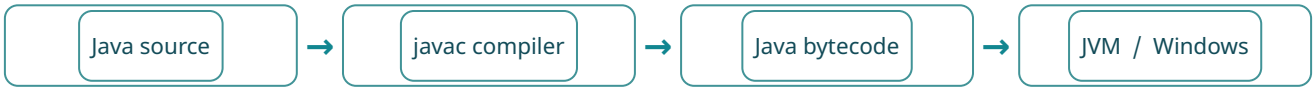
LESSON 074 • CHAPTER 07

Restart lab

App close කර run_windows.bat නැවත run කරන්න. `ledger.tsv` saved events කියවයි.

අපි උදාහරණයකින් බලමු

Previous amounts නැවත පෙන්වෙයි.



ඔබේ වැඩේ / TRY IT

Restart පසු history rows ගණන ලියන්න.

_____ _____

මතක තබාගන්න: Persistence පරීක්ෂාව QA වල වැදගත්.

සැලකිල්ල: File අතින් edit කර test results වෙනස් නොකරන්න.

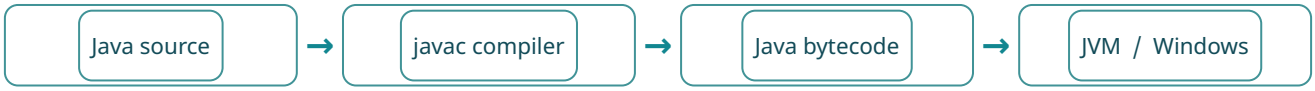
LESSON 075 • CHAPTER 07

Help සහ troubleshooting

`javac` not recognized නම් JDK PATH; `ClassNotFound` නම් classpath හෝ folder; corrupted ledger නම් demo backup restore.

අපි උදාහරණයකින් බලමු

Error message එක copy කර cause හඳුනාගන්න.



ඔබේ වැඩේ / TRY IT

Error එකක් දී එයට solution එකක් සටහන් කරන්න.

_____ _____

මතක තබාගන්න: Errors are clues.

සැලකිල්ල: අහඹු commands හෝ internet script blindly run නොකරන්න.

CHAPTER 08 / QUALITY ASSURANCE

Testing, QA සහ Documentation

08

මෙම කොටසේ පාඩම් 10ක් ඉගෙනගෙන අවසානයේ කුඩා වැඩක් කරමු.



මෙම කොටසේ ඉගෙනගන්නා මාතෘකා

Testing කියන්නේ මොකක්ද? • Unit tests • Integration tests • Test cases ලියන හැටි • Bug report • Regression testing

කේතය කටපාඩම් කරන්න එපා. එක් එක් පියවර මොකක් සඳහාද කියලා සොයා බලන්න.

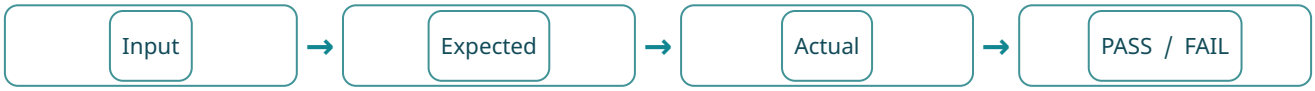
LESSON 076 • CHAPTER 08

Testing කියන්නේ මොකක්ද?

App එක expected result ලබාදෙනවාද කියලා input සහ output සසඳන ක්‍රියාවලිය testing.

අපි උදාහරණයකින් බලමු

Deposit 250 -> 10250.



ඔබේ වැඩේ / TRY IT

Test case එකක expected සහ actual දෙක ලියන්න.

_____ _____

මතක තබාගන්න: Test එකක් proof of absence of all bugs නොවේ.

සැලකිල්ල: Manual + automated tests දෙකම වැදගත්.

LESSON 077 • CHAPTER 08

Unit tests

ඉතා කුඩා logic කොටස් වෙන වෙනම පරීක්ෂා කරමු. Money.parse වැරදි amounts reject කරනවාද බලන්න.

අපි උදාහරණයකින් බලමු

`Money.parse("1.234")` reject.

○○○ VS CODE / TERMINAL — Java / Windows CMD

```
java -cp out egotech.bank.BankTests
// Expected: SUCCESS: 17 tests passed
```

ඔබේ වැඩේ / TRY IT

Zero, negative, 2-decimal amount test inputs 3ක් ලියන්න.

_____ _____

> **මතක තබාගන්න:** Edge cases නිතර bugs හඳුනාගන්නවා.

සැලකිල්ල: No external test framework required in this starter.

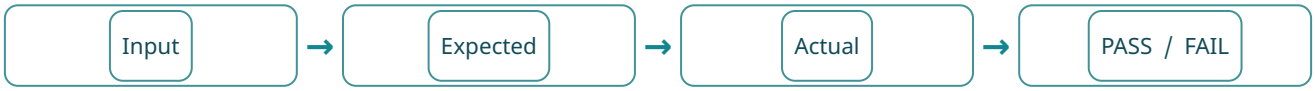
LESSON 078 • CHAPTER 08

Integration tests

BankService සහ LedgerStore එකට වැඩ කර saved state reload වෙනවාද කියා බලමු.

අපි උදාහරණයකින් බලමු

Deposit -> close -> reopen -> same balance.



ඔබේ වැඩේ / TRY IT

App restart scenario එක ලියන්න.

_____ _____

මතක තබාගන්න: Layers එකට ගැළපීමද පරීක්ෂා කරන්න.

සැලකිල්ල: සැබෑ banks වල integration tests දැඩි security යටතේ කරයි.

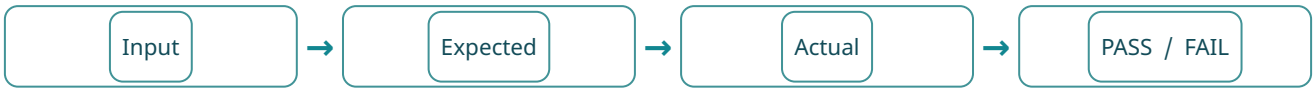
LESSON 079 • CHAPTER 08

Test cases ලියන හැටි

Test ID, precondition, steps, expected, actual, status. QA documentation එකට මේ fields ගැළපේ.

අපි උදාහරණයකින් බලමු

TC09: withdraw too much -> reject/no balance change.



ඔබේ වැඩේ / TRY IT

TC09 complete test case එකක් ලියන්න.

_____ _____

මතක තබාගන්න: Repeatable QA steps ඕනෑම tester කෙනෙකුට අනුගමනය කළ හැක.

සැලකිල්ල: “Works” කියලා පමණක් ලිවීම ප්‍රමාණවත් නැහැ.

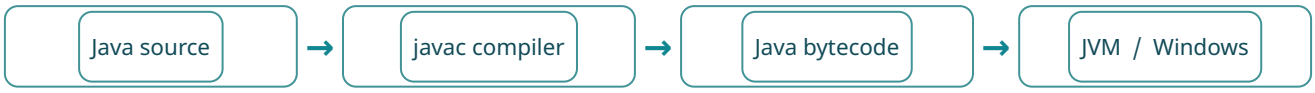
LESSON 080 • CHAPTER 08

Bug report

වැරැද්දක් හමු වූ විට reproducible steps, environment, expected, actual, evidence සටහන් කරන්න.

අපි උදාහරණයකින් බලමු

Example: malformed amount accept වුණොත් defect.



ඔබේ වැඩේ / TRY IT

Sample bug report එකක් සාදන්න.

_____ _____

මතක තබාගන්න: Developerට තේරෙන විදිහට ගැටලුව විස්තර කරන්න.

සැලකිල්ල: රහස් දත්ත screenshot එකක පෙන්වන්න එපා.

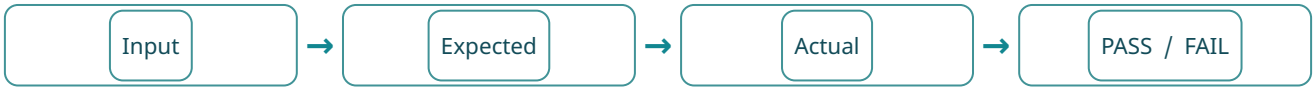
LESSON 081 • CHAPTER 08

Regression testing

Code එක වෙනස් කිරීමෙන් පරණ functionality බිඳෙනවාද බලන්න. Transfer fix කළ පසු deposit ද test කරන්න.

අපි උදාහරණයකින් බලමු

Run BankTests before and after change.



ඔබේ වැඩේ / TRY IT

Method change එකක් කර නැවත tests run කරන්න.

_____ _____

මතක තබාගන්න: Small fixes create new bugs sometimes.

සැලකිල්ල: Tests හැකි නම් automated කරන්න.

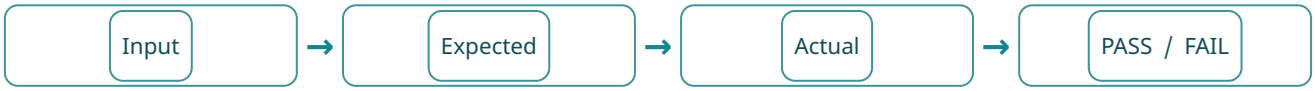
LESSON 082 • CHAPTER 08

Boundary testing

0, 0.01, 999999.99, 1000000.00, 1000000.01 වැනි සීමා අගයන්ට tests ලියන්න.

අපි උදාහරණයකින් බලමු

1000000.01 reject; 0.01 accept.



ඔබේ වැඩේ / TRY IT

Boundary table එකක් හදන්න.

_____ _____

මතක තබාගන්න: අගයන්ගේ සීමා වල bugs බහුලයි.

සැලකිල්ල: Production limits business/legal rules මත තීරණය වෙයි.

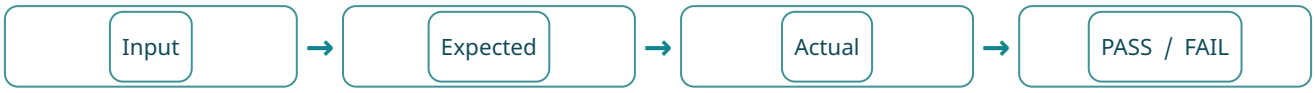
LESSON 083 • CHAPTER 08

Smoke testing

App එක විවෘත වෙනවාද, balances පෙන්වනවාද, buttons වැඩද, close/reopen කළ හැකිද කියලා ඉක්මනින් බලමු.

අපි උදාහරණයකින් බලමු

Start -> deposit -> history -> exit.



ඔබේ වැඩේ / TRY IT

5-step smoke checklist ලියන්න.

_____ _____

> **මතක තබාගන්න:** සෑම release එකකටම smoke test.

සැලකිල්ල: Smoke test එක full QA වෙනුවට නොවේ.

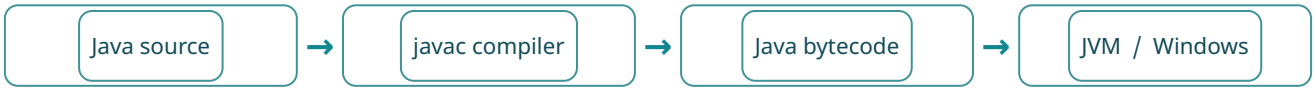
LESSON 084 • CHAPTER 08

QA Report

Test summary, pass/fail count, defects, release risk සහ limitations ඇතුළත් කරන්න.

අපි උදාහරණයකින් බලමු

17 automated tests pass; manual Windows QA independently required.



ඔබේ වැඩේ / TRY IT

QA summary වාක්‍ය පහකින් ලියන්න.

_____ _____

මතක තබාගන්න: Limitations සඟවන්නේ නැතුව ලියන්න.

සැලකිල්ල: “100% bug-free” කියන ප්‍රකාශයට සාධක නැහැ.

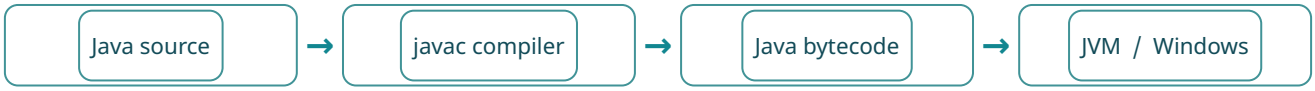
LESSON 085 • CHAPTER 08

SRS review

SRS හි FR01-07 සහ tests අතර සම්බන්ධය බලන්න. Requirement එකක් test නොමැති නම් gap එකක්.

අපි උදාහරණයකින් බලමු

FR03 withdraw balance check -> TC09.



ඔබේ වැඩේ / TRY IT

FR02 deposit සඳහා test case එකක් match කරන්න.

_____ _____

මතක තබාගන්න: Traceability මඟින් අතපසු වීම අඩු කරයි.

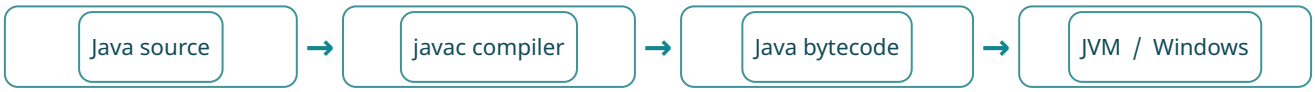
සැලකිල්ල: Scope වෙනස් වුණොත් SRS update කරන්න.

CHAPTER 09 / SAFETY BY DESIGN

Banking Security and Ethics

09

මෙම කොටසේ පාඩම් 10ක් ඉගෙනගෙන අවසානයේ කුඩා වැඩක් කරමු.



> **මෙම කොටසේ ඉගෙනගන්නා මාතෘකා**
>
> Real banking app එකක් නොවේ • Authentication සහ Authorization • Passwords නිවැරදිව • Encryption සහ TLS • ACID transactions • Audit logs

කේතය කටපාඩම් කරන්න එපා. එක් එක් පියවර මොකක් සඳහාද කියලා සොයා බලන්න.

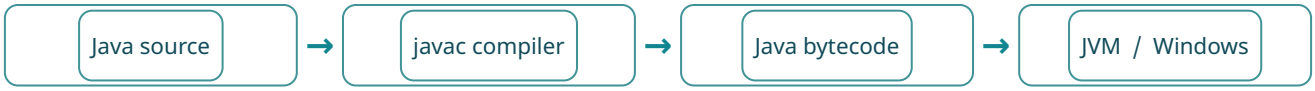
LESSON 086 • CHAPTER 09

Real banking app එකක් නොවේ

Simulator එකට account verification, login, encryption, secure server, real DB transactions නැහැ. Demo balances සැබෑ වටිනාකමක් නෑ.

අපි උදාහරණයකින් බලමු

UI title එකේ DEMO MONEY ONLY පෙන්වයි.



ඔබේ වැඩේ / TRY IT

මෙය real banking සඳහා භාවිත කළ නොහැකි හේතු තුනක් ලියන්න.

_____ _____

මතක තබාගන්න: Learning demo = prototype, not production.

සැලකිල්ල: Real bank account or card data කිසිවිටක ඇතුළත් නොකරන්න.

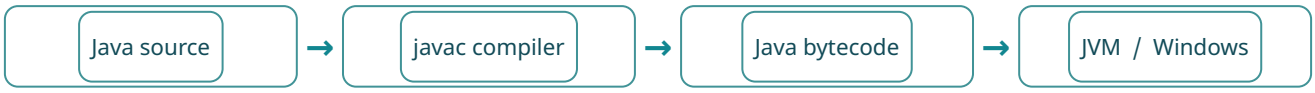
LESSON 087 • CHAPTER 09

Authentication සහ Authorization

Authentication: කවුද කියලා තහවුරු කිරීම. Authorization: එම පුද්ගලයාට කරන්න අවසර තිබේද කියලා පරීක්ෂා කිරීම.

අපි උදාහරණයකින් බලමු

Login success වුණත් වෙන account එකකට access නොලැබිය යුතුයි.



ඔබේ වැඩේ / TRY IT

Authentication හා authorization උදාහරණ දෙකක් ලියන්න.

_____ _____

මතක තබාගන්න: Identity සහ permission වෙනස් සංකල්ප.

සැලකිල්ල: Simulator එකේ දෙකම නැහැ.

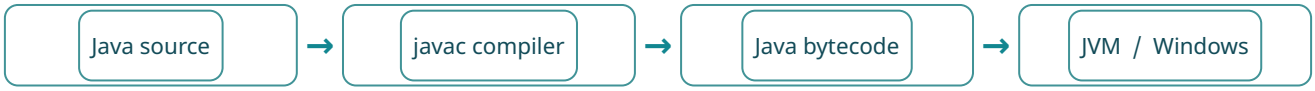
LESSON 088 • CHAPTER 09

Passwords නිවැරදිව

සැබෑ systems වල passwords plain text හෝ CSV file වල නොතබන්න. Dedicated password hashing (Argon2id/bcrypt වැනි) policies අවශ්‍යයි.

අපි උදාහරණයකින් බලමු

මෙම project එක කිසිම password එකක් ඉල්ලා නොසිටී.



ඔබේ වැඩේ / TRY IT

Plain text password file එකක් අනතුරුදායක ඇයි?

_____ _____

මතක තබාගන්න: Security design මුලින්ම සැලසුම් කරන්න.

සැලකිල්ල: SHA-256 hash alone password hashing සඳහා ප්‍රමාණවත් නැහැ.

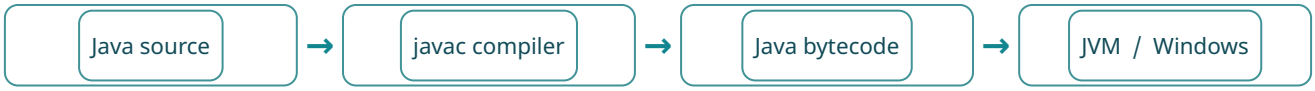
LESSON 089 • CHAPTER 09

Encryption සහ TLS

Network හරහා data යවන විට TLS මඟින් ගමන් කරන data ආරක්ෂා කිරීමට උපකාරී. Storage encryption වෙනම අවශ්‍යයි.

අපි උදාහරණයකින් බලමු

Bank mobile app -> HTTPS API.



ඔබේ වැඩේ / TRY IT

Transit සහ at-rest encryption වෙන්කර ලියන්න.

_____ _____

මතක තබාගන්න: Network security නමින් පමණක් නොලැබේ.

සැලකිල්ල: Certificate warnings නොසලකා නොහරින්න.

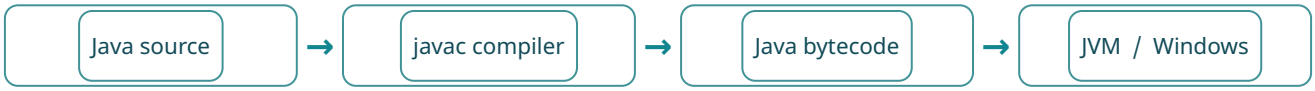
LESSON 090 • CHAPTER 09

ACID transactions

Atomicity, Consistency, Isolation, Durability කියන්නේ DB transactions වල වැදගත් ගුණාංග.

අපි උදාහරණයකින් බලමු

Transfer දෙපස record එකක් ලෙස commit විය යුතුයි.



ඔබේ වැඩේ / TRY IT

ACID අකුරු හතරේ අදහස සිංහලෙන් ලියන්න.

_____ _____

මතක තබාගන්න: Money operations වල partial updates නරකයි.

සැලකිල්ල: අපේ TSV journal එක full ACID නොවේ.

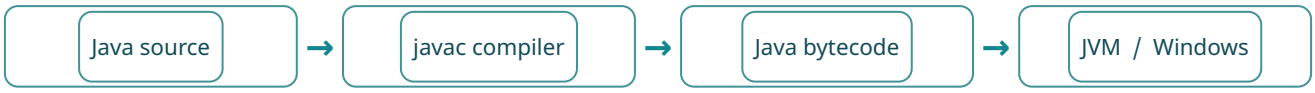
LESSON 091 • CHAPTER 09

Audit logs

කවුද, කවදා, මොකක් කළාද ලියා තැබීම investigations සඳහා උපකාරී. Log tampering වැළැක්වීමත් වැදගත්.

අපි උදාහරණයකින් බලමු

Real service එකක request ID, actor, timestamp වැනි metadata තිබේ.



ඔබේ වැඩේ / TRY IT

Audit log සහ screen history වෙනස ලියන්න.

_____ _____

මතක තබාගන්න: Traceability banking trust එකට වැදගත්.

සැලකිල්ල: Secrets/full card numbers logs වල නොලියන්න.

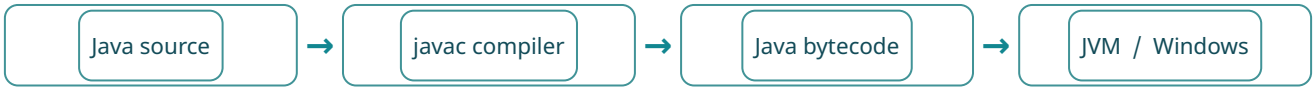
LESSON 092 • CHAPTER 09

Fraud detection

අමුතු spending patterns, repeated failed attempts, unusual locations වැනි signals review කිරීමට systems යොදාගනී.

අපි උදාහරණයකින් බලමු

මෙය high-level topic එකක් පමණයි; simulator fraud check නොකරයි.



ඔබේ වැඩේ / TRY IT

Fraud flag එකකට human review අවශ්‍ය වන්නේ ඇයි?

_____ _____

මතක තබාගන්න: Automated decision errors ඇතිවිය හැක.

සැලකිල්ල: False positives ගැන සැලකිලිමත් වන්න.

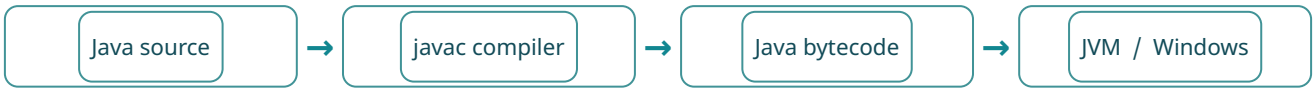
LESSON 093 • CHAPTER 09

Data privacy

සැබෑ customers ලාගේ details purpose එකට අදාළව සීමා කර පමණක් collect/retain කළ යුතුයි.

අපි උදාහරණයකින් බලමු

Demo එකේ නම දෙකක් පමණක්, real national ID numbers නැහැ.



ඔබේ වැඩේ / TRY IT

Fake data භාවිතා කරන වාසිය ලියන්න.

_____ _____

මතක තබාගන්න: Minimize sensitive information.

සැලකිල්ල: Screenshots share කිරීමට පෙර sensitive fields ඉවත් කරන්න.

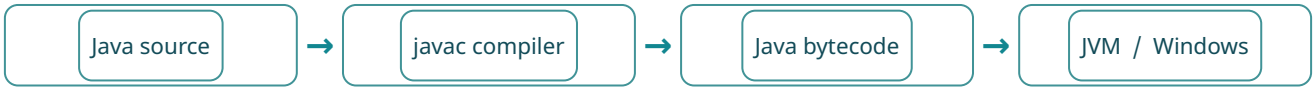
LESSON 094 • CHAPTER 09

2026 Java enterprise ecosystem

Java 25 LTS, Spring Boot වැනි frameworks සහ databases භාවිතයෙන් විශාල backend apps හදනවා. මෙය Java Swing beginner project එකකි.

අපි උදාහරණයකින් බලමු

Desktop prototype -> Java REST API -> DB transaction system කියන ඉදිරි ගමන.



ඔබේ වැඩේ / TRY IT

Spring Boot සහ Swing කාර්යභාර වෙනස ලියන්න.

_____ _____

> **මතක තබාගන්න:** Learning path එකෙන් technologies එකින් එක එකතු කරන්න.

සැලකිල්ල: Production use සඳහා qualified review සහ security testing අවශ්‍යයි.

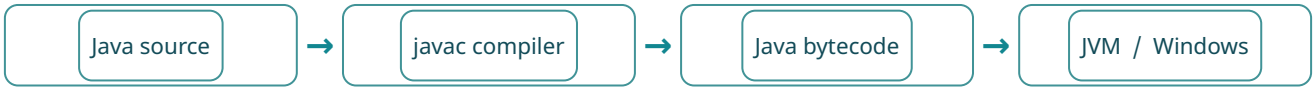
LESSON 095 • CHAPTER 09

Fairness සහ responsibility

Banking software වල bugs මුදල් හා විශ්වාසයට බලපානවා. ඒ නිසා accessible UI, clear fees, record accuracy සහ human support වැදගත්.

අපි උදාහරණයකින් බලමු

Customer balance එක වැරදි පෙන්වීමෙන් ඇතිවන ගැටලු සිතා බලන්න.



ඔබේ වැඩේ / TRY IT

Responsible developer කෙනෙකුගේ පුරුදු පහක් ලියන්න.

_____ _____

මතක තබාගන්න: Software ethics coding එකට සමාන වැදගත්.

සැලකිල්ල: AI-generated banking code audit නොකර deploy නොකරන්න.

CHAPTER 10 / NEXT STEPS AND PRACTICE

Publish නොව Portfolio Demo

10

මෙම කොටසේ පාඩම් 10ක් ඉගෙනගෙන අවසානයේ කුඩා වැඩක් කරමු.



මෙම කොටසේ ඉගෙනගන්නා මාතෘකා

Source control Git • README ලියමු • Release ZIP • JAR file කියන්නේ • Windows exe ගැන හැඳින්වීම • Customer support test

කේතය කටපාඩම් කරන්න එපා. එක් එක් පියවර මොකක් සඳහාද කියලා සොයා බලන්න.

LESSON 096 • CHAPTER 10

Source control Git

Version control මඟින් code changes පසුපසින් බලන්න පුළුවන්. Local repo එකක fictional project එක version කරන්න.

අපි උදාහරණයකින් බලමු

`git init`, `git add .`, `git commit -m "first demo"`.

○○○ VS CODE / TERMINAL — Java / Windows CMD

```
git init
git add .
git commit -m "Add Java demo banking simulator"
```

ඔබේ වැඩේ / TRY IT

`.gitignore` එකෙන් out/ සහ data/ මඟහරින්න.

_____ _____

මතක තබාගන්න: Generated files source repo එකට නිතර අවශ්‍ය නැහැ.

සැලකිල්ල: Secrets කිසිවිටක GitHub වෙත push නොකරන්න.

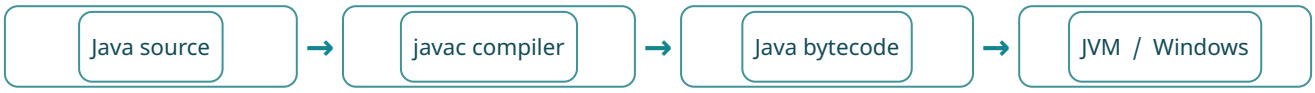
LESSON 097 • CHAPTER 10

README ලියමු

Project setup, requirements, commands, limitations සහ screenshots ඇතුළත් README එකක් හොඳ documentation.

අපි උදාහරණයකින් බලමු

`run_windows.bat` සහ `test_windows.bat` කියන්නේ මොනවාද.



ඔබේ වැඩේ / TRY IT

README එකෙන් යාළුවෙකුට app run කරවන්න.

_____ _____

මතක තබාගන්න: Clear documentation එක project quality එකකි.

සැලකිල්ල: Safety warning එක පැහැදිලිව තබන්න.

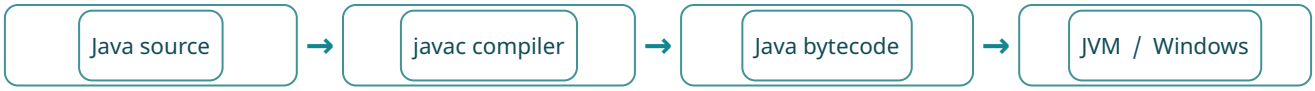
LESSON 098 • CHAPTER 10

Release ZIP

Source, docs, README සහ Windows scripts එක්කර ZIP file එකක් සෑදිය හැක. Built demo history ඉවත් කර seed on first run.

අපි උදාහරණයකින් බලමු

Project folder -> Compress to ZIP.



ඔබේ වැඩේ / TRY IT

ZIP extract කර අලුත් folder එකක compile කරන්න.

_____ _____

මතක තබාගන්න: Fresh-folder test අනිවාර්යයි.

සැලකිල්ල: `out/` compiled classes ZIP එකට අනිවාර්ය නැහැ.

LESSON 099 • CHAPTER 10

JAR file කියන්නේ

JAR මඟින් Java classes එක්කර ගොනුවක් හදන්න පුළුවන්. එය run කිරීමට සුදුසු JVM සහ manifest ඕනේ.

අපි උදාහරණයකින් බලමු

```
jar --create --file LittleStarBank.jar --main-class egotech.bank.Main -C out ..
```

○○○ VS CODE / TERMINAL — Java / Windows CMD

```
jar --create --file LittleStarBank.jar --main-class egotech.bank.Main -C out .
java -jar LittleStarBank.jar
```

ඔබේ වැඩේ / TRY IT

JAR එක Java සමඟ run වන හැටි ලියන්න.

_____ _____

මතක තබාගන්න: JAR = Java archive.

සැලකිල්ල: JAR එක `.exe` නැහැ; Windows app package කරන වෙනත් tools ඇත.

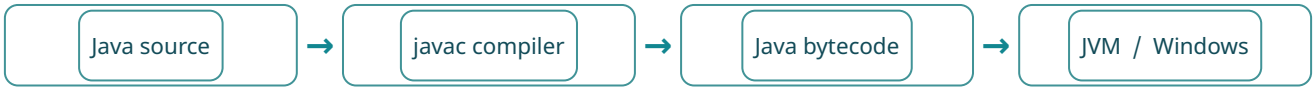
LESSON 100 • CHAPTER 10

Windows exe ගැන හැඳින්වීම

`jpackage` මඟින් supported environment එකේ installable app package සකස් කළ හැක; Windows package Windows OS එකක build කිරීම ප්‍රායෝගිකයි.

අපි උදාහරණයකින් බලමු

Build first -> package -> install test.



ඔබේ වැඩේ / TRY IT

Installer QA අවශ්‍ය හේතුව ලියන්න.

_____ _____

මතක තබාගන්න: Deployment යනු coding ඉවර වූ පසු තවත් අදියරකි.

සැලකිල්ල: Platform-specific options official docs අනුව තෝරන්න.

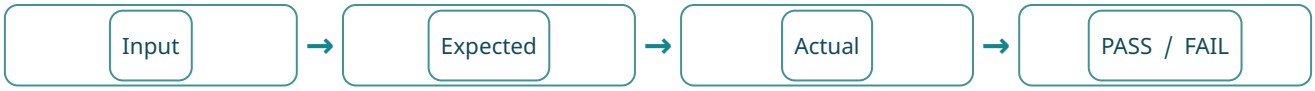
LESSON 101 • CHAPTER 10

Customer support test

Use case සහ issue reporting දැනගන්න: “Balance doesn’t change after deposit” කියන complaint එක review කරන්න.

අපි උදාහරණයකින් බලමු

Steps, transaction entry, expected/actual trace බලන්න.



ඔබේ වැඩේ / TRY IT

Support ticket එකක් sample data වලින් ලියන්න.

_____ _____

මතක තබාගන්න: Developer සහ QA අතර සන්නිවේදනය වැදගත්.

සැලකිල්ල: Real bank customer data support chat වල නොබෙදාහරින්න.

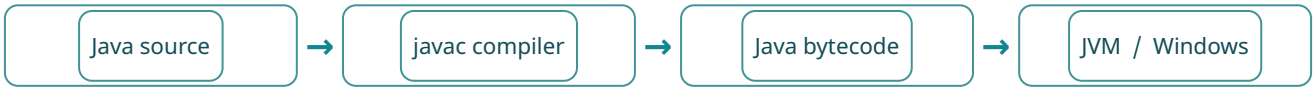
LESSON 102 • CHAPTER 10

Improvement 1: statement filter

Date හෝ transaction type අනුව history filter කිරීමට UI field එකක් එකතු කළ හැක.

අපි උදාහරණයකින් බලමු

TRANSFER events පමණක් පෙන්වීම.



ඔබේ වැඩේ / TRY IT

Pseudo-code හෝ sketch එකක් හදන්න.

_____ _____

මතක තබාගන්න: New features වලට new tests අවශ්‍යයි.

සැලකිල්ල: Recorded history delete නොකර filter කරන්න.

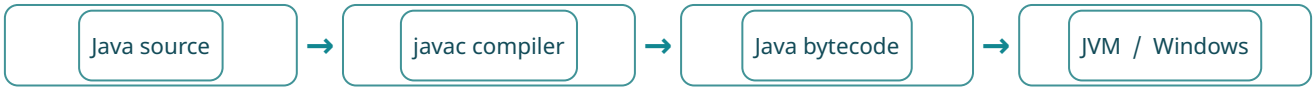
LESSON 103 • CHAPTER 10

Improvement 2: reports

Daily deposit totals වැනි report එකක් lesson extension එකක්. `BigDecimal` sum කිරීම සහ timestamps grouping ඉගෙනගන්න.

අපි උදාහරණයකින් බලමු

2 deposits: 100 + 200 = 300.



ඔබේ වැඩේ / TRY IT

Daily totals සඳහා inputs/outputs සැලසුම් කරන්න.

_____ _____

මතක තබාගන්න: Reporting raw data එකට වෙනස් view එකක්.

සැලකිල්ල: Timestamp timezone policy තීරණය කළ යුතුයි.

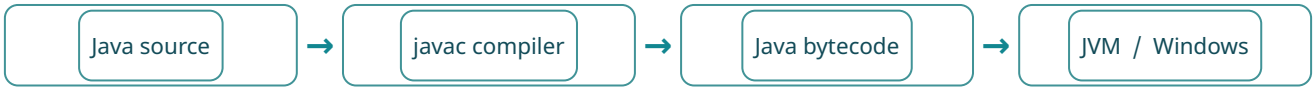
LESSON 104 • CHAPTER 10

Improvement 3: database upgrade

File ledger වෙනුවට JDBC + transaction database එක්කරණය advanced topic එකක්. Connections, SQL parameters සහ transaction commits ඉගෙනගන්න.

අපි උදාහරණයකින් බලමු

Database table: ledger(id, type, from_id, to_id, amount, time).



ඔබේ වැඩේ / TRY IT

SQL table diagram එක අඳින්න.

_____ _____

මතක තබාගන්න: Storage layer වෙන් කළ නිසා replace කිරීම පහසුයි.

සැලකිල්ල: Schema security සහ migrations අවශ්‍යයි.

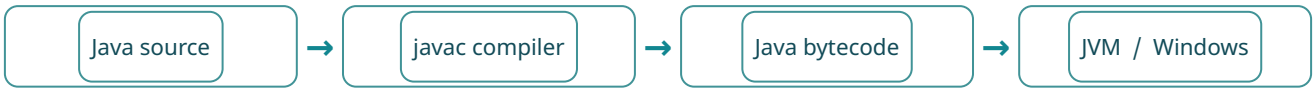
LESSON 105 • CHAPTER 10

සම්පූර්ණ project review

Java ඇයි? BigDecimal ඇයි? Deposit/withdraw/transfer rules මොනවාද? QA proof කොහෙද? මේ ප්‍රශ්නවලට ඔබට පිළිතුරු දිය හැකි නම් සාර්ථකයි.

අපි උදාහරණයකින් බලමු

README, source, tests සහ GUI preview එක්කර portfolio summary ලියන්න.



ඔබේ වැඩේ / TRY IT

මිතුරෙකුට project එක විනාඩි දෙකකින් පැහැදිලි කරන්න.

_____ _____

මතක තබාගන්න: Teach back method මඟින් ඉගෙනගත් දේ තහවුරු වේ.

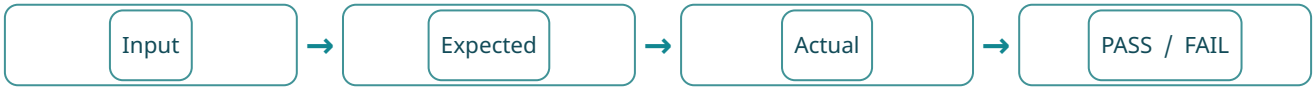
සැලකිල්ල: මෙය certificate of banking compliance එකක් නොවේ.

HANDS-ON LAB 01

Mini Lab A - HelloBank

ගොනුවක් සාදා Java compiler එක වැඩද බලමු.

01	HelloBank.java ගොනුවක් සෑදීම
02	main method සහ println ලිවීම
03	javac HelloBank.java
04	java HelloBank



CHECKPOINTS

- ☐ Message එක පෙනේ
- ☐ Compiler error නැහැ

ඔබේ සටහන් / Result

_____

_____

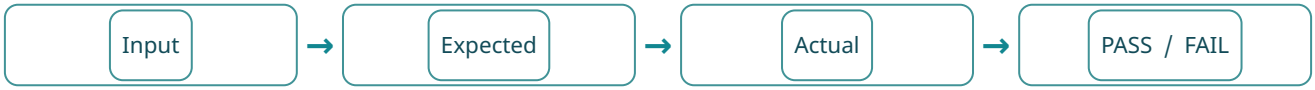
_____

HANDS-ON LAB 02

Mini Lab B - Money class

BigDecimal සහ String input වෙනස බලමු.

01	Money.parse("250.00") success
02	Money.parse("-10") rejects
03	Money.parse("2.345") rejects
04	Money.format(amount) display



CHECKPOINTS

- ☐ 2 decimals
- ☐ invalid amounts reject

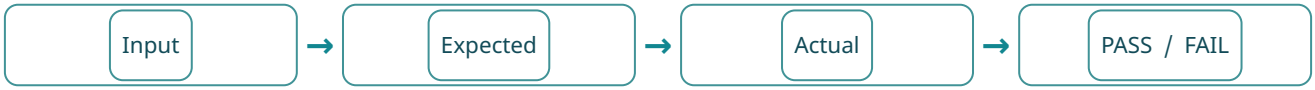
ඔබේ සටහන් / Result

HANDS-ON LAB 03

Mini Lab C - Deposit

A100 ගිණුමට fictional demo money එකතු කරන්න.

01	First run -> Rs 10000
02	From A100, amount 250
03	Click Deposit
04	History row and balance check



CHECKPOINTS

- ☐ A100 = Rs 10250
- ☐ New DEPOSIT event

ඔබේ සටහන් / Result

_____

_____

_____

HANDS-ON LAB 04

Mini Lab D - Withdraw

සීමාව ඉක්මවා මුදල් ඉවත් කිරීම වැළැක්වීම.

01	From A100, amount 100
02	Click Withdraw
03	Balance check
04	Attempt amount 999999

Input	→	Expected	→	Actual	→	PASS / FAIL

CHECKPOINTS

- ☐ Legal withdrawal succeeds
- ☐ Overdraft rejected

ඔබේ සටහන් / Result

HANDS-ON LAB 05

Mini Lab E - Transfer

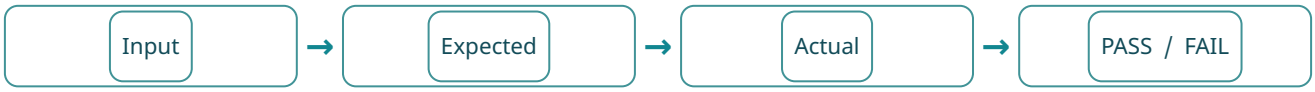
මාරු කළ මුදල නැතිවී නොයන බව පරීක්ෂා කරන්න.

01 From A100 / To A200

02 Amount 150

03 Click Transfer

04 Check both balances and total



CHECKPOINTS

- ☐ Sender decreases
- ☐ Receiver increases
- ☐ Total unchanged

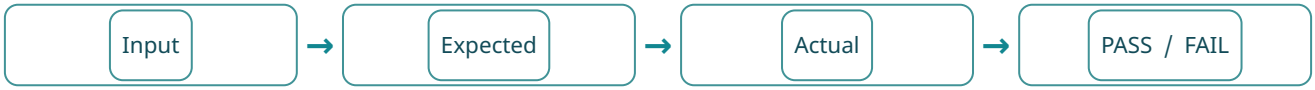
ඔබේ සටහන් / Result

HANDS-ON LAB 06

Mini Lab F - Restart

Demo transaction history file එක reload වේද?

01	Close app
02	Open data/ledger.tsv (read only)
03	Run app again
04	Compare balances and events



CHECKPOINTS

☐ State persisted

☐ No opening duplicates

ඔබේ සටහන් / Result

_____

_____

_____

HANDS-ON LAB 07

Mini Lab G - QA Matrix

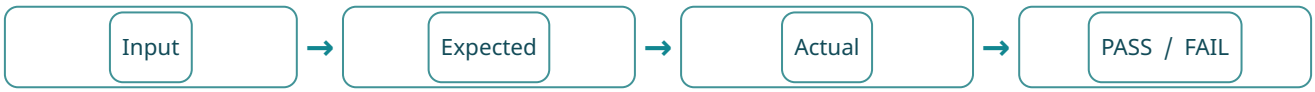
Test plan එකක් ලියන අභ්‍යාසය.

01 Create TC ID

02 Write input and action

03 Write expected output

04 Write pass/fail and notes



CHECKPOINTS

☐ Reproducible steps

☐ Evidence recorded

ඔබේ සටහන් / Result

_____

_____

_____

HANDS-ON LAB 08

Mini Lab H - Security Thinking

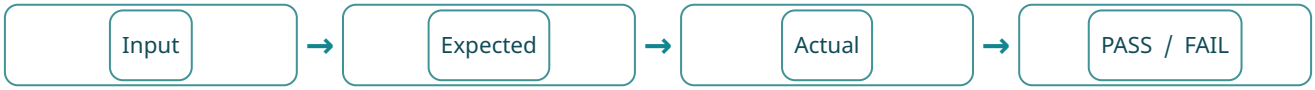
Demo එක real bank එකක් නොවන හේතු විස්තර කරන්න.

01 List missing login

02 List DB transaction needs

03 List privacy rules

04 Describe safe sample data



CHECKPOINTS

☐ No real customer data

☐ Clear limitations

ඔබේ සටහන් / Result

_____

_____

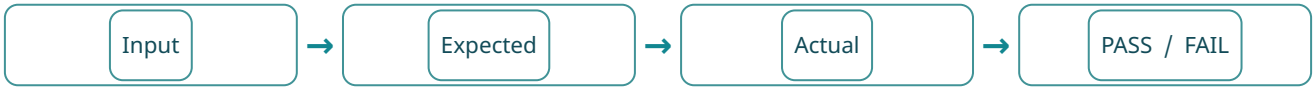
_____

HANDS-ON LAB 09

Mini Lab I - README

ඔබේම source ZIP එකේ උපදෙස් පරීක්ෂා කරන්න.

01	Fresh folder extract
02	Check java -version
03	Compile sources
04	Run app and tests



CHECKPOINTS

☐ No missing files

☐ Works from clean folder

ඔබේ සටහන් / Result

_____

_____

_____

HANDS-ON LAB 10

Mini Lab J - Explain Back

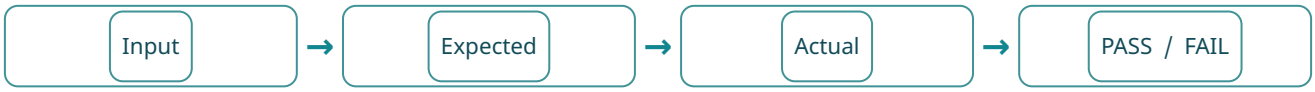
ඔබට යාළුවෙකුට ව්‍යාපෘතිය විස්තර කළ හැකිද?

01 Why Java?

02 Why BigDecimal?

03 How transfer works?

04 Which tests passed?



CHECKPOINTS

☐ Can explain without copy

☐ Can demo safely

ඔබේ සටහන් / Result

_____

_____

_____

QA • REAL CHECK

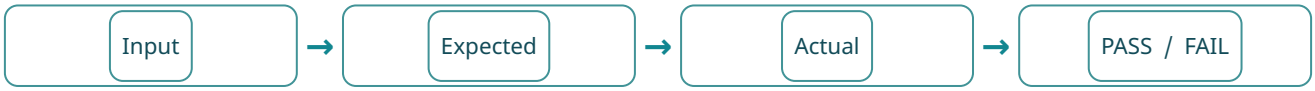
Automated Tests Run කළ විට

Project source එක සමඟ දී ඇති BankTests.java තුළ initial balances, deposit, withdraw, transfer, invalid amount සහ restart persistence වැනි cases තිබෙනවා.

○○○ VS CODE / TERMINAL — Windows CMD

```
javac -encoding UTF-8 -d out src\egotech\bank\*.java
java -cp out egotech.bank.BankTests

PASS: initial account A
PASS: deposit
PASS: transfer conserves funds
PASS: restart events persisted
SUCCESS: 17 tests passed
```



සෑම පරිගණකයකම manual GUI testing වෙනම කරන්න. මේ output එක sample environment එකේ tests වලින් ලැබුණු ප්‍රතිඵලයක් ලෙස සලකන්න.

SOURCE CODE APPENDIX • JAVA

BankService.java • කොටස 1/5

මෙය download කළ source ZIP එකේ `src/egotech/bank/BankService.java` ගොනුවේ සැබෑ code එකයි. වෙනස් කළ පසු tests නැවත run කරන්න.

○○○ VS CODE / TERMINAL — BankService.java

```
package egotech.bank;

import java.io.IOException;
import java.math.BigDecimal;
import java.nio.file.Path;
import java.time.Instant;
import java.util.ArrayList;
import java.util.LinkedHashMap;
import java.util.List;
import java.util.Map;

/** Business rules for fictional accounts only. */
public final class BankService {
    private static final String ALICE = "A100";
    private static final String BOB = "A200";
    private final LedgerStore store;
    private final List<LedgerEntry> events = new ArrayList<>();
    private final Map<String, BigDecimal> balances = new LinkedHashMap<>();

    public BankService(Path path) throws IOException {
        store = new LedgerStore(path);
        balances.put(ALICE, BigDecimal.ZERO.setScale(2));
        balances.put(BOB, BigDecimal.ZERO.setScale(2));
        if (!store.exists()) {
            post(new LedgerEntry(Instant.now(), "OPEN", "-", ALICE, new BigDecimal("10000.00")));
```

> **Practice:** මෙම කොටසේ class, method හෝ validation line එකක් හඳුනාගෙන එය කරන වැඩ වාක්‍යයකින් ලියන්න.

SOURCE CODE APPENDIX • JAVA

BankService.java • කොටස 2/5

මෙය download කළ source ZIP එකේ `src/egotech/bank/BankService.java` ගොනුවේ සැබෑ code එකයි. වෙනස් කළ පසු tests නැවත run කරන්න.

○○○ VS CODE / TERMINAL — BankService.java

```
            post(new LedgerEntry(Instant.now(), "OPEN", "-", BOB, new BigDecimal("5000.00")));
        } else {
            for (LedgerEntry entry : store.load()) {
                apply(entry);
                events.add(entry);
          }
       }
    }

    public Map<String, String> accounts() {
        return Map.of(ALICE, "Amaya (A100)", BOB, "Nimal (A200)");
    }

    private void requireAccount(String id) {
        if (!balances.containsKey(id)) throw new IllegalArgumentException("Unknown account");
    }

    public synchronized BigDecimal balance(String id) {
        requireAccount(id);
        return balances.get(id);
    }

    public synchronized List<LedgerEntry> history() {
        return List.copyOf(events);
    }
```

> **Practice:** මෙම කොටසේ class, method හෝ validation line එකක් හඳුනාගෙන එය කරන වැඩ වාක්‍යයකින් ලියන්න.

SOURCE CODE APPENDIX • JAVA

BankService.java • කොටස 3/5

මෙය download කළ source ZIP එකේ `src/egotech/bank/BankService.java` ගොනුවේ සැබෑ code එකයි. වෙනස් කළ පසු tests නැවත run කරන්න.

VS CODE / TERMINAL — BankService.java

```
    public synchronized void deposit(String id, String amount) throws IOException {
        requireAccount(id);
        post(new LedgerEntry(Instant.now(), "DEPOSIT", "-", id, Money.parse(amount)));
    }

    public synchronized void withdraw(String id, String amount) throws IOException {
        requireAccount(id);
        BigDecimal value = Money.parse(amount);
        if (balance(id).compareTo(value) < 0) {
            throw new IllegalArgumentException("Insufficient demo balance");
        }
        post(new LedgerEntry(Instant.now(), "WITHDRAW", id, "-", value));
    }

    public synchronized void transfer(String from, String to, String amount) throws IOException {
        requireAccount(from);
        requireAccount(to);
        if (from.equals(to)) throw new IllegalArgumentException("Choose a different recipient");
        BigDecimal value = Money.parse(amount);
        if (balance(from).compareTo(value) < 0) {
            throw new IllegalArgumentException("Insufficient demo balance");
        }
        // A single journal line describes BOTH sides of this demo transfer.
        post(new LedgerEntry(Instant.now(), "TRANSFER", from, to, value));
```

> **Practice:** මෙම කොටසේ class, method හෝ validation line එකක් හඳුනාගෙන එය කරන වැඩ වාක්‍යයකින් ලියන්න.

SOURCE CODE APPENDIX • JAVA

BankService.java • කොටස 4/5

මෙය download කළ source ZIP එකේ `src/egotech/bank/BankService.java` ගොනුවේ සැබෑ code එකයි. වෙනස් කළ පසු tests නැවත run කරන්න.

VS CODE / TERMINAL — BankService.java

```
    }

    private void post(LedgerEntry entry) throws IOException {
        // Save first. If disk writing fails, do not change the screen balance.
        store.append(entry);
        apply(entry);
        events.add(entry);
    }

    private void apply(LedgerEntry entry) {
        BigDecimal amount = entry.amount();
        switch (entry.type()) {
            case "OPEN", "DEPOSIT" -> change(entry.to(), amount);
            case "WITHDRAW" -> change(entry.from(), amount.negate());
            case "TRANSFER" -> {
                change(entry.from(), amount.negate());
                change(entry.to(), amount);
          }
            default -> throw new IllegalArgumentException("Bad entry");
       }
    }

    private void change(String id, BigDecimal diff) {
        requireAccount(id);
        BigDecimal result = balances.get(id).add(diff);
```

> **Practice:** මෙම කොටසේ class, method හෝ validation line එකක් හඳුනාගෙන එය කරන වැඩ වාක්‍යයකින් ලියන්න.

SOURCE CODE APPENDIX • JAVA

BankService.java • කොටස 5/5

මෙය download කළ source ZIP එකේ `src/egotech/bank/BankService.java` ගොනුවේ සැබෑ code එකයි. වෙනස් කළ පසු tests නැවත run කරන්න.

○○○ VS CODE / TERMINAL — BankService.java

```
        if (result.signum() < 0) throw new IllegalArgumentException("Invalid history: negative
balance");
        balances.put(id, result);
    }
}
```

> **Practice:** මෙම කොටසේ class, method හෝ validation line එකක් හඳුනාගෙන එය කරන වැඩ වාක්‍යයකින් ලියන්න.

SOURCE CODE APPENDIX • JAVA

BankTests.java • කොටස 1/3

මෙය download කළ source ZIP එකේ `src/egotech/bank/BankTests.java` ගොනුවේ සැබෑ code එකයි. වෙනස් කළ පසු tests නැවත run කරන්න.

VS CODE / TERMINAL — BankTests.java

```
package egotech.bank;

import java.math.BigDecimal;
import java.nio.file.Files;
import java.nio.file.Path;

/** Run with: java -cp out egotech.bank.BankTests */
public final class BankTests {
    private static int passed;

    private static void check(boolean ok, String name) {
        if (!ok) throw new AssertionError("FAIL: " + name);
        passed++;
        System.out.println("PASS: " + name);
    }

    private static void rejects(Runnable work, String name) {
        boolean rejected = false;
        try { work.run(); } catch (IllegalArgumentException e) { rejected = true; }
        check(rejected, name);
    }

    public static void main(String[] args) throws Exception {
        Path dir = Files.createTempDirectory("egotech-bank-tests");
        try {
```

> **Practice:** මෙම කොටසේ class, method හෝ validation line එකක් හඳුනාගෙන එය කරන වැඩ වාක්‍යයකින් ලියන්න.

SOURCE CODE APPENDIX • JAVA

BankTests.java • කොටස 2/3

මෙය download කළ source ZIP එකේ `src/egotech/bank/BankTests.java` ගොනුවේ සැබෑ code එකයි. වෙනස් කළ පසු tests නැවත run කරන්න.

○○○ VS CODE / TERMINAL — BankTests.java

```
            Path file = dir.resolve("ledger.tsv");
            BankService bank = new BankService(file);
            check(bank.balance("A100").compareTo(new BigDecimal("10000.00")) == 0, "initial account
A");
            check(bank.balance("A200").compareTo(new BigDecimal("5000.00")) == 0, "initial account
B");
            bank.deposit("A100", "250.00");
            check(bank.balance("A100").compareTo(new BigDecimal("10250.00")) == 0, "deposit");
            bank.withdraw("A100", "100.00");
            check(bank.balance("A100").compareTo(new BigDecimal("10150.00")) == 0, "withdraw");
            bank.transfer("A100", "A200", "150.00");
            check(bank.balance("A100").compareTo(new BigDecimal("10000.00")) == 0, "transfer
sender");
            check(bank.balance("A200").compareTo(new BigDecimal("5150.00")) == 0, "transfer
receiver");
            check(bank.balance("A100").add(bank.balance("A200"))
                    .compareTo(new BigDecimal("15150.00")) == 0, "transfer conserves funds");
            rejects(() -> Money.parse("-5"), "reject negative");
            rejects(() -> Money.parse("0"), "reject zero");
            rejects(() -> Money.parse("1.234"), "reject 3 decimals");
            rejects(() -> Money.parse("abc"), "reject letters");
            rejects(() -> { try { bank.withdraw("A100", "900000.00"); }
                            catch (java.io.IOException e) { throw new RuntimeException(e); } },
"reject overdraft");
            rejects(() -> { try { bank.transfer("A100", "A100", "20.00"); }
                            catch (java.io.IOException e) { throw new RuntimeException(e); } },
"reject same account");
            check(bank.history().size() == 5, "number of journal events");
            BankService reloaded = new BankService(file);
            check(reloaded.balance("A100").compareTo(bank.balance("A100")) == 0, "restart A
persisted");
            check(reloaded.balance("A200").compareTo(bank.balance("A200")) == 0, "restart B
persisted");
```

> **Practice:** මෙම කොටසේ class, method හෝ validation line එකක් හඳුනාගෙන එය කරන වැඩ වාක්‍යයකින් ලියන්න.

SOURCE CODE APPENDIX • JAVA

BankTests.java • කොටස 3/3

මෙය download කළ source ZIP එකේ `src/egotech/bank/BankTests.java` ගොනුවේ සැබෑ code එකයි. වෙනස් කළ පසු tests නැවත run කරන්න.

○○○ VS CODE / TERMINAL — BankTests.java

```
            check(reloaded.history().size() == 5, "restart events persisted");
            System.out.println("SUCCESS: " + passed + " tests passed");
        } finally {
            try (var stream = Files.walk(dir)) {
                stream.sorted(java.util.Comparator.reverseOrder()).forEach(p -> {
                    try { Files.deleteIfExists(p); } catch (Exception e) { /* temp only */ }
                });
            }
        }
    }
}
```

> **Practice:** මෙම කොටසේ class, method හෝ validation line එකක් හඳුනාගෙන එය කරන වැඩ වාක්‍යයකින් ලියන්න.

SOURCE CODE APPENDIX • JAVA

BankWindow.java • කොටස 1/6

මෙය download කළ source ZIP එකේ `src/egotech/bank/BankWindow.java` ගොනුවේ සැබෑ code එකයි. වෙනස් කළ පසු tests නැවත run කරන්න.

○○○ VS CODE / TERMINAL — BankWindow.java

```
package egotech.bank;

import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Font;
import java.awt.GridLayout;
import java.util.Comparator;
import java.util.List;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.JTextField;
import javax.swing.SwingConstants;
import javax.swing.border.EmptyBorder;

/** Simple Swing screen for the fictional Little Star Bank. */
public final class BankWindow extends JFrame {
    private final BankService service;
    private final JComboBox<String> from = new JComboBox<>(new String[] {"A100", "A200"});
    private final JComboBox<String> to = new JComboBox<>(new String[] {"A200", "A100"});
```

> **Practice:** මෙම කොටසේ class, method හෝ validation line එකක් හඳුනාගෙන එය කරන වැඩ වාක්‍යයකින් ලියන්න.

SOURCE CODE APPENDIX • JAVA

BankWindow.java • කොටස 2/6

මෙය download කළ source ZIP එකේ `src/egotech/bank/BankWindow.java` ගොනුවේ සැබෑ code එකයි. වෙනස් කළ පසු tests නැවත run කරන්න.

○○○ VS CODE / TERMINAL — BankWindow.java

```
    private final JTextField amount = new JTextField("250.00");
    private final JLabel balanceA = new JLabel();
    private final JLabel balanceB = new JLabel();
    private final JTextArea history = new JTextArea(14, 64);

    public BankWindow(BankService service) {
        super("Little Star Bank | Learning simulator");
        this.service = service;
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(800, 620);
        setLocationRelativeTo(null);
        setLayout(new BorderLayout(12, 12));
        getContentPane().setBackground(Color.WHITE);

        JLabel title = new JLabel("LITTLE STAR BANK  |  DEMO MONEY ONLY", SwingConstants.CENTER);
        title.setFont(new Font("SansSerif", Font.BOLD, 23));
        title.setForeground(new Color(18, 79, 95));
        title.setBorder(new EmptyBorder(20, 10, 4, 10));
        add(title, BorderLayout.NORTH);

        JPanel main = new JPanel(new BorderLayout(10, 12));
        main.setBackground(Color.WHITE);
        main.setBorder(new EmptyBorder(10, 20, 14, 20));
        JPanel cards = new JPanel(new GridLayout(1, 2, 12, 4));
        cards.setBackground(Color.WHITE);
```

> **Practice:** මෙම කොටසේ class, method හෝ validation line එකක් හඳුනාගෙන එය කරන වැඩ වාක්‍යයකින් ලියන්න.

SOURCE CODE APPENDIX • JAVA

BankWindow.java • කොටස 3/6

මෙය download කළ source ZIP එකේ `src/egotech/bank/BankWindow.java` ගොනුවේ සැබෑ code එකයි. වෙනස් කළ පසු tests නැවත run කරන්න.

○○○ VS CODE / TERMINAL — BankWindow.java

```
        styleBalance(balanceA);
        styleBalance(balanceB);
        cards.add(balanceA);
        cards.add(balanceB);
        JPanel topPanel = new JPanel(new BorderLayout(12, 12));
        topPanel.setBackground(Color.WHITE);
        topPanel.add(cards, BorderLayout.NORTH);

        JPanel actions = new JPanel(new GridLayout(2, 4, 8, 8));
        actions.setBackground(Color.WHITE);
        actions.add(new JLabel("From account"));
        actions.add(new JLabel("To account"));
        actions.add(new JLabel("Amount (Rs)"));
        actions.add(new JLabel("Refresh"));
        actions.add(from);
        actions.add(to);
        actions.add(amount);
        JButton refresh = new JButton("Refresh");
        refresh.addActionListener(e -> refreshScreen());
        actions.add(refresh);

        JPanel buttons = new JPanel(new GridLayout(1, 3, 8, 8));
        buttons.setBackground(Color.WHITE);
        buttons.add(button("Deposit", "DEPOSIT"));
        buttons.add(button("Withdraw", "WITHDRAW"));
```

> **Practice:** මෙම කොටසේ class, method හෝ validation line එකක් හඳුනාගෙන එය කරන වැඩ වාක්‍යයකින් ලියන්න.

SOURCE CODE APPENDIX • JAVA

BankWindow.java • කොටස 4/6

මෙය download කළ source ZIP එකේ `src/egotech/bank/BankWindow.java` ගොනුවේ සැබෑ code එකයි. වෙනස් කළ පසු tests නැවත run කරන්න.

○○○ VS CODE / TERMINAL — BankWindow.java

```
        buttons.add(button("Transfer", "TRANSFER"));

        JPanel controls = new JPanel(new BorderLayout(8, 10));
        controls.setBackground(Color.WHITE);
        controls.add(actions, BorderLayout.NORTH);
        controls.add(buttons, BorderLayout.SOUTH);
        topPanel.add(controls, BorderLayout.CENTER);
        main.add(topPanel, BorderLayout.NORTH);

        history.setFont(new Font("Monospaced", Font.PLAIN, 12));
        history.setEditable(false);
        history.setBackground(Color.WHITE);
        JPanel lower = new JPanel(new BorderLayout());
        lower.setBackground(Color.WHITE);
        lower.add(new JLabel("  Transaction history (fictional examples)"), BorderLayout.NORTH);
        lower.add(new JScrollPane(history), BorderLayout.CENTER);
        main.add(lower, BorderLayout.CENTER);
        add(main, BorderLayout.CENTER);

        JLabel footer = new JLabel("egotechworld.com  |  No real accounts, passwords or payments",
SwingConstants.CENTER);
        footer.setBorder(new EmptyBorder(8, 4, 12, 4));
        add(footer, BorderLayout.SOUTH);
        refreshScreen();
    }
```

> **Practice:** මෙම කොටසේ class, method හෝ validation line එකක් හඳුනාගෙන එය කරන වැඩ වාක්‍යයකින් ලියන්න.

SOURCE CODE APPENDIX • JAVA

BankWindow.java • කොටස 5/6

මෙය download කළ source ZIP එකේ `src/egotech/bank/BankWindow.java` ගොනුවේ සැබෑ code එකයි. වෙනස් කළ පසු tests නැවත run කරන්න.

○○○ VS CODE / TERMINAL — BankWindow.java

```
    private void styleBalance(JLabel label) {
        label.setFont(new Font("SansSerif", Font.BOLD, 16));
        label.setHorizontalAlignment(SwingConstants.CENTER);
        label.setBorder(javax.swing.BorderFactory.createLineBorder(new Color(18, 119, 124)));
    }

    private JButton button(String label, String operation) {
        JButton b = new JButton(label);
        b.addActionListener(e -> execute(operation));
        return b;
    }

    private void execute(String operation) {
        try {
            String id = (String) from.getSelectedItem();
            String recipient = (String) to.getSelectedItem();
            switch (operation) {
                case "DEPOSIT" -> service.deposit(id, amount.getText());
                case "WITHDRAW" -> service.withdraw(id, amount.getText());
                case "TRANSFER" -> service.transfer(id, recipient, amount.getText());
                default -> throw new IllegalArgumentException("Unsupported operation");
          }
            refreshScreen();
            JOptionPane.showMessageDialog(this, "Demo transaction completed.");
        } catch (Exception ex) {
```

> **Practice:** මෙම කොටසේ class, method හෝ validation line එකක් හඳුනාගෙන එය කරන වැඩ වාක්‍යයකින් ලියන්න.

SOURCE CODE APPENDIX • JAVA

BankWindow.java • කොටස 6/6

මෙය download කළ source ZIP එකේ `src/egotech/bank/BankWindow.java` ගොනුවේ සැබෑ code එකයි. වෙනස් කළ පසු tests නැවත run කරන්න.

○○○ VS CODE / TERMINAL — BankWindow.java

```
            JOptionPane.showMessageDialog(this, ex.getMessage(), "Check your input",
JOptionPane.WARNING_MESSAGE);
        }
    }

    private void refreshScreen() {
        balanceA.setText("Amaya (A100): " + Money.format(service.balance("A100")));
        balanceB.setText("Nimal (A200): " + Money.format(service.balance("A200")));
        List<LedgerEntry> entries = service.history();
        StringBuilder output = new StringBuilder();
        for (int i = entries.size() - 1; i >= 0; i--) {
            LedgerEntry e = entries.get(i);
            output.append(e.at().toString(), 0, 19).append("  ")
                .append(String.format("%-8s", e.type()))
                .append("  ").append(e.from()).append(" -> ").append(e.to())
                .append("  ").append(Money.format(e.amount())).append("\n");
        }
        history.setText(output.toString());
    }
}
```

> **Practice:** මෙම කොටසේ class, method හෝ validation line එකක් හඳුනාගෙන එය කරන වැඩ වාක්‍යයකින් ලියන්න.

SOURCE CODE APPENDIX • JAVA

LedgerEntry.java • කොටස 1/2

මෙය download කළ source ZIP එකේ `src/egotech/bank/LedgerEntry.java` ගොනුවේ සැබෑ code එකයි. වෙනස් කළ පසු tests නැවත run කරන්න.

○○○ VS CODE / TERMINAL — LedgerEntry.java

```
package egotech.bank;

import java.math.BigDecimal;
import java.time.Instant;

/** One demo transaction. It is not a real financial ledger. */
public record LedgerEntry(Instant at, String type, String from,
                          String to, BigDecimal amount) {
    public LedgerEntry {
        if (!java.util.Set.of("OPEN", "DEPOSIT", "WITHDRAW", "TRANSFER").contains(type)) {
            throw new IllegalArgumentException("Unknown transaction type");
       }
        if (amount == null || amount.signum() <= 0 || amount.scale() != 2) {
            throw new IllegalArgumentException("Invalid stored amount");
       }
   }

    public String serialize() {
        // Tabs separate fields; our fixed account IDs contain no tabs.
        return at.toString() + "\t" + type + "\t" + from + "\t" + to
                + "\t" + amount.toPlainString();
   }

    public static LedgerEntry deserialize(String line) {
        String[] fields = line.split("\\t", -1);
```

> **Practice:** මෙම කොටසේ class, method හෝ validation line එකක් හඳුනාගෙන එය කරන වැඩ වාක්‍යයකින් ලියන්න.

SOURCE CODE APPENDIX • JAVA

LedgerEntry.java • කොටස 2/2

මෙය download කළ source ZIP එකේ `src/egotech/bank/LedgerEntry.java` ගොනුවේ සැබෑ code එකයි. වෙනස් කළ පසු tests නැවත run කරන්න.

○○○ VS CODE / TERMINAL — LedgerEntry.java

```
        if (fields.length != 5) throw new IllegalArgumentException("Bad data row");
        return new LedgerEntry(Instant.parse(fields[0]), fields[1], fields[2],
                fields[3], new BigDecimal(fields[4]));
    }
}
```

> **Practice:** මෙම කොටසේ class, method හෝ validation line එකක් හඳුනාගෙන එය කරන වැඩ වාක්‍යයකින් ලියන්න.

SOURCE CODE APPENDIX • JAVA

LedgerStore.java • කොටස 1/2

මෙය download කළ source ZIP එකේ `src/egotech/bank/LedgerStore.java` ගොනුවේ සැබෑ code එකයි. වෙනස් කළ පසු tests නැවත run කරන්න.

○○○ VS CODE / TERMINAL — LedgerStore.java

```
package egotech.bank;

import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.StandardOpenOption;
import java.util.ArrayList;
import java.util.List;

/** A small plain-text journal for learning only. Not an ACID database. */
public final class LedgerStore {
    private final Path path;

    public LedgerStore(Path path) { this.path = path; }

    public boolean exists() { return Files.exists(path); }

    public List<LedgerEntry> load() throws IOException {
        List<LedgerEntry> items = new ArrayList<>();
        if (!Files.exists(path)) return items;
        for (String line : Files.readAllLines(path, StandardCharsets.UTF_8)) {
            if (!line.isBlank()) items.add(LedgerEntry.deserialize(line));
      }
        return items;
```

> **Practice:** මෙම කොටසේ class, method හෝ validation line එකක් හඳුනාගෙන එය කරන වැඩ වාක්‍යයකින් ලියන්න.

SOURCE CODE APPENDIX • JAVA

LedgerStore.java • කොටස 2/2

මෙය download කළ source ZIP එකේ `src/egotech/bank/LedgerStore.java` ගොනුවේ සැබෑ code එකයි. වෙනස් කළ පසු tests නැවත run කරන්න.

○○○ VS CODE / TERMINAL — LedgerStore.java

```
    }

    public void append(LedgerEntry entry) throws IOException {
        if (path.getParent() != null) Files.createDirectories(path.getParent());
        // This is a teaching example; production requires transactional storage.
        Files.writeString(path, entry.serialize() + System.lineSeparator(),
                StandardCharsets.UTF_8, StandardOpenOption.CREATE, StandardOpenOption.APPEND);
    }
}
```

> **Practice:** මෙම කොටසේ class, method හෝ validation line එකක් හඳුනාගෙන එය කරන වැඩ වාක්‍යයකින් ලියන්න.

SOURCE CODE APPENDIX • JAVA

Main.java • කොටස 1/1

මෙය download කළ source ZIP එකේ `src/egotech/bank/Main.java` ගොනුවේ සැබෑ code එකයි. වෙනස් කළ පසු tests නැවත run කරන්න.

○○○ VS CODE / TERMINAL — Main.java

```
package egotech.bank;

import java.nio.file.Path;
import javax.swing.SwingUtilities;

/** Start the Little Star Bank learning simulator. */
public final class Main {
    private Main() { }

    public static void main(String[] args) {
        // Store demo data in the project directory (not real money).
        Path file = Path.of("data", "ledger.tsv");
        try {
            BankService service = new BankService(file);
            SwingUtilities.invokeLater(() -> new BankWindow(service).setVisible(true));
        } catch (Exception exception) {
            System.err.println("Cannot start demo: " + exception.getMessage());
            System.exit(1);
       }
   }
}
```

> **Practice:** මෙම කොටසේ class, method හෝ validation line එකක් හඳුනාගෙන එය කරන වැඩ වාක්‍යයකින් ලියන්න.

SOURCE CODE APPENDIX • JAVA

Money.java • කොටස 1/1

මෙය download කළ source ZIP එකේ `src/egotech/bank/Money.java` ගොනුවේ සැබෑ code එකයි. වෙනස් කළ පසු tests නැවත run කරන්න.

VS CODE / TERMINAL — Money.java

```
package egotech.bank;

import java.math.BigDecimal;
import java.math.RoundingMode;

/** Use decimal arithmetic instead of floating-point money. */
public final class Money {
    private Money() { }

    public static BigDecimal parse(String input) {
        if (input == null || !input.trim().matches("[0-9]+(\\.[0-9]{1,2})?")) {
            throw new IllegalArgumentException("Enter a positive amount, e.g. 150.00");
       }
        BigDecimal amount = new BigDecimal(input.trim());
        if (amount.signum() <= 0 || amount.compareTo(new BigDecimal("1000000.00")) > 0) {
            throw new IllegalArgumentException("Amount must be between 0.01 and 1000000.00");
       }
        return amount.setScale(2, RoundingMode.UNNECESSARY);
    }

    public static String format(BigDecimal amount) {
        return "Rs " + amount.setScale(2, RoundingMode.UNNECESSARY).toPlainString();
    }
}
```

> **Practice:** මෙම කොටසේ class, method හෝ validation line එකක් හඳුනාගෙන එය කරන වැඩ වාක්‍යයකින් ලියන්න.

2026 • OFFICIAL REFERENCES

තව ඉගෙනගන්න නිල සබැඳි

Java releases
https://www.oracle.com/java/technologies/downloads/

Java 25 API documentation
https://docs.oracle.com/en/java/javase/25/docs/api/

Java Swing tutorial
https://docs.oracle.com/javase/tutorial/uiswing/

BigDecimal
https://docs.oracle.com/en/java/javase/25/docs/api/java.base/java/math/BigDecimal.html

OWASP
https://cheatsheetseries.owasp.org/

VS Code Java
https://code.visualstudio.com/docs/languages/java

Java version note

October 2026 වනවිට Java 27 නවතම සාමාන්‍ය feature release එකයි; Java 25 නවතම LTS එකයි. අපේ project එක Java 21+ සඳහා ලියලා තිබෙනවා.

වෙනත් tutorial එකක් තෝරාගන්න කලින් ඔබේම අවබෝධයෙන් Java code කියවා test කර බලන්න.

FINAL CHECKLIST

ඔබේ Java journey සම්පූර්ණයිද?

- ☐ Java භාවිතා කරන්නේ ඇයි කියා පැහැදිලි කළ හැක
- ☐ JDK, JVM, javac සහ java commands දන්නවා
- ☐ Variables, methods, classes සහ if/else තේරෙනවා
- ☐ Money සඳහා BigDecimal ගැළපෙන බව දන්නවා
- ☐ Deposit/Withdraw/Transfer logic test කළා
- ☐ Desktop Swing window ධාවනය කළා
- ☐ Saved history restart කළ පසු පෙන්වනවා
- ☐ Unit tests සහ manual QA වෙනස දන්නවා
- ☐ SRS, SDLC සහ test case නිර්මාණය කළා
- ☐ සැබෑ banking system එකකට අවශ්‍ය security අවබෝධයි

```
CODE → TEST → DOCUMENT → IMPROVE
```

අද ඔබ banking app එකක් ඉගෙනගත්තා. හෙට එම ක්‍රමය library, school, POS හෝ inventory system එකකටත් යොදාගන්න පුළුවන්.

EGOTECHWORLD • Learn by Building • 2026