JAVA PROGRAMMING

Manual Book

For Beginners

2nd Edition

By
Tin Mai Zaw

Published in 2025

Northern City, Yangon, Myanmar

အမှာစကား

ဤစာအုပ်သည် Java Programming ကို အခြေခံမှစတင် လေ့လာမည့် သူများအတွက် လက်စွဲစာအုပ်ဖြစ်ပါတယ်။ ဒုတိယအကြိမ် မြန်မာလို ထုတ်ဝေခြင်းဖြစ်ပါတယ်။ Java Reference Books, Online Reference Website တွေအပြင် Chat GPT, Deepseek AI tools တွေသုံးပြီး အချို့သင်ခန်းစာတွေကို ထပ်မံဖြည့်စွက်ထားပါတယ်။ ကွန်ပျူတာကျောင်း သူကျောင်းသားများ၊ အင်ဂျင်နီယာအိုင်တီ ကျောင်းသူကျောင်းသားများကို တစ်ဖက်တစ်လမ်း အထောက်အကူပြုစေရန်အတွက် ရည်ရွယ် ထုတ်ဝေရခြင်းလည်းဖြစ်ပါတယ်။ အိုင်တီနဲ့ အသက်မွေးဝမ်းကြောင်းပြုလိုသူများ ၊ software programmer ဖြစ်ချင်သူများ၊ ကမ္ဘာကျော် software application တွေတီထွင်ပြီး millionaire သူဌေးဖြစ်ဖို့ စိတ်ကူးအိမ်မက်ရှိသူ မည်သူမဆို လွယ်လွယ်ကူကူ လေ့လာနိုင်အောင် လေ့ကျင့်ခန်း ဥပမာများ ၊ လက်တွေ့ real life application သင်ခန်းစာ များနဲ့ တွဲပြီးတော့ ပြည့်ပြည့်စုံစုံ ရေးသားဖန်းတီးထားတဲ့ စာအုပ်ဖြစ်ပါတယ်။

💻 ဘာလို့ Java ကို လေ့လာသင့်သလဲ?

Java Programming Language ကို ကမ္ဘာတစ်ဝှမ်းမှာ အသုံးများရတဲ့ အဓိက အကြောင်းရင်းတွေကတော့ Platform Independent ဖြစ်လို့ပါ။ ဘယ် OS မှာ မဆို အလုပ်လုပ်နိုင်ပါတယ်။ "Write Once, Run Anywhere" ဆိုတဲ့ အားသာချက်ရှိပါတယ်။ JVM (Java Virtual Machine) ကြောင့် Windows, Mac, Linux စသည့် OS အားလုံးမှာ run နိုင်ပါတယ်။

Object-Oriented Programming (OOP) ကို အပြည့်အဝ ထောက်ပံ့နိုင်တဲ့အတွက် ကြီးမားတဲ့ large-scale applications, နိုင်ငံတော် project တွေနဲ့ အထူးသင့်တော်တဲ့ language ဖြစ်ပါတယ်။

အားကောင်းတဲ့ Memory Management တွေကို လုပ်နိုင်ပါတယ်။ Automatic Garbage Collection စနစ်ပါဝင်တာကြောင့် memory leak ဖြစ်နိုင်ခြေနည်းပါတယ်။ Developer တွေအနေနဲ့ memory ကို manual manage လုပ်စရာမလိုပဲ အလိုအလျောက် ရှင်းလင်းပေးတာ ဖြစ်ပါတယ်။

အင်မတန်လုံခြုံစိတ်ချရတဲ့ Programming language တစ်ခု ဖြစ်ပါတယ်။ Pointer concept မပါဝင်တာကြောင့် လုံခြုံမှုပိုမြင့်မားတယ်။ Security Manager နဲ့ Bytecode Verifier တို့ကြောင့် malicious code တွေကို ကောင်းကောင်း ကာကွယ်နိုင်တယ်။

Multithreading Support လုပ်ပေးတဲ့အတွက် Java နဲ့ ရေးထားတဲ့ app တွေရဲ့ Performance တွေက အရမ်းကို smooth ဖြစ်ပါတယ်။ ပြောချင်တာက အရမ်းကို stable ဖြစ်ပြီး error တွေ application crash ဖြစ်တာတွေ အခြား language တွေထက်စာရင် အရမ်းနည်းပါတယ်။ Built-in multithreading support ပါဝင်တာကြောင့် concurrent programming လုပ်ရတာ အဆင်ပြေတယ်။ High-performance applications တွေဖန်တီးဖို့ အထောက်အကူဖြစ်စေတယ်။

Rich API Library ရှိတဲ့ language တစ်ခုဖြစ်ပါတယ်။ Networking, Database Connection, XML Processing, Utilities စတဲ့ built-in libraries အများအပြားရှိနေပါတယ်။ Third-party libraries တွေလည်း အများကြီးရှိနေတာကြောင့် ဘယ်လိုမျိုး app တွေရေးမလဲ၊ ဘယ်လိုမျိုး services

တွေပေးမလဲ။ ဘာလာလာ အကုန်ဒေါင်းလို့ရတဲ့ စွယ်စုံသုံး programming language တစ်ခုဖြစ်ပါတယ်။

Enterprise-Level Application တွေအတွက်လည်း အထူးသင့်တော်ပါတယ်။ Banking systems, E-commerce platforms စတဲ့ large-scale applications တွေမှာ အများဆုံးအသုံးပြုကြတာ ဖြစ်ပါတယ်။ Spring, Hibernate စတဲ့ powerful frameworks တွေရှိပါတယ်။ Android Development အတွက်အခြေခံ အဖြစ်လေ့လာထားသင့်တဲ့ language လည်းဖြစ်ပါတယ်။ Android OS က Java/Kotlin ကိုအခြေခံထားတာကြောင့် mobile app development အတွက်အရေးကြီးပါတယ်။

ကမ္ဘာ့အကြီးဆုံး Community Support ရှိပါတယ်။ StackOverflow, GitHub စတဲ့နေရာတွေမှာ help resources အများကြီးရှိပါတယ်။ Learning Career အတွက်ပဲ ဖြစ်ဖြစ် ၊ Developer Career အတွက်ပဲ ဖြစ်ဖြစ် ပြဿနာတစ်ခုကြုံတိုင်း solution ရှာဖို့အရမ်းလွယ်ကူပါတယ်။

Backward Compatibility ရှိတဲ့အတွက် နှစ်ပေါင်းများစွာ update လုပ်လာပေမယ့် ဆက်လက်ပြီး လာမယ့်အနာဂါတ်မှာ java code တွေရဲ့ compatibility ကိုထိန်းသိမ်းထားပေးပါတယ်။ ဆိုလိုတာက ရှေးဟောင်း Java code တွေကိုပါ အခုထိ run နိုင်ဆဲဖြစ်ပါတယ်။ အဲဒါကြောင့် Java ဟာ နှစ်ပေါင်း 20 ကျော်ကြာ အသုံးပြုလာကြတဲ့အတွက် ရင့်ကျက်ပြီး stable ဖြစ်တဲ့ language တစ်ခုဖြစ်ပါတယ်။ ဒါကြောင့် enterprise applications တွေ၊ financial systems တွေ၊ government systems တွေမှာ Java ကို အဓိကထားအသုံးပြုကြတာဖြစ်ပါတယ်။

💻 Eclipse Installation

Eclipse IDE က Java Developer တွေအတွက် အထူးရည်ရွယ်ပြီး ထုတ်လုပ်ထားတဲ့အတွက် အကျိုးကျေးဇူးတွေ အများကြီး ရှိပါတယ်။ ပထမအချက်ကတော့ အခမဲ့နဲ့ Open Source ဖြစ်နေတာပဲဖြစ်ပါတယ်။ Eclipse က လုံးဝအခမဲ့ဖြစ်ပြီး Open Source ဖြစ်တာကြောင့် ငွေကြေးအကုန်အကျမရှိဘဲ သုံးနိုင်ပါတယ်။

Cross-Platform support လုပ်တဲ့အတွက် Windows, macOS, Linux စတဲ့ OS အမျိုးမျိုးမှာ အလုပ်လုပ်နိုင်ပါတယ်။ ပြီးပြည့်စုံတဲ့ Java Development Tools လည်းဖြစ်ပါတယ်။ Java SE, Java EE, Jakarta EE, Android Development စတာတွေအတွက် ပြည့်စုံတဲ့ Toolset ပါဝင်ပါတယ်။ Code Completion, Refactoring, Debugging စတဲ့ Features တွေ Built-in ပါပါတယ်။

Plugin Ecosystem ကြီးမားခြင်းကြောင့် Maven, Gradle, Git, Spring, Hibernate စတဲ့ Framework တွေနဲ့ အလွယ်တကူချိတ်ဆက်နိုင်ပါတယ်။ ကိုယ်လိုချင်တဲ့ Functionality တွေအတွက Plugin တွေ အများကြီးရှိပါတယ်။ စွမ်းဆောင်ရည်မြင့်မားခြင်းကြောင့် ကြီးမားတဲ့ Project တွေကိုတောင် ထိထိရောက်ရောက် Manage လုပ်နိုင်ပါတယ်။

ဟုတ်ကဲ့ IntelliJ IDEA (အထူးသဖြင့် Ultimate Edition) က ပိုမိုကောင်းမွန်ပေမယ့် Paid Version ဖြစ်ပါတယ်။ NetBeans ကလည်း အခမဲ့ဖြစ်ပေမယ့် Eclipse လောက် Plugin Ecosystem မကြီးမားပါဘူး။ အဲဒါကြောင့် Java Beginner မှ Advanced Developer အထိ၊ အထူးသဖြင့် Enterprise Java Development လုပ်သူများ၊ Plugin-based Custom Development လုပ်ချင်သူများ။ ဒီစာအုပ်မှာ Eclipse IDE ကို သုံးပြီး လေ့ကျင့်ခန်းတွေကို ရေးပြသွားမှာ ဖြစ်ပါတယ်။

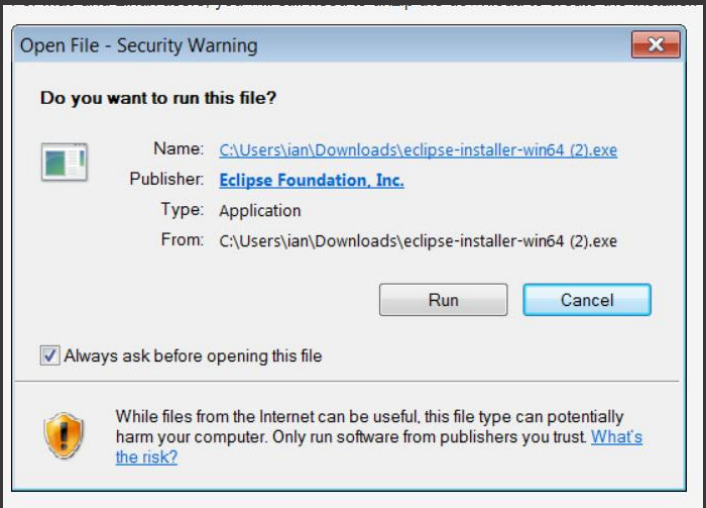
Installation step တွေကို အောက်ပါအတိုင်း အဆင့်ဆင့် လုပ်ဆောင်ပါ။

Step 1:

Eclipse installer ကို ဒေါင်းလုပ်ဆွဲပါ။

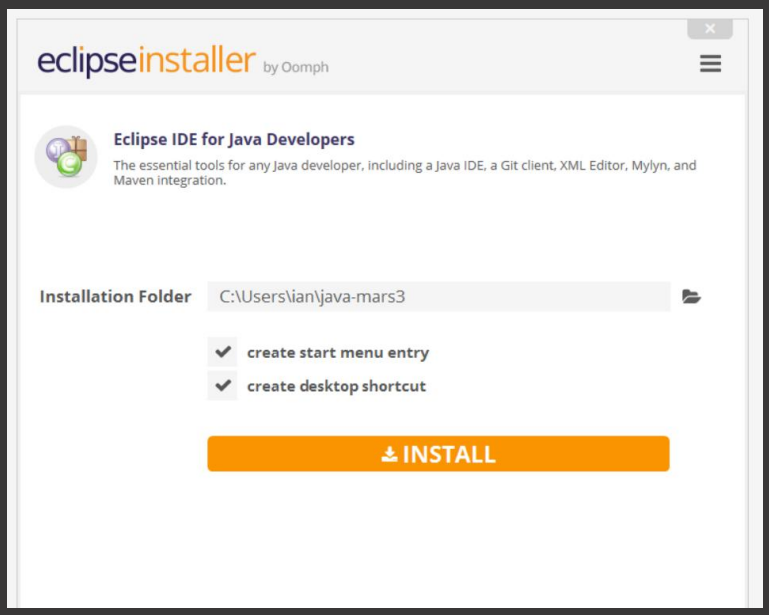
Download Eclipse Installer from http://www.eclipse.org/downloads

Step 2:

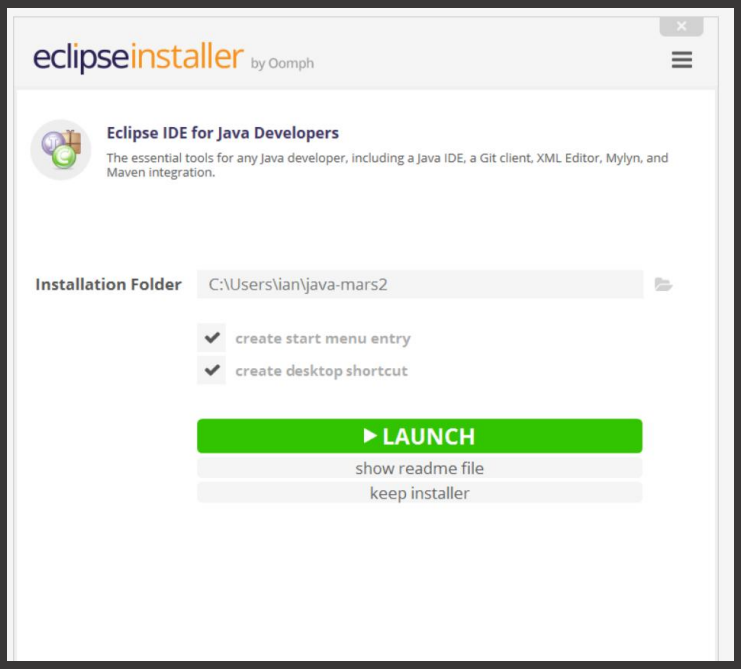


Step 3:

Step 4:

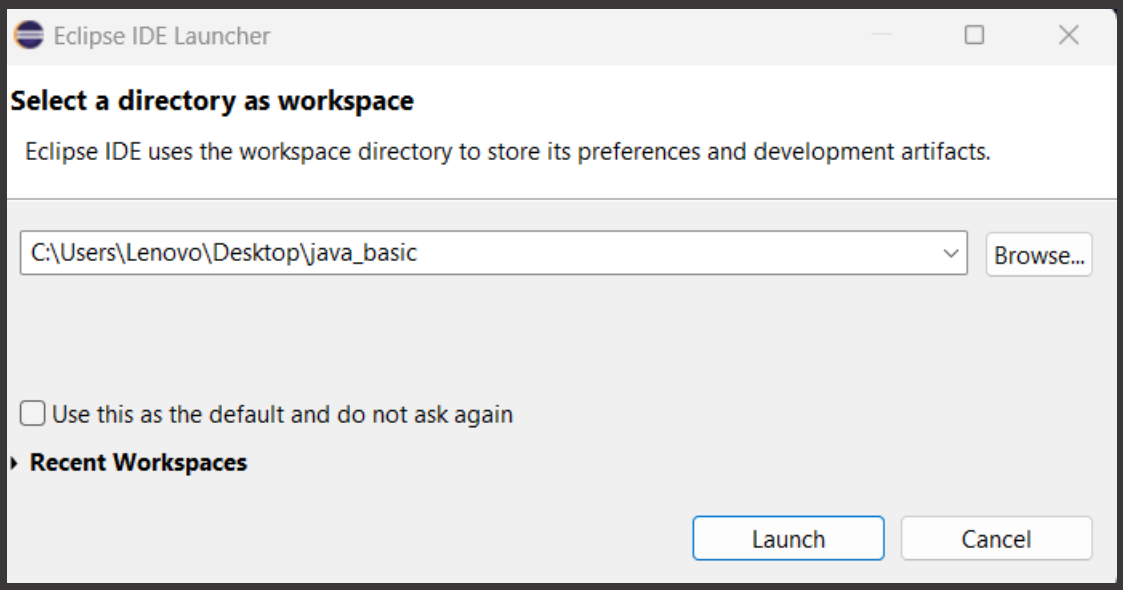

eclipseinstaller by Oomph
Eclipse IDE for Java Developers
The essential tools for any Java developer, including a Java IDE, a Git client, XML Editor, Mylyn, and Maven integration.
Installation Folder
C:\Users\ian\java-mars3
create start menu entry
create desktop shortcut
INSTALL

Step 5:

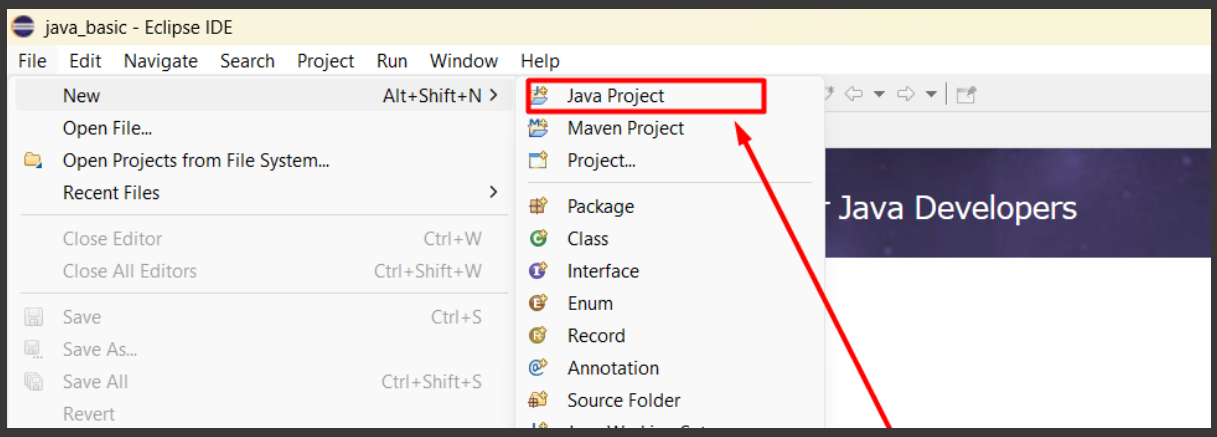

eclipseinstaller by Oomph
Eclipse IDE for Java Developers
The essential tools for any Java developer, including a Java IDE, a Git client, XML Editor, Mylyn, and Maven integration.
Installation Folder
C:\Users\ian\java-mars2
create start menu entry
create desktop shortcut
LAUNCH
show readme file
keep installer

Step 6:

လေ့ကျင့်ခန်းတွေကို သိမ်းမယ့် location ကိုရွေးရပါမယ်။ အခု က desktop မှာ java_basic folder မှာ သိမ်းဖို့ ညွှန်းထားပါတယ်။

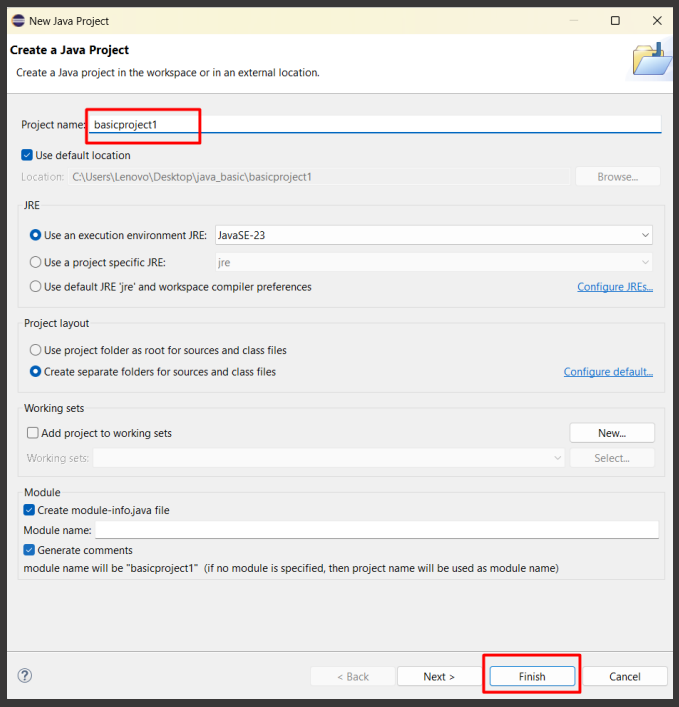


Step 7:



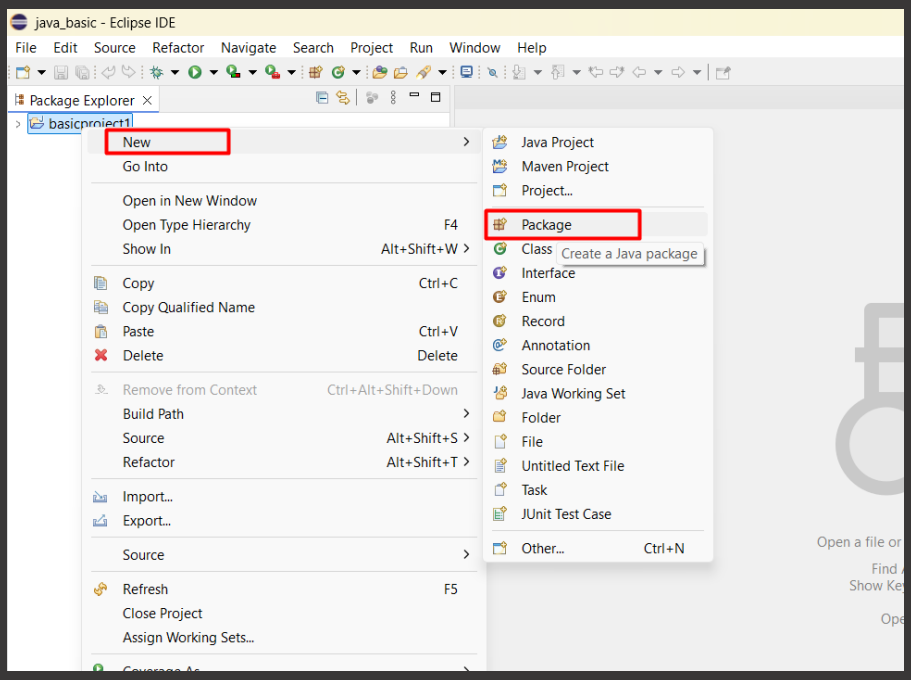
Step 8:

Project name ကို small letter နဲ့ရေးရပါမယ်။



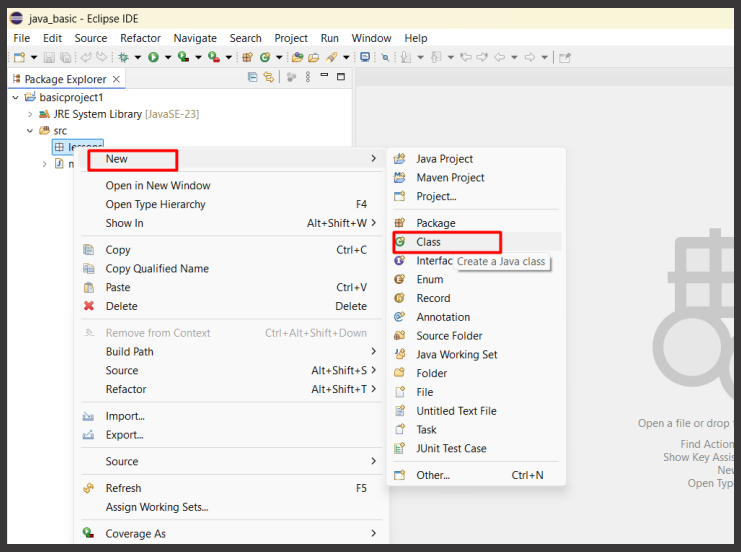
Step 9:

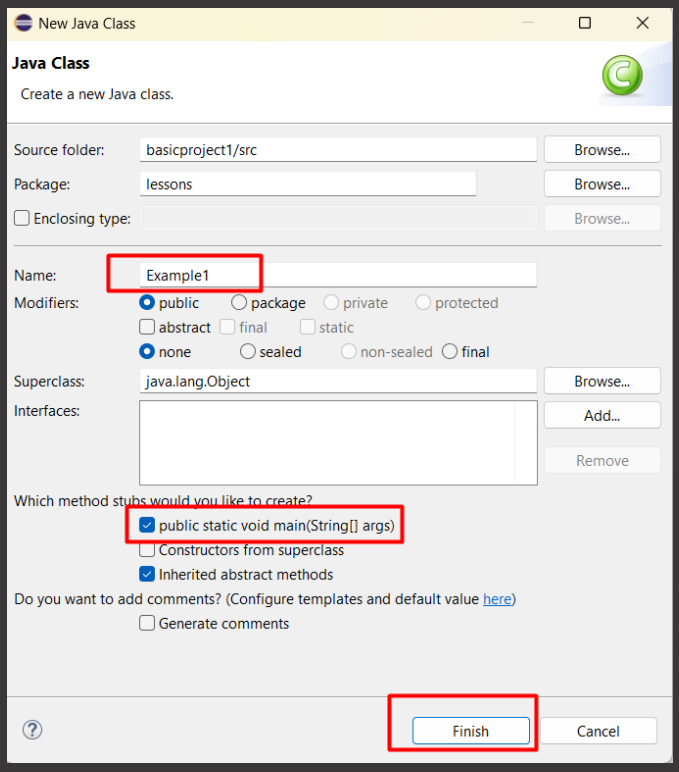
Package name ကိုလည်း small letter နဲ့ ရေးရပါမယ်။



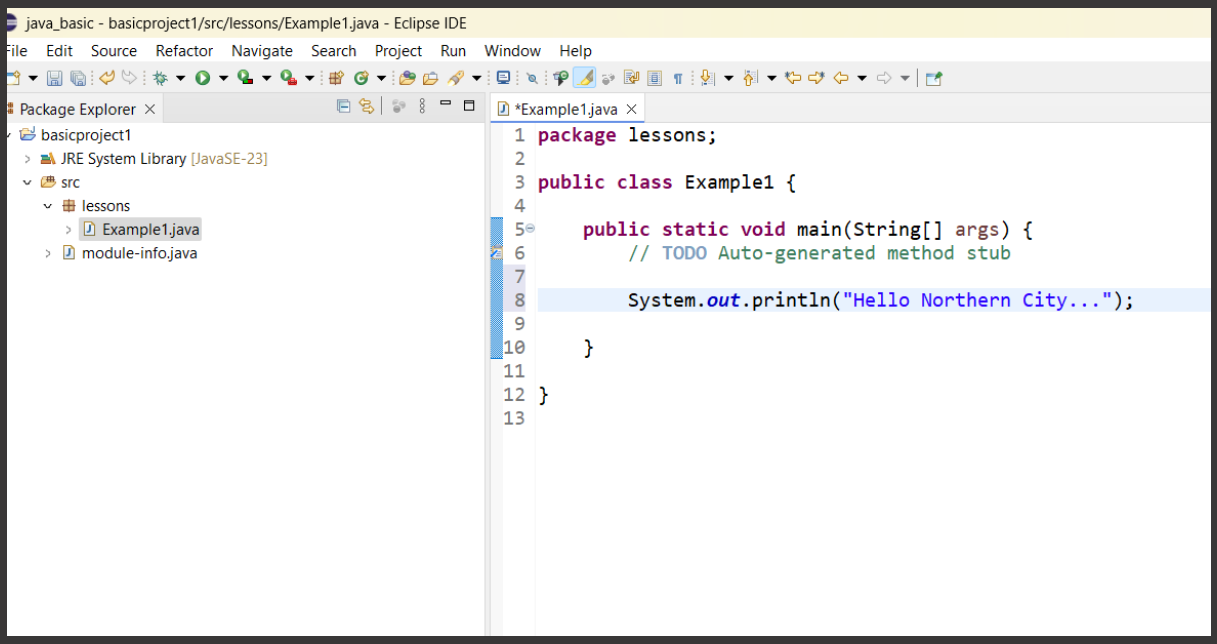
Step 10:

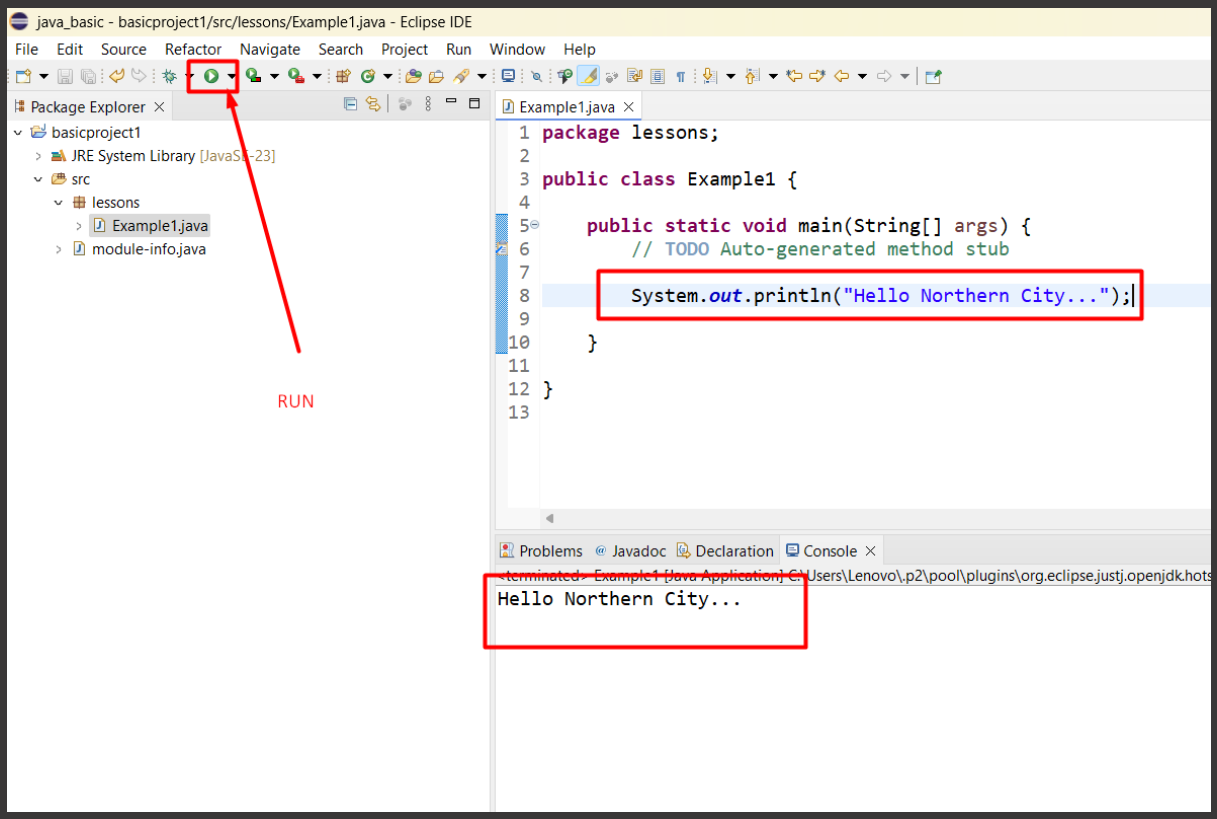
Class name ကိုတော့ capital letter နဲ့ ရေးပေးရပါမယ်။





Step 11:


java_basic - basicproject1/src/lessons/Example1.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
basicproject1
JRE System Library [JavaSE-23]
src
lessons
Example1.java
module-info.java
*Example1.java
1 package lessons;
2
3 public class Example1 {
4
5 public static void main(String[] args) {
6 // TODO Auto-generated method stub
7
8 System.out.println("Hello Northern City...");
9
10 }
11
12 }
13


java_basic - basicproject1/src/lessons/Example1.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
basicproject1
JRE System Library [JavaSE-23]
src
lessons
Example1.java
module-info.java
RUN
Example1.java
1 package lessons;
2
3 public class Example1 {
4
5 public static void main(String[] args) {
6 // TODO Auto-generated method stub
7
8 System.out.println("Hello Northern City...");
9
10 }
11
12 }
13
Problems Javadoc Declaration Console
Hello Northern City...

✍ Short Notes:

1. Project name ကို small letter နဲ့ ရေးရပါမယ်။
2. Package name ကို small letter နဲ့ ရေးရပါမယ်။
3. Class name ကို capital letter နဲ့ ရေးရပါမယ်။

💻 Data Types

Java မှာ Data Types ဆိုတာ **variable တစ်ခုမှာ သိမ်းဆည်းမယ့် data အမျိုးအစားကို သတ်မှတ်ပေးတဲ့အရာ** ဖြစ်ပါတယ်။ Java မှာ Data Types ကို အဓိကအားဖြင့် ၂ မျိုးခွဲထားပါတယ်။

1. Primitive Data Types

ဒီ Data Types တွေက အရမ်းရိုးရှင်းပြီး မူလအတိုင်းပါဝင်တဲ့ အမျိုးအစားတွေဖြစ်ပါတယ်။

Data Type	Size	Default Value	Range	Example
byte	1 byte	0	-128 to 127	byte b = 100;
short	2 bytes	0	-32,768 to 32,767	short s = 5000;
int	4 bytes	0	-2^31 to 2^31-1	int i = 100000;
long	8 bytes	0L	-2^63 to 2^63-1	long l = 15000000000L;
float	4 bytes	0.0f	~6-7 decimal digits	float f = 5.75f;
double	8 bytes	0.0d	~15 decimal digits	double d = 19.99d;
char	2 bytes	'\u0000'	0 to 65,535	char c = 'A';
boolean	1 bit	false	true/false	boolean isJavaFun = true;

ဥပမာ Primitive Types အသုံးပြုပုံ:

Code:
```
int age = 25;
double salary = 500000.50;
char grade = 'A';
boolean isActive = true;
```

2. Non-Primitive Data Types

ဒီ Data Types တွေက Object တွေကို ရည်ညွှန်းပြီး Class တွေကနေ ဖန်တီးထားတာဖြစ်ပါတယ်။

Data Type	Description	Example
String	Character sequence	String name = "John";
Array	Fixed-size data structure	int[] numbers = {1, 2, 3};
Class	User-defined blueprint	Student stu = new Student();
Interface	Abstract type	List<String> list = new ArrayList<>();

ဥပမာ Non-Primitive Types အသုံးပြုပုံ:

Code:
```
String name = "Myanmar";

int[] marks = {80, 90, 85};
ArrayList<String> fruits = new ArrayList<>();
```

Primitive vs Non-Primitive Data Types

Feature	Primitive	Non-Primitive
အမျိုးအစား	Predefined	User-defined
အရွယ်အစား	Fixed	Dynamic
တန်ဖိုး	Stored directly	Store reference
Default Value	Yes	null
Usage	Faster	More features

✍ Short Notes:

1. **Type Casting** - int ကို double အဖြစ်ပြောင်းတာမျိုး

```
Code:

int num = 10;
double dNum = num; // Automatic casting
```

2. **Wrapper Classes** - Primitive တွေကို Object အဖြစ်သုံးဖို့

```
Code:

Integer num = 10; // Autoboxing
int n = num;     // Unboxing
```

3. **String က Special Case** - Non-primitive ဖြစ်ပေမယ့် Java မှာ အထူးအဆင်ပြေအောင်လုပ်ထားတယ်

Java Data Types တွေကို ကောင်းကောင်းနားလည်ထားရင် program တွေရေးတဲ့အခါ memory ကိုထိထိရောက်ရောက်သုံးနိုင်မယ်၊ bug တွေလည်းနည်းမယ်။

💻 Variables

Java မှာ variable ဆိုတာ **data တွေသိမ်းဆည်းဖို့ memory location တစ်ခုကို နာမည်ပေးထားတာ** ဖြစ်ပါတယ်။ Variable တိုင်းမှာ data type, name နဲ့ value တွေပါဝင်ပါတယ်။

Variable ကြေငြာနည်း ၃ မျိုး

1. **Local Variables** - Method တွေအတွင်းမှာသာ အသုံးပြုနိုင်တယ်

Code:

```
public void calculate() {
   int x = 10;  // Local variable
   System.out.println(x);
}
```

2. **Instance Variables** - Class အတွင်းမှာကြေငြာပြီး object တိုင်းအတွက် သီးသန့်ရှိတယ်

Code:

```
class Student {
   String name;  // Instance variable
   int age;      // Instance variable
}
```

3. **Static Variables** - Class level မှာရှိပြီး object အားလုံးအတွက် တူညီတယ်

Code:

```
class School {
   static String schoolName = "ABC School";  // Static variable
}
```

💻 Variable Case Styles

Java မှာ variable name တွေရေးတဲ့အခါ အောက်ပါ style တွေကို အသုံးပြုပါတယ်။

Case Style	ဥပမာ	အသုံးပြုပုံ
camelCase	studentName, accountBalance	Variable/method names
PascalCase	Student, BankAccount	Class/Interface names
snake_case	MAX_SIZE, MIN_VALUE	Constants (final variables)
kebab-case	အသုံးမပြုပါ	Java မှာမသုံးပါ

Variable Naming Rules

1. စာလုံးအသေး/အကြီး၊ $ နဲ့ _ နဲ့စနိုင်တယ် (ဂဏန်းနဲ့မစရ)

   Code:

   ```
   String name;
   int _count;
   double $price;
   ```

2. Java keywords တွေကို variable name အဖြစ်မသုံးရ

   Code:

   ```
   int class;  // Error - 'class' is a keyword
   ```

3. Case sensitive ဖြစ်တယ်

   Code:

   ```
   String firstName;
   String FirstName;  // ဒါကွဲပြားတဲ့ variable
   ```

4. Meaningful names သုံးပါ

   Code:

   ```
   int a;       // မကောင်း
   int age;     // ကောင်း
   ```

```
int userAge;  // ပိုကောင်း
```

Good vs Bad Variable Names

Code:

```
// Bad Examples
int a = 10;
String n = "John";

// Good Examples
int studentCount = 10;
String userName = "JohnDoe";
final double PI_VALUE = 3.14159;  // Constant
```

သိထားရမယ့် အရေးကြီးအချက်များ

✓ Variable name တွေကို camelCase နဲ့ရေးပါ

✓ Constants တွေကို UPPER_SNAKE_CASE နဲ့ရေးပါ

✓ Class name တွေကို PascalCase နဲ့ရေးပါ

✓ Meaningful names တွေသုံးပြီး abbreviation တွေရှောင်ပါ

ဥပမာ - ကောင်းမွန်တဲ့ variable names တွေ:

Code:

```
String firstName = "Kyaw Kyaw";

int maxScore = 100;
boolean isActive = true;
final double TAX_RATE = 0.05;
```

Variable naming convention တွေကို မှန်မှန်ကန်ကန်သုံးမယ်ဆိုရင် code တွေကို ဖတ်ရတာ ပိုလွယ်ကူမယ်၊ maintain လုပ်ရတာ ပိုအဆင်ပြေမယ်။

💻 Basic Calculations

Java Basic Calculations with Scanner Input

Java မှာ အခြေခံသင်္ချာလုပ်ဆောင်ချက်တွေ (ပေါင်း၊ နုတ်၊ မြှောက်၊ စား) လုပ်ဖို့ နမူနာ program တစ်ခုရေးပြပါမယ်။ user ကနေ keyboard ကနေ number ၂ ခုရိုက်ထည့်ပြီး ရလဒ်တွေကို ပြသပေးမယ်။

Addition:

Code:

```
import java.util.Scanner;  // Scanner class ကို import လုပ်တယ်

public class Example1 {
   public static void main(String[] args) {
      // Scanner object ဖန်တီးတယ်
      Scanner input = new Scanner(System.in);

      // User ကနေ number ၂ ခုရိုက်ထည့်ခိုင်းတယ်
      System.out.print("Enter Num1: ");
      int num1 = input.nextInt();

      System.out.print("Enter Num2: ");
      int num2 = input.nextInt();

      // Calculation တွေလုပ်တယ်
      int sum = num1 + num2;        // ပေါင်းခြင်း

      // ရလဒ်တွေကိုပြသမယ်
      System.out.println("\nThe Answer is :");
      System.out.println(num1 + " + " + num2 + " = " + sum);

      input.close();  // Scanner ကိုပိတ်တယ်
   }
}
```

Divide:

Code:

```
import java.util.Scanner;  // Scanner class ကို import လုပ်တယ်

public class Example1 {
   public static void main(String[] args) {
      // Scanner object ဖန်တီးတယ်
      Scanner input = new Scanner(System.in);

      // User ကနေ number ၂ ခုရိုက်ထည့်ခိုင်းတယ်
      System.out.print("Enter Num1: ");
      double num1 = input.nextDouble();

      System.out.print("Enter Num2: ");
      double num2 = input.nextDouble();

      // Calculation တွေလုပ်တယ်
      double quotient = num1 / num2;         // ပေါင်းခြင်း

      // ရလဒ်တွေကိုပြသမယ်
      System.out.println("\nThe answer is :");
      System.out.println(num1 + " / " + num2 + " = " + quotient);


      input.close();  // Scanner ကိုပိတ်တယ်
   }
}
```

🕮 အသေးစိတ်ရှင်းလင်းချက်

1. **Scanner Import လုပ်တယ်**

   Code:

   ```
   import java.util.Scanner;
   ```

   - Keyboard input ယူဖို့ Scanner class လိုတယ်

2. Scanner Object ဖန်တီးတယ်

   Code:

   ```
   Scanner input = new Scanner(System.in);
   ```

3. User Input ယူတယ်

Code:

```
System.out.print("Enter Num1: ");
double num1 = input.nextDouble();
```

   - nextDouble() method က decimal numbers တွေဖတ်ဖို့သုံးတယ်

4. တွက်ချက်မှုတွေလုပ်တယ်

Code:

```
double sum = num1 + num2;
```

5. ရလဒ်တွေပြသမယ်

Code:

```
System.out.println(num1 + " + " + num2 + " = " + sum);
```

6. Scanner ပိတ်တယ်

Code:

```
input.close();
```

💻 ဥပမာ Run ကြည့်ရင်

Result:

```
Enter Num1: 15
Enter Num2: 3

The answer is :
15.0 + 3.0 = 18.0
```

💻 Conditional Statements

Java မှာ Conditional Statements (အခြေအနေစစ်ဆေးချက်များ) ဆိုတာ program ရဲ့လုပ်ဆောင်ချက်တွေကို အခြေအနေအလိုက် ပြောင်းလဲဖို့အတွက် အသုံးပြုပါတယ်။

Conditional Statements အမျိုးအစားများ

1.if Statement

Syntax:
```
if (condition) {
   // condition မှန်ရင် ဒီcodeတွေအလုပ်လုပ်မယ်
}
```

ဥပမာ:

Code:
```
int age = 18;
if (age >= 18) {
   System.out.println("You are eligible to vote now...");
}
```

2. if-else Statement

Syntax:
```
if (condition) {
   // condition မှန်ရင် ဒီcode
} else {
   // condition မှားရင် ဒီcode
}
```

ဥပမာ:

Code:
```
int mark = 40;
if (mark >= 50) {
   System.out.println("Pass...");
} else {
```

```
    System.out.println("Fail...");
}
```

3.if-else if Ladder

Syntax:

```
if (condition1) {
    // code 1
} else if (condition2) {
    // code 2
} else {
    // default code
}
```

ဥပမာ:

Code:

```
int score = 75;
if (score >= 80) {
    System.out.println("Grade A");
} else if (score >= 70) {
    System.out.println("Grade B");
} else if (score >= 60) {
    System.out.println("Grade C");
} else {
    System.out.println("Grade D");
}
```

4. Nested if

Syntax:

```
if (condition1) {
    if (condition2) {
        // code
    }
}
```

ဥပမာ:

Code:

```
int age = 20;
boolean hasID = true;

if (age >= 18) {
    if (hasID) {
        System.out.println("OK... Pass...");
    } else {
        System.out.println("Show Your ID Card");
    }
} else {
```

```
  System.out.println("Sorry…You cannot enter…");
}
```

5. switch Statement

Syntax:

```
switch (expression) {
  case value1:
    // code
    break;
  case value2:
    // code
    break;
  default:
    // default code
}
```

ဥပမာ:

Code:

```
int day = 3;
switch (day) {
  case 1:
    System.out.println("Monday");
    break;
  case 2:
    System.out.println("Tuesday");
    break;
  case 3:
    System.out.println("Wednesday");
    break;
  default:
    System.out.println("Unknown Day…");
}
```

Conditional Operator (Ternary Operator)

Syntax:

```
variable = (condition) ? expressionTrue : expressionFalse;
```

ဥပမာ:

Code:

```
int time = 20;
String result = (time < 18) ? "Day" : "Night";
System.out.println(result);  // "Night"
```

✍ Short Notes:

1. **Comparison Operators** တွေသုံးပါ: ==, !=, >, <, >=, <=
2. **Logical Operators** တွေသုံးပါ: && (and), || (or), ! (not)
3. switch မှာ break မသုံးရင် **fall-through** ဖြစ်တတ်တယ်
4. **Ternary Operator** ကို simple conditions အတွက်သုံးပါ

💻 Practice 1:

Code:

```
import java.util.Scanner;

public class WeatherAdvice {
   public static void main(String[] args) {
      Scanner input = new Scanner(System.in);
      System.out.print("Enter Temperature: ");
      int temp = input.nextInt();

      if (temp > 30) {
         System.out.println("The weather is so Hot.. Drink Water…");
      } else if (temp > 20) {
         System.out.println("The weather is fine…");
      } else if (temp > 10) {
         System.out.println("The weather is cold… Keep yourself warm…");
      } else {
         System.out.println("The weather is very cold…don't go outside…");
      }

      input.close();
   }
}
```

Conditional Statements တွေက Java Programming မှာ အရေးကြီးတဲ့ အခြေခံဖြစ်ပြီး ဆုံးဖြတ်ချက်ချတဲ့ logic တွေရေးတဲ့အခါ အမြဲသုံးရပါမယ်။

💻 Practice 2:

ဒီနမူနာမှာ user ကနေ ဂဏန်း ၃ ခုရိုက်ထည့်ခိုင်းပြီး **if-else** statement သုံးကာ အကြီးဆုံးဂဏန်းကို ရှာပြပါမယ်။

Code:

```
import java.util.Scanner;

public class FindLargestNumber {
   public static void main(String[] args) {
      Scanner input = new Scanner(System.in);

      // ဂဏန်း ၃ ခုရိုက်ထည့်ခိုင်းမယ်
      System.out.print("Enter Num1: ");
      double a = input.nextDouble();

      System.out.print("Enter Num2: ");
      double b = input.nextDouble();

      System.out.print("Enter Num3: ");
      double c = input.nextDouble();

      // အကြီးဆုံးဂဏန်းရှာမယ်
      double largest;

      if (a >b) {
         largest = a;
      }
      else {
         largest = b;
      }

      If(c>largest){
           Largest=c;
       }
      // ရလဒ်ပြမယ်
      System.out.println("\nThe largest no is : " + largest);


      input.close();
   }
}
```

📖 အသေးစိတ်ရှင်းလင်းချက်

1. Scanner သုံးပြီး input ယူတယ်

   Code:

   ```
   Scanner input = new Scanner(System.in);
   double a = input.nextDouble();
   ```

2. if-else condition တွေနဲ့ အကြီးဆုံးကိုစစ်မယ်

Code:

```
if (a >= b && a >= c) {
   largest = a;
}
else if (b >= a && b >= c) {
   largest = b;
}
else {
   largest = c;
}
```

3. ရလဒ်ပြမယ်

Code:

```
System.out.println("\nThe largest no is : " + largest);
```

💻 ဥပမာ Run ကြည့်ရင်

Result:

```
Enter Num1:  25
Enter Num2: 50
Enter Num3: 10

The largest no is : 50.0
```

🏭 Advanced Version

Code:

```
import java.util.InputMismatchException;
import java.util.Scanner;

public class SafeLargestNumberFinder {
   public static void main(String[] args) {
      Scanner input = new Scanner(System.in);

      try {
         System.out.print(“Enter Num1: “);
         double a = input.nextDouble();

         System.out.print(“Enter Num2: “);
         double b = input.nextDouble();

         System.out.print(“Enter Num3: “);
```

```
        double c = input.nextDouble();

        double largest;

        if (a == b && b == c) {
           System.out.println("\nThe inputs no are the same...");
           largest = a;
        }
        else if (a >= b && a >= c) {
           largest = a;
        }
        else if (b >= a && b >= c) {
           largest = b;
        }
        else {
           largest = c;
        }

        System.out.println("\n The largest no is : " + largest);


     } catch (InputMismatchException e) {
        System.out.println("Invalid Inputs...");
     } finally {
        input.close();
     }
  }
}
```

ဒီ Advanced Version မှာ အသစ်ထပ်ထည့်ထားတာတွေ

✓ ဂဏန်းအားလုံးတူနေရင် special message ပြတာ

✓ Input မှားရင် error handling လုပ်ထားတာ (try-catch)

✓ Scanner ကို finally block မှာ ပိတ်ထားတာ

Conditional statements တွေကို ဒီလိုအခြေအနေတွေမှာ အသုံးဝင်ပါတယ်:

- User input validation
- Business logic decisions
- Game development (e.g., scoring systems)
- Algorithm implementations

💻 လေ့ကျင့်ရန် -

Find the largest No

Enter Num1: 2

Enter Num2: 44

Enter Num3: 32

Enter Num4:25

The largest no is : 44

💻 Loops

Java မှာ Loops (repeat control) ဆိုတာ code တစ်ပိုင်းကို အထပ်ထပ် အလုပ်လုပ်စေဖို့ အသုံးပြုပါတယ်။ Loops တွေကို အဓိကအားဖြင့် ၃ မျိုးခွဲထားပါတယ်။

1. for Loop

Syntax:
```
for (start; stop; increment/decrement) {
   // code block statement
}
```

ဥပမာ:

Code:
```
for (int I = 1; I <= 5; i++) {
   System.out.println("Hello " + i);
}
```

ရလဒ်:

Output:
```
Hello 1
Hello 2
Hello 3
Hello 4
Hello 5
```

2. while Loop

Syntax:
```
while (condition) {
   // condition မှန်နေသမျှ ဆက်လုပ်မယ်
}
```

ဥပမာ:

Code:
```
int count = 1;
while (count <= 3) {
   System.out.println("Count: " + count);
   count++;
}
```

ရလဒ်:

Output:
```
Count: 1
Count: 2
Count: 3
```

3. do-while Loop

Syntax:
```
do {
   // အနည်းဆုံး ၁ ခါတော့ အလုပ်လုပ်မယ်
} while (condition);
```

ဥပမာ:

Code:
```
int x = 5;
do {
   System.out.println("Value: " + x);
   x--;
} while (x > 0);
```

ရလဒ်:

Copy
```
Value: 5
Value: 4
Value: 3
Value: 2
Value: 1
```

Loops တွေရဲ့ အထူးအချက်များ

break Statement

```
Code:
for (int I = 1; I <= 10; i++) {
   if (I == 5) {
      break;  // loop ကနေ ထွက်သွားမယ်
   }
   System.out.println(i);
}
```

continue Statement

```
Code:
for (int I = 1; I <= 5; i++) {
   if (I == 3) {
      continue;  // ဒီအကြိမ်ကိုကျော်မယ်
   }
   System.out.println(i);
}
```

Nested Loops

```
Code:
for (int I = 1; I <= 3; i++) {
   for (int j = 1; j <= 2; j++) {
      System.out.println("i=" + I + ", j=" + j);
   }
}
```

ဘယ် Loop ကို ဘယ်အချိန်မှာ သုံးမလဲ?

Loop Type	အသုံးပြုရမယ့်အခြေအနေ
for	လုပ်ရမယ့်အကြိမ်ရေသိရင်

Loop Type	အသုံးပြုရမယ့်အခြေအနေ
while	လုပ်ရမယ့်အကြိမ်ရေမသိရင်
do-while	အနည်းဆုံး ၁ ခါလုပ်ဖို့လိုရင်

Practice 1:

Code:

```
import java.util.Scanner;

public class MultiplicationTable {
   public static void main(String[] args) {
      Scanner input = new Scanner(System.in);
      System.out.print("Enter a Number: ");
      int num = input.nextInt();

      System.out.println(num + " table list ... ");
      for (int I = 1; I <= 10; i++) {
         System.out.println(num + " x " + I + " = " + (num*i));
      }

      input.close();
   }
}
```

✍ **Short Notes:**

- Loops တွေကို array/list တွေနဲ့ တွဲသုံးလေ့ရှိတယ်
- Infinite loop (အဆုံးမရှိတဲ့ loop) ဖြစ်မသွားအောင် ဂရုစိုက်ပါ
- break နဲ့ continue တွေကို လိုအပ်မှသာသုံးပါ

Loops တွေက Java Programming မှာ အရေးပါတဲ့ အခြေခံ concept တစ်ခုဖြစ်ပြီး data တွေကို အထပ်ထပ်လုပ်ဆောင်ဖို့လိုတဲ့အခါတိုင်း အသုံးပြုရပါမယ်။

လေ့ကျင့်ရန်-

Practice 1:

i=10

j=20

```
####################
#                  #
#                  #
#                  #
####################
#   #          #   #
#   #          #   #
#   #          #   #
#   #          #   #
####################
```

Practice 2:

i=11

j=11

```
###########
#     #     #
#    ###    #
#   #####   #
#  ####### #
###########
# ####### #
#   #####   #
#    ###    #
#     #     #
###########
```

💻 String Manipulations

Java မှာ **String** တွေကို လွယ်လွယ်ကူကူ **အသုံးပြုနည်း**နဲ့ **လုပ်ဆောင်ချက်**တွေကို ရှင်းပြပေးမယ်။

1. String ဖန်တီးနည်း

Code:
```
String str1 = "Hello";        // String Literal (Memory-efficient)
String str2 = new String("Java"); // new Keyword (Heap Memory)
```

2. String Methods အသုံးပြုနည်း

(a) Length & Case ပြောင်းခြင်း

Code:
```
String text = "Java Programming";
System.out.println(text.length()); // 16 (စာလုံးရေ)

System.out.println(text.toUpperCase()); // "JAVA PROGRAMMING"
System.out.println(text.toLowerCase()); // "java programming"
```

(b) String ပေါင်းခြင်း (Concatenation)

Code:
```
String s1 = “Hello”;
String s2 = “Java”;
System.out.println(s1 + “ “ + s2); // “Hello Java”
System.out.println(s1.concat(s2)); // “HelloJava”
```

(c) String နှိုင်းယှဉ်ခြင်း (Comparison)

Code:
```
String a = “Java”;
String b = “java”;
System.out.println(a.equals(b)); // false (Case-sensitive)
System.out.println(a.equalsIgnoreCase(b)); // true
System.out.println(a.compareTo(b)); // -32 (ASCII diff)
```

(d) Substring & Character ဖြတ်ယူခြင်း

Code:
```
String str = "Programming";
System.out.println(str.substring(3)); // "gramming"
System.out.println(str.substring(3, 7)); // "gram"
System.out.println(str.charAt(4)); // 'r' (Index 4)
```

(e) String Search & Replace

Code:
```
String msg = "I love Java";
System.out.println(msg.contains("Java")); // true
System.out.println(msg.indexOf("love")); // 2
System.out.println(msg.replace("Java", "Python")); // "I love Python"
```

(f) String Split & Join

Code:
```
String data = “Apple,Banana,Mango”;
String[] fruits = data.split(“,”); // [“Apple”, “Banana”, “Mango”]
String joined = String.join(“-“, fruits); // “Apple-Banana-Mango”
```

3. StringBuilder vs StringBuffer

- **StringBuilder** → မြန်ဆန် (Thread-safe မဟုတ်)
- **StringBuffer** → **Thread-safe** (နှေးပေမယ့် Multi-thread အဆင်ပြေ)

Code:
```
StringBuilder sb = new StringBuilder("Hello");
sb.append(" Java");
System.out.println(sb.toString()); // "Hello Java"
```

4. Escape Characters

Code:
```
System.out.println("He said, \"Hello!\""); // "He said, "Hello!""
System.out.println("Line1\nLine2"); // New Line
System.out.println("Tab\tSpace"); // Tab
```

5. String Formatting

Code:
```
String name = "Alice";
int age = 25;
System.out.printf("Name: %s, Age: %d", name, age);
System.out.println(String.format("Name: %s", name));
```

အရေးကြီးသော Note

- Java **String** သည် **Immutable** (ပြောင်းလဲ၍မရ)
  ဖြစ်တာကြောင့် **StringBuilder/StringBuffer** ကို မကြာခဏ ပြုပြင်ရန် လိုအပ်ပါက သုံးသင့်ပါတယ်။

Java String Manipulation သည် **Project တိုင်းအတွက် အရေးကြီးပြီး** ဒီ Methods တွေကို ကျွမ်းကျင်စွာ အသုံးပြုနိုင်ရင် **Coding Efficiency** တက်မှာပါ!

💻 Static Array

Static Array ဆိုတာ **Fixed Size** ရှိတဲ့ Array တစ်မျိုးဖြစ်ပြီး Java မှာ အောက်ပါအတိုင်း အသုံးပြုနိုင်ပါတယ်။

1. Static Array ကို ဘယ်လိုသတ်မှတ်မလဲ?

(က) Declaration & Initialization

Code:
```
// နည်းလမ်း (၁) - Size သတ်မှတ်ပြီး နောက်မှတန်ဖိုးထည့်ခြင်း
int[] numbers = new int[5];
numbers[0] = 10;
numbers[1] = 20;

// နည်းလမ်း (၂) - တန်ဖိုးတွေကို တစ်ခါတည်းထည့်ခြင်း
String[] names = {"Alice", "Bob", "Charlie"};
```

(ခ) Array Length သိရှိခြင်း

Code:
```
System.out.println(names.length); // 3 (အရေအတွက်)
```

2. Static Array ရဲ့ အရေးကြီးသော Features

(က) Fixed Size

- **တစ်ခါသတ်မှတ်ပြီးရင် Size ပြောင်း၍မရ**
- ArrayIndexOutOfBoundsException ဖြစ်နိုင်တယ် (အကယ်၍ Invalid Index သုံးမိရင်)

(ခ) Fast Access (O(1) Time Complexity)

Index သုံးပြီး တန်ဖိုးတွေကို ချက်ချင်းရှာနိုင်တယ်

Code:
```
System.out.println(names[1]); // "Bob"
```

(ဂ) Primitive & Reference Types နှစ်မျိုးလုံးအတွက် အသုံးပြုနိုင်တယ်

Code:
```
int[] ages = {20, 25, 30}; // Primitive (int)
String[] cities = {"Yangon", "Mandalay"}; // Reference (String)
```

3. Static Array ကို Loop ပတ်ခြင်း

(က) For Loop သုံးခြင်း

Code:
```
for (int i = 0; i < numbers.length; i++) {
    System.out.println(numbers[i]);
}
```

(ခ) Enhanced For Loop (For-Each)

Code:
```
for (String name : names) {
    System.out.println(name);
}
```

4. Static Array vs Dynamic Array (ArrayList)

Feature	Static Array	ArrayList (Dynamic)
Size	Fixed (ပြောင်းလဲ၍မရ)	Resizable (အလိုအလျောက်ကြီးထွားနိုင်)
Performance	ပိုမြန်တယ် (Memory-efficient)	နည်းနည်းနှေးတယ် (ဒါပေမယ့် Flexible)
Methods	length, Index-based Access	add(), remove(), get(), size()

5. Static Array သုံးသင့်တဲ့အချိန်

- **အရွယ်အစားသေချာသိပြီး** မပြောင်းလဲတဲ့အခါမျိုး (ဥပမာ - Days of the Week)
- **အမြန်ဆုံး Access** လိုအပ်တဲ့အခါမျိုး

ဥပမာ - Days of the Week

Code
```
String[] days = {"Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"};
System.out.println(days[3]); // "Thu"
```

အရေးကြီးသော Note

- Java မှာ **Array** က **Object** ဖြစ်တာကြောင့် new keyword သုံးပြီး Memory မှာ နေရာယူရတယ်။
- **Static Array** ကို **အရွယ်အစားသေချာသိရင်** သုံးပါ၊ **မသိရင် ArrayList သုံးပါ။**

💻 Methods

Java မှာ Methods (မှတ်သားလုပ်ဆောင်ချက်များ) ဆိုတာ **code တွေကို အပိုင်းလိုက်ခွဲပြီး ပြန်လည်အသုံးပြုနိုင်အောင်** ဖန်တီးထားတဲ့ block တစ်ခုဖြစ်ပါတယ်။

Method ရဲ့ အခြေခံဖွဲ့စည်းပုံ

Syntax:
```
accessModifier returnType methodName(parameters) {
   // method body
   return value;  // return type ရှိရင်
}
```

Method အမျိုးအစားများ

1. Parameter မပါတဲ့ Method

Code:
```
public void greet() {
   System.out.println("မင်္ဂလာပါ!");
}

// ခေါ်သုံးပုံ
greet();  // Output: မင်္ဂလာပါ!
```

2. Parameter ပါတဲ့ Method

Code:
```
public void add(int a, int b) {
   System.out.println("ပေါင်းလဒ်: " + (a+b));
}

// ခေါ်သုံးပုံ
add(5, 3);  // Output: ပေါင်းလဒ်: 8
```

3. Return Value ပါတဲ့ Method

Code:
```
public int square(int num) {
   return num * num;
}

// ခေါ်သုံးပုံ
int result = square(4);
System.out.println(result);  // Output: 16
```

4. Static Method

Code:
```
public class Calculator {
   public static int multiply(int x, int y) {
      return x * y;
   }
}

// ခေါ်သုံးပုံ (object မလိုဘူး)
int ans = Calculator.multiply(5, 6);
System.out.println(ans);  // Output: 30
```

Method Overloading

တူညီတဲ့ method name နဲ့ parameter အမျိုးမျိုးနဲ့ method တွေဖန်တီးနိုင်တယ်။

Code:
```
public class Display {
   public void show(int num) {
      System.out.println("ဂဏန်း: " + num);
   }

   public void show(String text) {
      System.out.println("စာသား: " + text);
   }

   public void show(double num) {
      System.out.println("ဒဿမဂဏန်း: " + num);
   }
}

// ခေါ်သုံးပုံ
Display d = new Display();
d.show(10);      // ဂဏန်း: 10
d.show("Hello");  // စာသား: Hello
```

```
d.show(3.14);    // ဒဿမဂဏန်း: 3.14
```

Recursive Method (ပြန်ခေါ်သုံးနိုင်သော Method)

Code:

```
public int factorial(int n) {
   if (n == 1) return 1;
   return n * factorial(n-1);
}

// ခေါ်သုံးပုံ
System.out.println(factorial(5));  // Output: 120 (5!)
```

Method သုံးရတဲ့ အကျိုးကျေးဇူးများ

1. **Code Reusability** - တစ်ခါရေးပြီး အကြိမ်ကြိမ်သုံးနိုင်တယ်
2. **Modularity** – Code တွေကို အပိုင်းလိုက်ခွဲနိုင်တယ်
3. **Maintainability** - ပြင်ဆင်ရတာ ပိုလွယ်တယ်
4. **Abstraction** – Complex logic တွေကို ဖုံးကွယ်နိုင်တယ်

Practice 1:

Code:

```
import java.util.Scanner;

public class MethodExample {
   // BMI တွက်တဲ့ method
   public static double calculateBMI(double weight, double height) {
      return weight / (height * height);
   }

   public static void main(String[] args) {
      Scanner input = new Scanner(System.in);

      System.out.print("Enter Weight (kg): ");
      double w = input.nextDouble();

      System.out.print("Enter Height (m): ");
      double h = input.nextDouble();

      double bmi = calculateBMI(w, h);
      System.out.printf("Your BMI: %.2f ", bmi);

      input.close();
```

```
  }
}
```

✍ **Short Notes:**

- void return type ဆိုရင် return statement မလိုဘူး
- Method name တွေကို camelCase နဲ့ရေးပါ (ဥပမာ - calculateTotal)
- Parameter တွေကို comma ခံပြီးသတ်မှတ်ပါ
- Java မှာ Pass by Value သာရှိတယ် (Pass by Reference မရှိဘူး)

Methods တွေက Java Programming မှာ အရေးကြီးတဲ့ building blocks တွေဖြစ်ပြီး program တွေကို ပိုမိုစနစ်ကျအောင် ရေးသားနိုင်ဖို့ ကူညီပေးပါတယ်။

လေ့ကျင့်ရန် -

Practice 1:

MENU

1.Add

2.Subtract

---

3. Exit

💻 OOP

Java မှာ OOP ဆိုတာ **Object-Oriented Programming** ပါ။ ဒါက real-world entities တွေကို **objects** အဖြစ် model လုပ်ပြီး program တွေရေးသားတဲ့နည်းလမ်းတစ်ခုပါ။ OOP က program တွေကို ပိုပြီး organized, reusable နဲ့ maintainable ဖြစ်အောင် ကူညီပေးပါတယ်။ Java OOP က **Encapsulation, Inheritance, Polymorphism, Abstraction** ဆိုတဲ့ အခြေခံ concepts (4 pillars) တွေပေါ်မှာ အခြေခံထားပါတယ်။ Real-world problems တွေကို objects အဖြစ် model လုပ်ရာမှာ အများကြီး အထောက်အကူပြုပါတယ်။ အသေးစိတ်ကို ဆက်လက်ပြီး လေ့လာကြည့်ရအောင်။

💻 Constructor

Java မှာ Constructor ဆိုတာက **Object တစ်ခုကို အသစ်ဖန်တီးတဲ့အခါ အလိုအလျောက်ခေါ်တဲ့ special method** တစ်ခုပါ။ Constructor က object တွေရဲ့ **ကနဦးအခြေအနေ (initial state)** ကို သတ်မှတ်ဖို့အတွက် အသုံးပြုပါတယ်။

Constructor ရဲ့ အဓိက အချက်များ

1. **Class နာမည်နဲ့ တူရမယ်**
2. **Return type မထည့်ရဘူး** (void, int စတာတွေ မပါဘူး)
3. **new keyword သုံးပြီး object ဖန်တီးတဲ့အခါ အလိုအလျောက်အလုပ်လုပ်တယ်**
4. **Overloading လုပ်လို့ရတယ်** (Parameter အမျိုးမျိုးနဲ့ Constructor အများကြီးသတ်မှတ်နိုင်တယ်)

Constructor အမျိုးအစားများ

1. Default Constructor (Parameter မပါတဲ့ Constructor)

- Java Compiler က **အလိုအလျောက်** ဖန်တီးပေးတယ် (ဒါမှမဟုတ် ကိုယ်တိုင်လည်းရေးလို့ရတယ်)

- Instance variables တွေကို **default values** (int=0, String=null) တွေနဲ့ စပါတယ်

ဥပမာ:

Code:
```
class Phone {
   String brand;

   // Default Constructor
   Phone() {
      brand = "Unknown";
   }
}

public class Main {
   public static void main(String[] args) {
      Phone myPhone = new Phone(); // Constructor ကိုခေါ်တယ်

      System.out.println(myPhone.brand); // Output: Unknown

   }
}
```

2. Parameterized Constructor (Parameter ပါတဲ့ Constructor)

- Object ဖန်တီးတဲ့အခါ **တန်ဖိုးတွေပို့ပြီး အစပြုလုပ်နိုင်တယ်**

ဥပမာ:

Code:
```
class Student {
   String name;
   int age;

   // Parameterized Constructor
   Student(String n, int a) {
      name = n;
      age = a;
   }
}

public class Main {
   public static void main(String[] args) {
      Student stu1 = new Student("Su Myat", 18);

      System.out.println(stu1.name + " age is : " + stu1.age);

      //Output: Su Myat age is: 18

   }
}
```

3. Constructor Overloading (Constructor အများကြီး)

- **တူညီတဲ့ Class** ထဲမှာ **Parameter အမျိုးမျိုး** နဲ့ Constructor တွေအများကြီးသတ်မှတ်နိုင်တယ်

ဥပမာ:

Code:
```
class Laptop {
   String brand;
   double price;

   // Constructor 1: No-args
   Laptop() {
      brand = "Default";
      price = 0.0;
   }

   // Constructor 2: Brand only
   Laptop(String b) {
      brand = b;
      price = 0.0;
   }

   // Constructor 3: Full details
   Laptop(String b, double p) {
      brand = b;
      price = p;
   }
}

public class Main {
   public static void main(String[] args) {
      Laptop lap1 = new Laptop(); // Constructor 1
      Laptop lap2 = new Laptop("Lenovo"); // Constructor 2
      Laptop lap3 = new Laptop("MacBook", 1500.0); // Constructor 3
   }
}
```

this Keyword သုံးနည်း

- **Instance variable** နဲ့ **Parameter** နာမည်တူနေရင် this သုံးပြီးခွဲခြားနိုင်တယ်

ဥပမာ:

Code:
```
class Person {
```

```
  String name;

  Person(String name) {
    this.name = name; // this.name = instance variable, name = parameter
  }
}
```

super(): Parent Class Constructor ခေါ်နည်း

- **Inheritance** မှာ Child Class Constructor က Parent Class Constructor ကို super() နဲ့ခေါ်နိုင်တယ်
- **super() က Constructor ရဲ့ ပထမဆုံး statement ဖြစ်ရမယ်**

ဥပမာ:

Code:

```
class Vehicle {
  Vehicle() {
    System.out.println("Vehicle Constructor");
  }
}

class Car extends Vehicle {
  Car() {
    super(); // Vehicle Constructor ကိုခေါ်တယ်
    System.out.println("Car Constructor");
  }
}

public class Main {
  public static void main(String[] args) {
    Car myCar = new Car();
    /* ရလဒ်:
      Vehicle Constructor
      Car Constructor
    */
  }
}
```

✍ Short Notes:

✓ Constructor က Object ဖန်တီးတိုင်း အလိုအလျောက်အလုပ်လုပ်တယ်

✓ Constructor ကို ပုံမှန် method တွေလို ခေါ်လို့မရဘူး (new နဲ့ပဲခေါ်ရတယ်)

✓ Constructor ကို private လုပ်ပြီး Singleton Pattern တွေမှာအသုံးပြုနိုင်တယ်

💻 Encapsulation

Encapsulation (Data Hiding) ဆိုတာ Java OOP ရဲ့ အရေးကြီးတဲ့ concept တစ်ခုဖြစ်ပြီး **data နဲ့ methods တွေကို တစ်စုတစ်စည်းတည်း ထုပ်ပိုးကာ အပြင်ကနေ တိုက်ရိုက်ဝင်ရောက်မှုကို ကန့်သတ်တဲ့နည်းလမ်း** ဖြစ်ပါတယ်။

Encapsulation ရဲ့ အဓိက အချက်များ

1. **Data hiding** - Class အတွင်းရှိ data တွေကို private လုပ်ထားခြင်း
2. **Controlled access** - Getter/Setter methods တွေကနေ တဆင့်သာ data တွေကို access လုပ်ခြင်း
3. **Increased security** - Data တွေကို မလိုလားအပ်တဲ့ ပြောင်းလဲမှုတွေကနေ ကာကွယ်ပေးခြင်း

Encapsulation ကို ဘယ်လို Implement လုပ်မလဲ?

1.Variables တွေကို private လုပ်ပါ

Code:
```
public class Student {
   private String name;  // private variable
   private int age;      // private variable
}
```

2. Getter/Setter Methods တွေ ဖန်တီးပါ

Code:
```
public class Student {
   private String name;
   private int age;

   // Getter for name
   public String getName() {
      return name;
   }

   // Setter for name
   public void setName(String name) {
```

```
        this.name = name;
    }

    // Getter for age
    public int getAge() {
        return age;
    }

    // Setter for age with validation
    public void setAge(int age) {
        if(age > 0) {  // Age validation
            this.age = age;
        } else {
            System.out.println("အသက်က အနုတ်တန်ဖိုး မဖြစ်ရဘူး");
        }
    }
}
```

3.Main Class မှာ အသုံးပြုပုံ

Code:

```
public class Main {
    public static void main(String[] args) {
        Student student = new Student();

        // Set values using setters
        student.setName("မောင်မောင်");
        student.setAge(20);

        // Get values using getters
        System.out.println("အမည်: " + student.getName());
        System.out.println("အသက်: " + student.getAge());

        // Invalid age example
        student.setAge(-5);  // Error message will be shown
    }
}
```

Encapsulation ရဲ့ အကျိုးကျေးဇူးများ

1. **Data Security** - Data တွေကို unauthorized access ကနေ ကာကွယ်ပေးတယ်
2. **Flexibility** - Internal implementation ကို ပြောင်းလဲရင် အပြင်က code တွေကို မထိခိုက်စေဘူး
3. **Validation** - Data တွေထည့်တဲ့အခါ validation rules တွေ ထည့်ပေးလို့ရတယ်
4. **Easy Maintenance** - Code တွေကို ပြန်ပြင်ရတာ ပိုလွယ်ကူတယ်

တကယ့် Project တွေမှာ Encapsulation အသုံးပြုပုံ

Code:

```
public class BankAccount {
   private String accountNumber;
   private double balance;

   public BankAccount(String accountNumber) {
      this.accountNumber = accountNumber;
      this.balance = 0.0;
   }

   public void deposit(double amount) {
      if(amount > 0) {
         balance += amount;
         System.out.println(amount + " Deposited and Update Balance now is : " + balance);
      } else {
         System.out.println("Invalid Amount...");
      }
   }

   public void withdraw(double amount) {
      if(amount > 0 && amount <= balance) {
         balance -= amount;
         System.out.println(amount + " is deducted and remaining balance is: " + balance);
      } else {
         System.out.println("Invalid amount or insufficient amount...");
      }
   }

   public double getBalance() {
      return balance;
   }
}
```

✍ Short Notes:

✓ Encapsulation ကို Java Beans တွေမှာ အများဆုံးအသုံးပြုတယ်

✓ Android Development မှာ Model Classes တွေရေးတဲ့အခါ အရေးကြီးတယ်

✓ Spring Framework တွေမှာ DTO (Data Transfer Objects) တွေရေးတဲ့အခါ အသုံးဝင်တယ်

Encapsulation က Java Programming မှာ အရေးကြီးတဲ့ concept တစ်ခုဖြစ်ပြီး professional level programming တွေအတွက် မရှိမဖြစ်လိုအပ်ပါတယ်။

💻 Inheritance

Inheritance (အမွေဆက်ခံခြင်း) ဆိုတာ Java OOP ရဲ့ အရေးကြီးတဲ့ concept တစ်ခုဖြစ်ပြီး **တစ်ခုထက်ပိုတဲ့ class တွေကြားမှာ parent-child relationship ဖန်တီးပေးတဲ့နည်းလမ်း** ဖြစ်ပါတယ်။

Inheritance ရဲ့ အဓိက အချက်များ

1. **Code Reusability** - Parent class က code တွေကို child class က ပြန်သုံးလို့ရတယ်
2. **Method Overriding** - Child class က parent class ရဲ့ method တွေကို modify လုပ်လို့ရတယ်
3. **extends keyword** - Inheritance လုပ်ဖို့အတွက် သုံးတယ်
4. **IS-A Relationship** - "Child IS-A Parent" ဆိုတဲ့ ဆက်နွယ်မှုကို ဖော်ပြတယ်

Inheritance ကို ဘယ်လို Implement လုပ်မလဲ?

1. Base Class (Parent Class) ဖန်တီးပါ

Code:
```
class Animal {
   void eat() {
      System.out.println("It is eating...");
   }

   void sleep() {
      System.out.println("It is sleeping...");
   }
}
```

2.Derived Class (Child Class) ဖန်တီးပြီး extends လုပ်ပါ

Code:
```
class Dog extends Animal {  // Dog က Animal ကို inherit လုပ်တယ်
   void bark() {
      System.out.println(“It is barking now...”);
   }
}
```

3.Main Class မှာ အသုံးပြုပုံ

Code:
```
public class Main {
  public static void main(String[] args) {
    Dog myDog = new Dog();

    // Parent class ကနေ inherit လုပ်ထားတဲ့ methods တွေ
    myDog.eat();   // Output: It is eating...
    myDog.sleep(); // Output: It is sleeping...

    // Dog class က method
    myDog.bark();  // Output:It is barking now...
  }
}
```

Inheritance အမျိုးအစားများ

1.Single Inheritance

Code:
```
class A { }
class B extends A { } // B က A ကိုပဲ inherit လုပ်တယ်
```

1. Multilevel Inheritance

Code:
```
class A { }
class B extends A { }
class C extends B { } // C → B → A
```

2. Hierarchical Inheritance

Code:
```
class A { }
class B extends A { }
class C extends A { } // B နဲ့ C က A ကိုပဲ inherit လုပ်တယ်
```

Method Overriding လုပ်နည်း

Code:

```
class Animal {
   void makeSound() {
      System.out.println("Animal Sound... ");
   }
}

class Cat extends Animal {
   @Override
   void makeSound() {  // Parent class က method ကို override လုပ်တယ်
      System.out.println("Cat Sound... Meow!");
   }
}

public class Main {
   public static void main(String[] args) {
      Animal myCat = new Cat();
      myCat.makeSound();  // Output: Cat Sound... Meow!
   }
}
```

super keyword အသုံးပြုနည်း

Code:

```
class Vehicle {
   int speed = 50;
}

class Bike extends Vehicle {
   int speed = 100;

   void display() {
      System.out.println("Bike speed: " + speed);         // 100
      System.out.println("Vehicle speed: " + super.speed); // 50
   }
}
```

Inheritance ရဲ့ အကျိုးကျေးဇူးများ

1. **Code Reuse** - ထပ်ခါထပ်ခါရေးစရာမလိုတော့ဘူး
2. **Method Overriding** – Polymorphism အတွက်အရေးကြီးတယ်
3. **Code Organization** – Class တွေကို hierarchy အလိုက်စီနိုင်တယ်

✍ Short Notes:

✓ Java မှာ **Multiple Inheritance** ကို class တွေနဲ့မလုပ်နိုင်ဘူး (interface တွေနဲ့ပဲလုပ်နိုင်တယ်)

✓ Constructor တွေကို inherit မလုပ်နိုင်ဘူး

✓ Private members တွေကို inherit မလုပ်နိုင်ဘူး

Inheritance က Java Programming မှာ အရေးကြီးတဲ့ concept တစ်ခုဖြစ်ပြီး real-world relationships တွေကို program ထဲမှာ model လုပ်တဲ့အခါ အသုံးဝင်ပါတယ်။

💻 Polymorphism

Polymorphism (တစ်ခုတည်းကို ပုံစံမျိုးစုံဖြင့် အသုံးပြုနိုင်ခြင်း) ဆိုတာ Java OOP ရဲ့ အရေးကြီးဆုံး concept တစ်ခုဖြစ်ပြီး **"တစ်ခုတည်းသော interface ကို အမျိုးမျိုးသော implementation တွေနဲ့ အသုံးပြုနိုင်တဲ့စွမ်းရည်"** လို့ အဓိပ္ပါယ်ဖွင့်ဆိုနိုင်ပါတယ်။

Polymorphism ရဲ့ အဓိက အချက်များ

1. **Method Overloading (Compile-time Polymorphism)**
2. **Method Overriding (Runtime Polymorphism)**
3. **Interface နဲ့ Abstract Class တွေကို အသုံးပြုခြင်း**

1. Method Overloading (Compile-time Polymorphism)

Code:

```
class Calculator {
   // တူညီတဲ့ method name နဲ့ parameters မတူတဲ့ methods တွေ
   int add(int a, int b) {
      return a + b;
   }

   double add(double a, double b) {
      return a + b;
   }

   String add(String a, String b) {
      return a + b;
   }
}

public class Main {
   public static void main(String[] args) {
      Calculator calc = new Calculator();
      System.out.println(calc.add(5, 10));      // 15
      System.out.println(calc.add(5.5, 10.5));   // 16.0
      System.out.println(calc.add("Hello", "World")); // HelloWorld
   }
}
```

2.Method Overriding (Runtime Polymorphism)

Code:

```
class Animal {
   void makeSound() {
      System.out.println("Animal Sound...");
```

```
    }
}

class Cat extends Animal {
    @Override
    void makeSound() {
        System.out.println("Cat Sound… Meow … ");

    }
}

class Dog extends Animal {
    @Override
    void makeSound() {
        System.out.println("Dog Barking … Wok Wok…");

    }
}

public class Main {
    public static void main(String[] args) {
        Animal myAnimal = new Animal();
        Animal myCat = new Cat();
        Animal myDog = new Dog();

        myAnimal.makeSound();  // Animal Sound …
        myCat.makeSound();     // Cat Sound … Meow …
        myDog.makeSound();     // Dog Barking … Wok Wok…
    }
}
```

3. Interface နဲ့ Polymorphism

Code:

```
interface Shape {
    void draw();
}

class Circle implements Shape {
    @Override
    public void draw() {
        System.out.println("Drawing Circle Now…");

    }
}

class Square implements Shape {
    @Override
    public void draw() {
        System.out.println("Drawing Rectangle Now…");

    }
}

public class Main {
    public static void main(String[] args) {
        Shape myShape = new Circle();
        myShape.draw();  // Drawing Circle Now…
```

```
        myShape = new Square();
        myShape.draw();  // Drawing Rectangle Now...
    }
}
```

Polymorphism ရဲ့ အကျိုးကျေးဇူးများ

1. **Code Flexibility** - တစ်ခုတည်းသော interface ကို အမျိုးမျိုးသုံးနိုင်တယ်
2. **Code Maintainability** – Code တွေကို ပြင်ဆင်ရတာ ပိုလွယ်ကူတယ်
3. **Extensibility** - အသစ်ထပ်ထည့်ရတာ လွယ်ကူတယ်

✍ Short Notes:

✓ Polymorphism ကို Java Collections Framework တွေမှာ အများဆုံးတွေ့ရတယ်

✓ Android Development မှာ Adapter Pattern တွေမှာ အသုံးများတယ်

✓ Spring Framework မှာ Dependency Injection လုပ်တဲ့အခါ အရေးပါတယ်

Polymorphism က Java Programming မှာ အရေးကြီးတဲ့ concept တစ်ခုဖြစ်ပြီး professional level programming တွေအတွက် မရှိမဖြစ်လိုအပ်ပါတယ်။

💻 Abstraction

Abstraction ဆိုတာ Java OOP ရဲ့ အရေးကြီးတဲ့ concept တစ်ခုဖြစ်ပြီး "**လိုအပ်တဲ့အချက်တွေကိုပဲပြပြီး အတွင်းပိုင်း အသေးစိတ်တွေကို ဖုံးကွယ်ထားတဲ့ နည်းလမ်း**" ဖြစ်ပါတယ်။ Abstraction ကို **abstract class** နဲ့ **interface** တွေသုံးပြီး implement လုပ်ပါတယ်။

Abstraction ရဲ့ အဓိက အချက်များ

1. **Essential features တွေကိုပဲပြတယ်**
2. **Implementation details တွေကို ဖုံးကွယ်ထားတယ်**
3. **abstract keyword သုံးပြီး ဖန်တီးတယ်**
4. **100% abstraction အတွက် interface တွေသုံးတယ်**

1. Abstract Class အသုံးပြုနည်း

Code:

```
abstract class Vehicle {
   // Abstract method (implementation မပါ)
   abstract void start();

   // Concrete method (implementation ပါ)
   void stop() {
      System.out.println("stop the vehicle now...");
   }
}

class Car extends Vehicle {
   @Override
   void start() {
      System.out.println("the car starts moving now...");
   }
}

class Bike extends Vehicle {
   @Override
   void start() {
      System.out.println("the bike starts moving now...");
   }
}
```

```
public class Main {
   public static void main(String[] args) {
      Vehicle myCar = new Car();
      Vehicle myBike = new Bike();

      myCar.start();  // the car starts moving now…
      myCar.stop();   // stop the vehicle now…

      myBike.start(); // the bike starts moving now…
      myBike.stop();  // stop the vehicle now…
   }
}
```

2. Interface အသုံးပြုနည်း (100% Abstraction)

Code:

```
interface Bank {
   // Abstract methods တွေပဲပါ
   void deposit(double amount);
   void withdraw(double amount);
   double getBalance();
}

class KBZBank implements Bank {
   private double balance;

   @Override
   public void deposit(double amount) {
      balance += amount;
   }

   @Override
   public void withdraw(double amount) {
      if(balance >= amount) {
         balance -= amount;
      }
   }

   @Override
   public double getBalance() {
      return balance;
   }
}

public class Main {
   public static void main(String[] args) {
      Bank bank = new KBZBank();
      bank.deposit(500000);
      bank.withdraw(200000);
      System.out.println("The balance is: " + bank.getBalance());

   }
```

```
}
```

Abstraction ရဲ့ အကျိုးကျေးဇူးများ

1. Complexity ကိုလျော့ချနိုင်တယ်
2. Code maintenance လုပ်ရတာလွယ်တယ်
3. Security ပိုကောင်းတယ်
4. Teamwork အတွက်အဆင်ပြေတယ်

Abstract Class vs Interface

Feature	Abstract Class	Interface
Variables	Any type	Only public static final
Methods	Abstract + Concrete	Only abstract (Java 8+)
Inheritance	Single inheritance	Multiple inheritance
Constructor	ရှိ	မရှိ
Access Modifiers	Any	Only public

✍ Short Notes:

✓ Android Development မှာ Adapter Pattern တွေမှာ အသုံးများတယ်

✓ Spring Framework မှာ Service Layer တွေမှာ အရေးပါတယ်

✓ Java Collections Framework မှာ List, Set, Map တွေမှာ အခြေခံအုတ်မြစ်ဖြစ်တယ်

Abstraction က Java Programming မှာ အရေးကြီးတဲ့ concept တစ်ခုဖြစ်ပြီး professional level programming တွေအတွက် မရှိမဖြစ်လိုအပ်ပါတယ်။

💻 Java Collections

Java Collections Framework မှာ Interface တွေက အရေးကြီးတဲ့ အခန်းကဏ္ဍမှာ ပါဝင်ပါတယ်။ Collection Interface တွေကို အသုံးပြုပြီး data structures တွေကို စနစ်တကျ စီမံနိုင်ပါတယ်။

Collections Framework ရဲ့ အဓိက Interfaces

1. Collection<E> Interface

Collection က root interface ဖြစ်ပြီး List, Set, Queue တို့အားလုံးက ဒီ interface ကို extend လုပ်ထားပါတယ်။

အဓိက method တွေ:

- add(E e),
- remove(Object o)
- size(),
- isEmpty()
- contains(Object o),
- clear()

2. List<E> Interface

List က ordered collection ဖြစ်ပြီး duplicate values တွေကို လက်ခံပါတယ်။

အသုံးများတဲ့ Classes: ArrayList, LinkedList, Vector

အဓိက method တွေ:

- get(int index)
- set(int index, E element)
- indexOf(Object o)

3. Set<E> Interface

Set က unique elements တွေကိုသာ သိမ်းပါတယ်။

အသုံးများတဲ့ Classes: HashSet, LinkedHashSet, TreeSet

အဓိက method တွေ:

- add(E e) (duplicate ဆိုရင် false return ပြန်မယ်)
- contains(Object o)

4. Queue<E> Interface
FIFO (First-In-First-Out) နည်းလမ်းနဲ့ အလုပ်လုပ်ပါတယ်။

အသုံးများတဲ့ Classes: LinkedList, PriorityQueue

အဓိက method တွေ:

- offer(E e) (element ထည့်ဖို့)
- poll() (ပထမဆုံး element ကို ဖယ်ရှားဖို့)

5. Map<K, V> Interface
Map က key-value pair တွေကို သိမ်းပါတယ်။

အသုံးများတဲ့ Classes: HashMap, LinkedHashMap, TreeMap

အဓိက method တွေ:

- put(K key, V value)
- get(Object key)
- containsKey(Object key)

ဥပမာ Code Snippet

Code:

```
import java.util.*;

public class CollectionsExample {
    public static void main(String[] args) {
        // List Example
        List<String> fruits = new ArrayList<>();
        fruits.add("Apple");
        fruits.add("Banana");
        System.out.println(fruits); // [Apple, Banana]

        // Set Example
        Set<Integer> numbers = new HashSet<>();
        numbers.add(10);
        numbers.add(20);
        numbers.add(10); // duplicate, not stored
        System.out.println(numbers); // [20, 10]

        // Map Example
        Map<String, Integer> ages = new HashMap<>();
        ages.put("Alice", 25);
        ages.put("Bob", 30);
```

```
        System.out.println(ages.get("Alice")); // 25
    }
}
```

✍ Short Notes:

- List → Ordered, Duplicates allowed
- Set → No Duplicates
- Queue → FIFO နည်းလမ်း
- Map → Key-Value Pair

Java Collections Framework ကို သေချာနားလည်ထားရင် data တွေကို ထိရောက်စွာ manage လုပ်နိုင်မှာ ဖြစ်ပါတယ်။

💻 List Interface

List Interface က **Collection** Framework ထဲမှာ အရေးကြီးတဲ့ အစိတ်အပိုင်းဖြစ်ပြီး **ordered collection** (အစဉ်လိုက်စီထားသော အချက်အလက်စု) ကို ကိုယ်စားပြုပါတယ်။ **List** က **duplicate values** တွေကို လက်ခံပြီး **index-based** နည်းလမ်းနဲ့ element တွေကို ထိန်းချုပ်နိုင်ပါတယ်။

List Interface ရဲ့ အဓိက အချက်များ

- အစဉ်လိုက် သိမ်းဆည်းပေးမယ် (Insertion Order ကို ထိန်းသိမ်းပေးပါတယ်)
- Duplicate တန်ဖိုးတွေကို ခွင့်ပြုမယ်
- Index သုံးပြီး element တွေကို access လုပ်နိုင်မယ်

List Interface ကို Implement လုပ်ထားတဲ့ အသုံးများတဲ့ Classes

- ArrayList → Dynamic array ကို အခြေခံထားတယ်၊ အများဆုံးသုံးတယ်
- LinkedList → Doubly-linked list ကို အခြေခံထားတယ်၊ frequent insertions/deletions အတွက် သင့်တော်တယ်
- Vector → Thread-safe ဖြစ်တယ်၊ သို့သော် အသုံးနည်းတယ်

List Interface ရဲ့ အဓိက Method များ

Method	ရှင်းလင်းချက်
add(E e)	List ရဲ့ နောက်ဆုံးမှာ element ထည့်မယ်
add(int index, E e)	သတ်မှတ်ထားတဲ့ index မှာ element ထည့်မယ်
get(int index)	သတ်မှတ်ထားတဲ့ index က element ကို return ပြန်မယ်

Method	ရှင်းလင်းချက်
set(int index, E e)	သတ်မှတ်ထားတဲ့ index မှာရှိတဲ့ element ကို update လုပ်မယ်
remove(int index)	သတ်မှတ်ထားတဲ့ index က element ကို ဖျက်မယ်
size()	List ထဲက element အရေအတွက်ကို return ပြန်မယ်
contains(Object o)	List ထဲမှာ element ရှိမရှိ စစ်မယ်
clear()	List ထဲက element အားလုံးကို ဖျက်မယ်

ဥပမာ-

Code:

```
import java.util.ArrayList;
import java.util.List;

public class ListExample {
   public static void main(String[] args) {
      // ArrayList ကို List Interface နဲ့ Declare လုပ်မယ်
      List<String> fruits = new ArrayList<>();

      // add() method နဲ့ element တွေထည့်မယ်
      fruits.add("Apple");
      fruits.add("Banana");
      fruits.add("Mango");
      fruits.add("Banana"); // Duplicate allowed

      System.out.println("Initial List: " + fruits);
      // Output: [Apple, Banana, Mango, Banana]

      // get() method နဲ့ element ရယူမယ်
      String firstFruit = fruits.get(0);
      System.out.println("First Fruit: " + firstFruit);
      // Output: Apple

      // set() method နဲ့ element ပြင်မယ်
      fruits.set(1, "Orange");
      System.out.println("After Update: " + fruits);
      // Output: [Apple, Orange, Mango, Banana]

      // remove() method နဲ့ element ဖျက်မယ်
      fruits.remove(2);
      System.out.println("After Removal: " + fruits);
      // Output: [Apple, Orange, Banana]

      // size() method နဲ့ list အရွယ်အစားစစ်မယ်
      System.out.println("List Size: " + fruits.size());
```

```
        // Output: 3
    }
}
```

ArrayList vs LinkedList

Feature	ArrayList	LinkedList
အခြေခံ Data Structure	Dynamic Array	Doubly-Linked List
Access Speed (get)	ပိုမြန်တယ် (O(1))	နှေးတယ် (O(n))
Insert/Delete Speed	နှေးတယ် (O(n))	ပိုမြန်တယ် (O(1))
Memory Usage	နည်းတယ်	ပိုသုံးတယ် (Node အတွက် extra memory)

ဘယ်အချိန်မှာ List ကိုသုံးမလဲ?

✓ **ArrayList** → **အများဆုံးသုံးတယ်**, random access လိုတဲ့အခါ (ဥပမာ: Database ကနေ data ဖတ်တဲ့အခါ)

✓ **LinkedList** → **အကြိမ်ရေများများ add/remove လုပ်ရတဲ့အခါ** (ဥပမာ: Real-time data processing)

List က Java Programming မှာ အရမ်းအသုံးဝင်တဲ့ Data Structure တစ်ခုဖြစ်ပြီး နားလည်ထားရင် Projects တွေမှာ ထိရောက်စွာ အသုံးချနိုင်ပါတယ်!

💻 Set Interface

Set Interface က Collection Framework ထဲမှာ အရေးကြီးတဲ့ အစိတ်အပိုင်းဖြစ်ပြီး unique elements only (ထူးခြားတဲ့ အချက်အလက်များသာ) ကို သိမ်းဆည်းပေးပါတယ်။ Set က duplicate values တွေကို လုံးဝမလက်ခံပါဘူး။

Set Interface ရဲ့အဓိက အချက်များ

- Duplicate တန်ဖိုးတွေကို လုံးဝမလက်ခံဘူး
- Insertion Order ကို အာမမခံဘူး (အမျိုးအစားပေါ်မူတည်ပါတယ်)
- null value ကို လက်ခံနိုင်တယ် (တစ်ခါတစ်ရံ)

Set Interface ကို Implement လုပ်ထားတဲ့ အသုံးများတဲ့ Classes

- **HashSet** → Hash table ကို အခြေခံထားတယ်၊ **အမြန်ဆုံး** access ရပြီး order မရှိဘူး
- **LinkedHashSet** → Insertion order ကို ထိန်းသိမ်းပေးတယ်
- **TreeSet** → Red-Black tree ကို အခြေခံထားတယ်၊ **sorted order** နဲ့ ပြပေးတယ်

Set Interface ရဲ့အဓိက Method များ

Method	ရှင်းလင်းချက်
add(E e)	Set ထဲကို element ထည့်မယ် (duplicate ဆိုရင် false return ပြန်မယ်)
remove(Object o)	Element ကို ဖျက်မယ်
contains(Object o)	Element ပါမပါ စစ်မယ်
size()	Set ထဲက element အရေအတွက်
isEmpty()	Set က ဗလာလားစစ်မယ်
clear()	Set ထဲက element အားလုံးကို ဖျက်မယ်

ဥပမာ-

Code:

```
import java.util.HashSet;
import java.util.Set;

public class SetExample {
    public static void main(String[] args) {
        // HashSet ကို Set Interface နဲ့ Declare လုပ်မယ်
        Set<String> fruits = new HashSet<>();

        // add() method နဲ့ element တွေထည့်မယ်
        fruits.add("Apple");
        fruits.add("Banana");
        fruits.add("Mango");
        fruits.add("Apple"); // Duplicate ဖြစ်လို့ မထည့်ဘူး

        System.out.println("Initial Set: " + fruits);
        // Output: [Apple, Banana, Mango] (order မသေချာ)

        // contains() method နဲ့ စစ်မယ်
        System.out.println("Contains Banana? " + fruits.contains("Banana"));
        // Output: true

        // remove() method နဲ့ ဖျက်မယ်
        fruits.remove("Mango");
        System.out.println("After Removal: " + fruits);
        // Output: [Apple, Banana]

        // size() method နဲ့ အရွယ်အစားစစ်မယ်
        System.out.println("Set Size: " + fruits.size());
        // Output: 2
    }
}
```

HashSet vs LinkedHashSet vs TreeSet

Feature	HashSet	LinkedHashSet	TreeSet
Ordering	No order	Insertion order	Sorted order
Performance	Fastest	Slightly slower	Slowest
null allowed	Yes	Yes	No
ဘယ်အချိန်သုံးမလဲ	Unique data အတွက်	Order ထိန်းဖို့လိုရင်	Sorted data လိုရင်

ဘယ်အချိန်မှာ Set ကိုသုံးမလဲ?

✓ ထူးခြားတဲ့တန်ဖိုးတွေသာ လိုအပ်တဲ့အခါ
✓ Duplicate ဖြစ်နေတာကို ရှောင်ချင်တဲ့အခါ
✓ အချက်အလက်တွေကို အမြန်ရှာဖွေဖို့လိုတဲ့အခါ

Set က Java Programming မှာ အရမ်းအသုံးဝင်တဲ့ Data Structure တစ်ခုဖြစ်ပြီး နားလည်ထားရင် Projects တွေမှာ ထိရောက်စွာ အသုံးချနိုင်ပါတယ်

💻 Map Interface

Map Interface က key-value pairs (သော့နှင့်တန်ဖိုးစုံ) တွေကို သိမ်းဆည်းဖို့အတွက် အသုံးပြုပါတယ်။ Collection Framework ထဲမှာ အရေးကြီးတဲ့ အစိတ်အပိုင်း ဖြစ်ပြီး List နဲ့ Set တို့နဲ့ မတူညီတဲ့ အောက်ပါအချက်တွေရှိပါတယ်။

Map Interface ရဲ့ အဓိက အချက်များ

1. **Key-Value Pair** အဖြစ် သိမ်းဆည်းမယ်
2. **Key တွေက unique ဖြစ်ရမယ်** (Duplicate keys မရှိဘူး)
3. **Value တွေက duplicate ဖြစ်နိုင်တယ်**
4. **Key တစ်ခုကို null တစ်ခါသာ ထည့်နိုင်တယ်** (HashMap/LinkedHashMap မှာ)
5. **Value တွေကို null မကန့်သတ်ဘူး**

Map Interface ကို Implement လုပ်ထားတဲ့ အသုံးများတဲ့ Classes

- **HashMap** → Hash table အခြေခံထားတယ်၊ **အမြန်ဆုံး** access ရပြီး order မရှိဘူး
- **LinkedHashMap** → Insertion order ကို ထိန်းသိမ်းပေးတယ်
- **TreeMap** → Red-Black tree အခြေခံထားတယ်၊ **key အလိုက် sort လုပ်ပေးတယ်**
- **Hashtable** → Thread-safe ဖြစ်တယ်၊ သို့သော် **အသုံးနည်းတယ်**

Map Interface ရဲ့ အဓိက Method များ

Method	ရှင်းလင်းချက်
put(K key, V value)	Map ထဲကို key-value pair ထည့်မယ်
get(Object key)	Key နဲ့သက်ဆိုင်တဲ့ value ကို return ပြန်မယ်
containsKey(Object key)	Key ရှိမရှိ စစ်မယ်
containsValue(Object value)	Value ရှိမရှိ စစ်မယ်

Method	ရှင်းလင်းချက်
remove(Object key)	Key နဲ့သက်ဆိုင်တဲ့ entry ကို ဖျက်မယ်
size()	Map ထဲက entry အရေအတွက်
keySet()	Key တွေအားလုံးကို Set အဖြစ် return ပြန်မယ်
values()	Value တွေအားလုံးကို Collection အဖြစ် return ပြန်မယ်
entrySet()	Key-value pairs အားလုံးကို Set အဖြစ် return ပြန်မယ်

ဥပမာ-

Code:

```
import java.util.HashMap;
import java.util.Map;

public class MapExample {
    public static void main(String[] args) {
        // HashMap ကို Map Interface နဲ့ Declare လုပ်မယ်
        Map<String, Integer> ageMap = new HashMap<>();

        // put() method နဲ့ entry တွေထည့်မယ်
        ageMap.put("Alice", 25);
        ageMap.put("Bob", 30);
        ageMap.put("Charlie", 35);
        ageMap.put("Alice", 26); // Duplicate key - value ကို update လုပ်မယ်

        System.out.println("Initial Map: " + ageMap);
        // Output: {Alice=26, Bob=30, Charlie=35}

        // get() method နဲ့ value ရယူမယ်
        int aliceAge = ageMap.get("Alice");
        System.out.println("Alice's Age: " + aliceAge);
        // Output: 26

        // containsKey() method နဲ့ စစ်မယ်
        System.out.println("Contains key 'Bob'? " + ageMap.containsKey("Bob"));
        // Output: true

        // remove() method နဲ့ entry ဖျက်မယ်
        ageMap.remove("Charlie");
        System.out.println("After Removal: " + ageMap);
        // Output: {Alice=26, Bob=30}

        // size() method နဲ့ အရွယ်အစားစစ်မယ်
        System.out.println("Map Size: " + ageMap.size());
        // Output: 2
```

```
  }
}
```

HashMap vs LinkedHashMap vs TreeMap

Feature	HashMap	LinkedHashMap	TreeMap
Ordering	No order	Insertion order	Sorted by keys
Performance	Fastest	Slightly slower	Slowest
null keys	1 allowed	1 allowed	Not allowed
null values	Allowed	Allowed	Allowed
ဘယ်အချိန်သုံးမလဲ	General use	Insertion order လိုရင်	Sorted keys လိုရင်

ဘယ်အချိန်မှာ Map ကိုသုံးမလဲ?

✓ **Key နဲ့ Value ဆက်စပ်မှုလိုတဲ့အခါ**

✓ **အချက်အလက်တွေကို key အရ အမြန်ရှာဖွေဖို့လိုတဲ့အခါ**

✓ **ဒေတာတွေကို အမျိုးအစားခွဲသိမ်းဖို့လိုတဲ့အခါ**

Map က Java Programming မှာ အရမ်းအသုံးဝင်တဲ့ Data Structure တစ်ခုဖြစ်ပြီး နားလည်ထားရင် Projects တွေမှာ ထိရောက်စွာ အသုံးချနိုင်ပါတယ်!

💻 Lambda

Java Lambda Expression (လမ်ဘဒါ အက်စ်ပရက်ရှင်း) သည် Java 8 မှစတင်မိတ်ဆက်ခဲ့တဲ့ အင်္ဂါရုပ်တစ်ခုဖြစ်ပြီး functional programming ကို ပိုမိုလွယ်ကူစွာ ရေးသားနိုင်ဖို့ ကူညီပေးပါတယ်။

Lambda ဆိုတာဘာလဲ?

Lambda သည် anonymous function (အမည်မဲ့ function) တစ်ခုဖြစ်ပြီး အောက်ပါအချက်တွေကို လုပ်ဆောင်နိုင်ပါတယ်:

- Method ကိုပါမယ့် parameter list တစ်ခုကို လက်ခံနိုင်ပါတယ်
- Body တစ်ခုရှိပါတယ်
- Return type ရှိပါတယ် (ဒါပေမယ့် မကြေညာဘဲထားနိုင်ပါတယ်)
- throws exception ပါနိုင်ပါတယ်

Lambda Syntax

Lambda expression ၏ အခြေခံပုံစံ :

```
Code:
(parameters) -> expression
```

သို့မဟုတ်

```
Code:
(parameters) -> { statements; }
```

ဥပမာများ

1. ရိုးရိုး Lambda

Code:
```
() -> System.out.println("Hello Lambda!");
```

2. Parameter တစ်ခုပါ Lambda

Code:
```
(name) -> System.out.println("Hello " + name)
```

3. Multiple Parameters

Code:
```
(a, b) -> a + b
```

4. Multiple Statements

Code:
```
(name) -> {
   String greeting = "Hello " + name;
   System.out.println(greeting);
   return greeting;
}
```

Lambda ကိုဘယ်မှာသုံးမလဲ?

Lambda expression တွေကို အထူးသဖြင့် functional interface တွေနဲ့ တွဲသုံးပါတယ်။

Functional interface ဆိုတာက method တစ်ခုပဲပါတဲ့ interface ဖြစ်ပါတယ်။

Runnable Interface နဲ့သုံးတဲ့ဥပမာ

Code:
```
// ရှေးရိုးနည်း
Runnable r1 = new Runnable() {
   @Override
   public void run() {
      System.out.println("Hello World 1");
   }
};

// Lambda သုံးပြီးရေးနည်း
Runnable r2 = ( ) -> System.out.println("Hello World 2");
```

Comparator Interface နဲ့သုံးတဲ့ဥပမာ

Code:
```
List<String> names = Arrays.asList("Alice", "Bob", "Charlie");

// ရှေးရိုးနည်း
Collections.sort(names, new Comparator<String>() {
   @Override
   public int compare(String a, String b) {
      return a.compareTo(b);
   }
});

// Lambda သုံးပြီးရေးနည်း
Collections.sort(names, (a, b) -> a.compareTo(b));
```

Lambda ရဲ့အားသာချက်များ

1. **ကုဒ်တိုတောင်းစေပါတယ်** - Anonymous class တွေထက် ပိုတိုတောင်းပါတယ်
2. **ဖတ်ရလွယ်ကူစေပါတယ်** - ကုဒ်ကိုပိုပြီးရှင်းလင်းစွာမြင်နိုင်ပါတယ်
3. **Functional programming ကိုအားပေးပါတယ်** - Java မှာ functional style နဲ့ရေးနိုင်ပါတယ်

Method References

Lambda နဲ့ဆက်စပ်တဲ့အခြားအင်္ဂါရုပ်တစ်ခုမှာ method reference ဖြစ်ပါတယ်။

ဥပမာ:

Code:
```
List<String> names = Arrays.asList("Alice", "Bob", "Charlie");

// Lambda သုံးပြီးရေးနည်း
names.forEach(name -> System.out.println(name));

// Method reference သုံးပြီးရေးနည်း
names.forEach(System.out::println);
```

Lambda expression များသည် Java programming ကို ပိုမိုတိုးတက်စေပြီး ကုဒ်ရေးသားမှုကို ပိုမိုလွယ်ကူစေပါတယ်။

Java Lambda အသုံးပြု၍ ပေါင်းခြင်းနှင့်နုတ်ခြင်း

1. Functional Interface သတ်မှတ်ခြင်း

ပထမဆုံး အပေါင်း အနှုတ်လုပ်ဆောင်ချက်များအတွက် functional interface တစ်ခုကိုသတ်မှတ်ပါမယ်။

Code:
```
@FunctionalInterface
interface MathOperation {
   int operate(int a, int b);
}
```

2. Lambda Expressions အသုံးပြုခြင်း

Code:
```
public class LambdaMathExample {
   public static void main(String[] args) {
      // Lambda expression အသုံးပြု၍ ပေါင်းခြင်း
      MathOperation addition = (a, b) -> a + b;

      // Lambda expression အသုံးပြု၍ နုတ်ခြင်း
      MathOperation subtraction = (a, b) -> a - b;

      // နံပတ်နှစ်ခုများ
      int num1 = 10;
      int num2 = 5;

      // ပေါင်းခြင်းကိုအသုံးပြုခြင်း
      System.out.println(num1 + " + " + num2 + " = " + addition.operate(num1, num2));

      // နုတ်ခြင်းကိုအသုံးပြုခြင်း
      System.out.println(num1 + " - " + num2 + " = " + subtraction.operate(num1, num2));
   }
}
```

3. Output ရလဒ်

အထက်ပါကုဒ်ကို run ပါက အောက်ပါအတိုင်းရလဒ်ထွက်ပါမယ်:

Console:
```
10 + 5 = 15
10 - 5 = 5
```

4. နောက်ထပ်နည်းလမ်း - Method Parameter အဖြစ် Lambda ကိုသုံးခြင်း

Code:
```
public class LambdaMathExample2 {
  public static void main(String[] args) {
    int num1 = 20;
    int num2 = 8;

    // Lambda expression ကို တိုက်ရိုက်သုံးခြင်း
    System.out.println(num1 + " + " + num2 + " = " + operate(num1, num2, (a, b) -> a + b));
    System.out.println(num1 + " - " + num2 + " = " + operate(num1, num2, (a, b) -> a - b));
  }

  public static int operate(int a, int b, MathOperation operation) {
    return operation.operate(a, b);
  }
}
```

5. ရှင်းလင်းချက်

1. MathOperation သည် functional interface တစ်ခုဖြစ်ပြီး operate ဆိုတဲ့ method တစ်ခုပါရှိပါတယ်။
2. addition နှင့် subtraction တို့သည် lambda expression များဖြစ်ကြပါတယ်။
3. Lambda expression များသည် interface တစ်ခုရဲ့ method ကိုအကောင်အထည်ဖော်ပေးပါတယ်။
4. ဒီနည်းလမ်းဖြင့် ကုဒ်များကိုပိုမိုတိုတောင်းစွာရေးသားနိုင်ပြီး ဖတ်ရှုရလွယ်ကူပါတယ်။

Lambda expression များသည် Java တွင် functional programming ကိုအသုံးပြုရာတွင် အလွန်အသုံးဝင်ပါတယ်။

လေ့ကျင့်ရန် -

Java Lambda အသုံးပြု၍ မြှောက်ခြင်း နဲ့ စားခြင်းကို လေ့ကျင့်ပါ။

Output:

10 * 5 = 150

10 /2 = 5.0

💻 File Handling

File နဲ့အလုပ်လုပ်ဖို့ java.io package ထဲက class တွေကိုအသုံးပြုပါတယ်။ File တွေကို ဖတ်ခြင်း၊ ရေးခြင်း၊ ဖျက်ခြင်းနဲ့ အခြား file operations တွေလုပ်ဆောင်နိုင်ပါတယ်။

File Class အဓိက အချက်များ

1. **File object** တစ်ခုက file သို့မဟုတ် directory path ကိုကိုယ်စားပြုတယ်
2. **File တည်ရှိမရှိ** စစ်ဆေးနိုင်တယ်
3. **File အချက်အလက်များ** (အရွယ်အစား၊ နောက်ဆုံးပြင်ဆင်မှုအချိန်) ရယူနိုင်တယ်
4. **File ဖန်တီးခြင်း၊ ဖျက်ခြင်း၊ အမည်ပြောင်းခြင်း** လုပ်ဆောင်နိုင်တယ်

အဓိက File Operations

1. File ဖန်တီးခြင်း

Code:
```
import java.io.File;
import java.io.IOException;

public class FileExample {
   public static void main(String[] args) {
      try {
         File myFile = new File("example.txt");
         if (myFile.createNewFile()) {
            System.out.println("File created: " + myFile.getName());
         } else {
            System.out.println("File already exists.");
         }
      } catch (IOException e) {
         System.out.println("An error occurred.");
         e.printStackTrace();
      }
   }
}
```

2. File အချက်အလက်များရယူခြင်း

Code:

```
File myFile = new File("example.txt");
if (myFile.exists()) {
   System.out.println("File name: " + myFile.getName());
   System.out.println("Absolute path: " + myFile.getAbsolutePath());
   System.out.println("File size (bytes): " + myFile.length());
   System.out.println("Last modified: " + new Date(myFile.lastModified()));
} else {
   System.out.println("File does not exist.");
}
```

3. File ဖတ်ခြင်းနဲ့ရေးခြင်း

File ရေးခြင်း (FileWriter အသုံးပြုခြင်း)

Code:

```
import java.io.FileWriter;

try (FileWriter writer = new FileWriter("example.txt")) {
   writer.write("Hello Java File Handling!");
   System.out.println("Successfully wrote to the file.");
} catch (IOException e) {
   System.out.println("An error occurred.");
   e.printStackTrace();
}
```

File ဖတ်ခြင်း (Scanner အသုံးပြုခြင်း)

Code:

```
import java.io.File;
import java.util.Scanner;

try {
   File myFile = new File("example.txt");
   Scanner myReader = new Scanner(myFile);
   while (myReader.hasNextLine()) {
      String data = myReader.nextLine();
      System.out.println(data);
   }
   myReader.close();
} catch (FileNotFoundException e) {
   System.out.println("File not found.");
   e.printStackTrace();
}
```

Directory အလုပ်လုပ်ခြင်း

Directory ဖန်တီးခြင်း

Code:
```
File dir = new File("myDirectory");
if (dir.mkdir()) {
   System.out.println("Directory created successfully");
} else {
   System.out.println("Failed to create directory");
}
```

Directory ထဲက File များစာရင်းရယူခြင်း

Code:
```
File dir = new File("myDirectory");
String[] files = dir.list();
for (String file : files) {
   System.out.println(file);
}
```

အရေးကြီးသော သတိပြုရန်များ

1. File operations တွေမှာ **IOException** ကို အမြဲ handle လုပ်ပါ
2. Resources တွေကို **try-with-resources** နဲ့ အသုံးပြုပါ (Java 7 နဲ့အထက်)
3. File path တွေမှာ **absolute path** နဲ့ **relative path** ကိုသတိထားပါ

ဘယ်အချိန်မှာ File Handling ကိုသုံးမလဲ?

✓ **Configuration files** တွေဖတ်ဖို့

✓ **အချက်အလက်များ** သိမ်းဆည်းဖို့

✓ **Log files** တွေရေးဖို့

✓ **အချက်အလက်များ** import/export လုပ်ဖို့

File Handling ကို နားလည်ထားရင် သင့် program တွေကို ပိုမိုအဆင့်မြင့်စွာ ရေးသားနိုင်မယ်ဖြစ်ပါတယ်!

💻 Exception Handling

Exception Handling ဆိုတာက program လုပ်ဆောင်နေစဉ် ဖြစ်ပေါ်လာနိုင်တဲ့ အမှားတွေ (errors) ကို ထိန်းချုပ်ဖြေရှင်းနည်းပါ။ Exception တွေကို ကောင်းစွာ handle လုပ်ခြင်းဖြင့် program crash မဖြစ်အောင် ကာကွယ်နိုင်ပါတယ်။

Exception အမျိုးအစားများ

1. Checked Exceptions (Compile Time Exceptions)
   - Compile time မှာ စစ်ဆေးတဲ့ exceptions
   - **ဥပမာ:** IOException, SQLException, ClassNotFoundException

2. Unchecked Exceptions (Runtime Exceptions)
   - Runtime မှာ ဖြစ်ပေါ်တဲ့ exceptions
   - **ဥပမာ:** NullPointerException, ArrayIndexOutOfBoundsException, ArithmeticException

3. Errors
   - System level errors (ပြင်ဆင်လို့မရတဲ့ အမှားများ)
   - **ဥပမာ:** OutOfMemoryError, StackOverflowError

Exception Handling အဓိက Keywords

Keyword	ရှင်းလင်းချက်
try	Exception ဖြစ်နိုင်တဲ့ code block
catch	Exception ကို handle လုပ်မယ့် block
finally	Exception ဖြစ်ဖြစ်/မဖြစ်ဖြစ် အမြဲ run မယ့် block
throw	Manual exception throw လုပ်ဖို့
throws	Method က exception throw လုပ်နိုင်ကြောင်း declare လုပ်ဖို့

Exception Handling ဥပမာများ

1. အခြေခံ try-catch အသုံးပြုပုံ

Code:
```
try {
   int result = 10 / 0;  // ArithmeticException ဖြစ်မယ်
} catch (ArithmeticException e) {
   System.out.println("သုညနဲ့စားလို့မရဘူး: " + e.getMessage());
}
```

2. Multiple catch blocks

Code:
```
try {
   int[] arr = new int[5];
   arr[10] = 50;  // ArrayIndexOutOfBoundsException
} catch (ArrayIndexOutOfBoundsException e) {
   System.out.println("Array index မှားနေပါတယ်");
} catch (Exception e) {
   System.out.println("တခြား exception တစ်ခုခုဖြစ်နေပါတယ်");
}
```

3. finally block အသုံးပြုပုံ

Code:
```
try {
   // File operations
} catch (IOException e) {
   System.out.println("File error: " + e);
} finally {
   System.out.println("ဒီ code အမြဲ run မယ်");
}
```

4. throw နဲ့ custom exception

Code:
```
void checkAge(int age) {
   if (age < 18) {
      throw new ArithmeticException("အသက်မပြည့်သေးပါ");
   } else {
      System.out.println("ဝင်ရောက်ခွင့်ပြုပါသည်");
   }
```

```
}

public static void main(String[] args) {
   checkAge(15);  // Exception throw လုပ်မယ်
}
```

5. throws keyword အသုံးပြုပုံ

Code:

```
void readFile() throws IOException {
   File file = new File("nonexistent.txt");
   FileReader fr = new FileReader(file);  // IOException ဖြစ်နိုင်တယ်
}
```

Custom Exception ဖန်တီးနည်း

Code:

```
class MyCustomException extends Exception {
   public MyCustomException(String message) {
      super(message);
   }
}

// အသုံးပြုပုံ
try {
   throw new MyCustomException("ကျွန်တော့် custom exception");
} catch (MyCustomException e) {
   System.out.println(e.getMessage());
}
```

✍ Short Notes:

1. **Specific exceptions** တွေကို ဦးစားပေးဖမ်းပါ
2. **Resource တွေကို** finally block မှာ close လုပ်ပါ
3. **Exception messages** တွေကို ရှင်းလင်းစွာရေးပါ
4. **အလွန်အကျွံ try-catch** မလုပ်ပါနဲ့
5. **Logging** ကိုအသုံးပြုပါ

Java Exception Handling ကို ကောင်းစွာနားလည်ထားရင် သင့် program တွေကို ပိုမိုခိုင်မာစွာ ရေးသားနိုင်မယ်ဖြစ်ပါတယ်!

💻 Thread

Java မှာ **Thread** ဆိုတာက program execution ၏ အသေးငယ်ဆုံး unit တစ်ခုဖြစ်ပြီး **multitasking** လုပ်ဆောင်နိုင်ဖို့အတွက် အသုံးပြုပါတယ်။ Thread တွေကို အသုံးပြုခြင်းဖြင့် ကျွန်တော်တို့ program တစ်ခုတည်းမှာ တစ်ချိန်တည်း task များစွာကို လုပ်ဆောင်နိုင်ပါတယ်။

Thread အမျိုးအစားများ

1. User Thread (Non-Daemon Thread)
   - Main program ရဲ့ အလုပ်တွေကို လုပ်ဆောင်တဲ့ thread
   - JVM က thread အားလုံး ပြီးဆုံးမှသာ program က ရပ်တန့်မယ်
2. Daemon Thread
   - Background service တွေအတွက် အသုံးပြုတဲ့ thread
   - User thread တွေ အားလုံးပြီးသွားရင် အလိုအလျောက် ရပ်တန့်သွားမယ်

Thread ဖန်တီးနည်းများ

1. Thread Class ကို Extend လုပ်ခြင်း

Code:
```
class MyThread extends Thread {
    public void run() {
        System.out.println("Thread is running...");
    }
}

public class Main {
    public static void main(String[] args) {
        MyThread t1 = new MyThread();
        t1.start();  // Thread စပါမယ်

    }
}
```

2. Runnable Interface ကို Implement လုပ်ခြင်း (ပိုကောင်းတဲ့နည်း)

Code:
```
class MyRunnable implements Runnable {
    public void run() {
        System.out.println("Runnable thread is running...");
    }
}

public class Main {
    public static void main(String[] args) {
        Thread t2 = new Thread(new MyRunnable());
        t2.start();
    }
}
```

Thread အဓိက Method များ

Method	**ရှင်းလင်းချက်**
start()	Thread ကို စတင်မယ်
run()	Thread ရဲ့ အလုပ်တွေကို သတ်မှတ်မယ်
sleep(long millis)	Thread ကို သတ်မှတ်ထားတဲ့ အချိန်ကြာမှ ပြန်စမယ်
join()	အခြား thread ပြီးတဲ့အထိ စောင့်မယ်
interrupt()	Thread ကို အနှောင့်အယှက်ပေးမယ်
isAlive()	Thread အလုပ်လုပ်နေဆဲလားစစ်မယ်

Thread Synchronization (အချိန်ကိုက်ညှိခြင်း)

Multiple threads တွေ shared resource တစ်ခုကို တစ်ချိန်တည်း access လုပ်တဲ့အခါ **race condition** ဖြစ်နိုင်ပါတယ်။ ဒါကို ကာကွယ်ဖို့ synchronization လုပ်ရပါမယ်။

synchronized Method အသုံးပြုပုံ

Code:

```
class Counter {
   private int count = 0;

   public synchronized void increment() {
      count++;
   }

   public int getCount() {
      return count;
   }
}
```

synchronized Block အသုံးပြုပုံ

Code:

```
public void doWork() {
   synchronized(this) {
      // synchronized code block
   }
}
```

Thread Life Cycle (သက်တမ်းစက်ဝန်း)

1. **New** - Thread object ဖန်တီးပြီး start() မခေါ်ရသေးခြင်း
2. **Runnable** - start() ခေါ်ပြီး CPU ရဖို့စောင့်နေခြင်း
3. **Running** - CPU ရပြီး execute လုပ်နေခြင်း
4. **Blocked/Waiting** - I/O operation သို့မဟုတ် synchronization ကြောင့်စောင့်နေခြင်း
5. **Terminated** - run() method ပြီးဆုံးခြင်း

Thread Pool အကြောင်း

Thread တွေကို ထပ်ခါထပ်ခါ create/destroy လုပ်တာက performance ကိုထိခိုက်စေနိုင်ပါတယ်။ Thread pool ကို အသုံးပြုခြင်းဖြင့် ဒီပြဿနာကို ဖြေရှင်းနိုင်ပါတယ်။

Code:
```
ExecutorService executor = Executors.newFixedThreadPool(5);

for (int i = 0; i < 10; i++) {
   Runnable worker = new WorkerThread("" + i);
   executor.execute(worker);
}
executor.shutdown();
```

ဘယ်အချိန်မှာ Thread ကိုသုံးမလဲ?

✓ **GUI applications** မှာ responsiveness မြင့်ဖို့

✓ **Server applications** မှာ multiple clients ကို handle လုပ်ဖို့

✓ **ကြာမြင့်တဲ့ operations** တွေကို background မှာ run ဖို့

✓ **Parallel processing** လုပ်ဖို့

Java Thread ကို ကောင်းစွာနားလည်ထားရင် သင့် program တွေကို ပိုမိုထိရောက်စွာ ရေးသားနိုင်မယ်ဖြစ်ပါတယ်!

💻 Regular Expression (Regex)

Regular Expression (Regex) ဆိုတာ **စာသားပုံစံများကို ရှာဖွေရန်၊ စစ်ဆေးရန်နှင့် ပြုပြင်ရန်** အသုံးပြုတဲ့ စံသတ်မှတ်ချက်တစ်ခုဖြစ်ပါတယ်။ Java မှာ java.util.regex package ကို အသုံးပြုပြီး Regex များကို အလွယ်တကူ အသုံးပြုနိုင်ပါတယ်။

1. Java Regex အခြေခံများ

Pattern Class နှင့် Matcher Class

Code:
```
import java.util.regex.*;

String text = "Java Programming 2023";
Pattern pattern = Pattern.compile("\\d+"); // ဂဏန်းများရှာမယ်
Matcher matcher = pattern.matcher(text);

while (matcher.find()) {
   System.out.println("တွေ့ရှိခဲ့သည်: " + matcher.group());
}
// ရလဒ်: တွေ့ရှိခဲ့သည်: 2023
```

String Class မှာ တိုက်ရိုက်သုံးနည်း

Code:
```
String email = “user@example.com”;
89oolean isValid = email.matches(“^[\\w.-]+@[\\w.-]+\\.[a-z]{2,6}$”);
System.out.println(“Email မှန်ကန်မှု: ” + isValid);
// ရလဒ်: Email မှန်ကန်မှု: true
```

2. အသုံးများသော Regex Patterns

Pattern	ဖော်ပြချက်	ဥပမာ
\\d	ဂဏန်းတစ်လုံး (0-9)	"Java8" → 8
\\D	ဂဏန်းမဟုတ်သော အက္ခရာ	"Java8" → J,a,v,a

Pattern	ဖော်ပြချက်	ဥပမာ
\\w	Word Character (a-z, A-Z, 0-9, _)	"user_name123" → u,s,e,r,_,n,a,m,e,1,2,3
\\W	Word Character မဟုတ်သော အက္ခရာ	"a@b#c" → @,#
\\s	White Space (space, tab, newline)	"a b c" → (space)
\\S	White Space မဟုတ်သော အက္ခရာ	"a b c" → a,b,c
[abc]	a, b သို့မဟုတ် c	"apple" → a,p,p
[^abc]	a, b, c မဟုတ်သော အက္ခရာ	"apple" → l,e
^pattern	စာကြောင်းအစနှင့် ကိုက်ညီမှု	^Java → "Java" (အစမှာရှိမယ်)
pattern$	စာကြောင်းအဆုံးနှင့် ကိုက်ညီမှု	Script$ → "JavaScript" (အဆုံးမှာရှိမယ်)

3. အသုံးဝင်သော Regex ဥပမာများ

(က) Email Validation

Code:
```
String emailRegex = "^[\\w.-]+@[\\w.-]+\\.[a-z]{2,6}$";
String email = "test.email+2023@gmail.com";
System.out.println(email.matches(emailRegex)); // true
```

ခ) Phone Number Validation (မြန်မာဖုန်းနံပါတ်)

Code:
```
String phoneRegex = "^(09|\\+?959)\\d{9}$";
String phone = "09123456789";
System.out.println(phone.matches(phoneRegex)); // true
```

(ဂ) Password Strength Checker

Code:
```
// အနည်းဆုံး 8 လုံး၊ စာကြီး၊ စာသေး၊ ဂဏန်း၊ အထူးသင်္ကေတ တစ်ခုပါရမယ်
String passwordRegex = "^(?=.*[A-Z])(?=.*[a-z])(?=.*\\d)(?=.*[@#$%^&+=]).{8,}$";
String password = "Passw0rd#";
System.out.println(password.matches(passwordRegex)); // true
```

(ဃ) HTML Tag ဖယ်ရှားခြင်း

Code:
```
String html = "<p>Hello <b>World</b></p>";
String cleanText = html.replaceAll("<[^>]*>", "");
```

```
System.out.println(cleanText); // Hello World
```

Grouping နှင့် Capturing

Code:

```
String dateText = "2023-12-31";
Pattern datePattern = Pattern.compile("(\\d{4})-(\\d{2})-(\\d{2})");
Matcher dateMatcher = datePattern.matcher(dateText);

if (dateMatcher.matches()) {
    System.out.println("နှစ်: " + dateMatcher.group(1)); // 2023
    System.out.println("လ: " + dateMatcher.group(2));    // 12
    System.out.println("ရက်: " + dateMatcher.group(3));  // 31
}
```

5. Regex Flags

Flag	အဓိပ္ပါယ်
Pattern.CASE_INSENSITIVE	အကြီးအသေး မခွဲခြားဘဲရှာမယ်
Pattern.MULTILINE	Multi-line mode တွင်သုံးမယ်
Pattern.DOTALL	Newline အပါအဝင် ကိုက်ညီမှုရှာမယ်

Code:

```
Pattern.compile("java", Pattern.CASE_INSENSITIVE).matcher("JAVA").find(); // true
```

အရေးကြီးသော Note များ

1. Java မှာ \\ ကို Escape Character အဖြစ်သုံးရပါတယ် (ဥပမာ - \\d ဆိုရင် \d ကိုဆိုလိုတာပါ)
2. Pattern နဲ့ Matcher ကို ကြိမ်ဖန်များစွာအသုံးပြုမယ်ဆိုရင် Pattern.compile() ကို အရင်လုပ်ပါ
3. String.matches() method က **တစ်ခုလုံးကိုက်ညီမှု** true ပြန်ပေးပါတယ်

အချုပ်ဆိုရသော် → Java Regex ကို ကျွမ်းကျင်စွာအသုံးပြုနိုင်ရင် **Text Processing** လုပ်ငန်းတွေကို အထူးလွယ်ကူမြန်ဆန်စွာ လုပ်ဆောင်နိုင်ပါတယ်!

💻 Random Number နှင့် Random String

1. Random Number

(က) java.util.Random Class အသုံးပြုခြင်း

Code:
```
import java.util.Random;

Random random = new Random();

// 0 မှ 99 အထိ ကျပန်းဂဏန်း
int randomNumber = random.nextInt(100);
System.out.println("ကျပန်းဂဏန်း: " + randomNumber);

// 1 မှ 6 အထိ (အန်စာတုံးလို)
int diceRoll = random.nextInt(6) + 1;
System.out.println("အန်စာတုံးလိုမ့်ခြင်း: " + diceRoll);

// ကျပန်း Boolean တန်ဖိုး
93oolean randomBool = random.nextBoolean();
System.out.println("ကျပန်း Boolean: " + randomBool);
```

ခ) Math.random() Method အသုံးပြုခြင်း

Code:
```
// 0.0 (အပါအဝင်) မှ 1.0 (မပါဝင်) အထိ
double randomDouble = Math.random();
System.out.println("Math.random() ရလဒ်: " + randomDouble);

// 1 မှ 100 အထိ ပြောင်းလဲခြင်း
int randomInt = (int)(Math.random() * 100) + 1;
System.out.println("1-100 အတွင်းကျပန်းဂဏန်း: " + randomInt);
```

(ဂ) Java 8+ ThreadLocalRandom Class

Code:
```
import java.util.concurrent.ThreadLocalRandom;

// ပိုမိုကောင်းမွန်သော နည်းလမ်း (Multi-thread အခြေအနေများအတွက်)
```

```
int randomNum = ThreadLocalRandom.current().nextInt(1, 101); // 1-100
System.out.println("ThreadLocalRandom: " + randomNum);
```

2. Random String (ကျပန်းစာသား) ထုတ်ယူနည်း

(က) အခြေခံနည်းလမ်း

Code:

```
String chars = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";
Random random = new Random();
StringBuilder sb = new StringBuilder();

for (int i = 0; i < 10; i++) {
   int index = random.nextInt(chars.length());
   sb.append(chars.charAt(index));
}

String randomString = sb.toString();
System.out.println("ကျပန်းစာသား: " + randomString);
```

(ခ) Java 8 Stream အသုံးပြုခြင်း

Code:

```
import java.util.stream.Collectors;
import java.util.stream.IntStream;

String randomStr = IntStream.range(0, 8)
   .mapToObj(i -> random.nextInt(36) < 26 ?
      String.valueOf((char)(random.nextInt(26) + 'a')) :
      String.valueOf(random.nextInt(10)))
   .collect(Collectors.joining());

System.out.println("Stream ဖြင့်ကျပန်းစာသား: " + randomStr);
```

(ဂ) UUID အသုံးပြုခြင်း

Code:

```
import java.util.UUID;

String uuid = UUID.randomUUID().toString();
System.out.println("UUID: " + uuid);
// ဥပမာ: 550e8400-e29b-41d4-a716-446655440000
```

3. အထူးကျပန်းစာသားများထုတ်ယူနည်း

(က) စာလုံးကြီးများသာ

Code:

```
String upperCaseRandom = random.ints(65, 91)
   .limit(8)
   .collect(StringBuilder::new, StringBuilder::appendCodePoint, StringBuilder::append)
   .toString();

System.out.println("စာလုံးကြီးများသာ: " + upperCaseRandom);
```

ခ) ဂဏန်းများသာ

Code:

```
String numericRandom = random.ints(48, 58)
   .limit(6)
   .collect(StringBuilder::new, StringBuilder::appendCodePoint, StringBuilder::append)
   .toString();

System.out.println("ဂဏန်းများသာ: " + numericRandom);
```

(ဂ) အထူးသင်္ကေတများပါဝင်သော

Code:

```
String specialChars = "!@#$%^&*()_+";
String allChars = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789" + speci
alChars;

String complexRandom = random.ints(0, allChars.length())
   .limit(12)
   .map(allChars::charAt)
   .collect(StringBuilder::new, StringBuilder::appendCodePoint, StringBuilder::append)
   .toString();

System.out.println("အထူးစာသား: " + complexRandom);
```

4. အသုံးဝင်သော နမူနာများ

(က) လုံခြုံရေးအတွက် ကျပန်းစာသား (Password Generator)

Code:

```
public static String generateSecurePassword(int length) {
   String 95owercase = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
   String 95owercase = "abcdefghijklmnopqrstuvwxyz";
   String numbers = "0123456789";
   String specialChars = "!@#$%^&*()_+";
   String combined = 95owercase + 95owercase + numbers + specialChars;

   Random random = new Random();
```

```
    StringBuilder sb = new StringBuilder();

    for (int I = 0; I < length; i++) {
        int index = random.nextInt(combined.length());
        sb.append(combined.charAt(index));
    }

    return sb.toString();
}

System.out.println("လုံခြုံရေးစာသား: " + generateSecurePassword(12));
```

(ခ) OTP (One-Time Password) ထုတ်ယူခြင်း

Code:

```
public static String generateOTP(int length) {
    String numbers = "0123456789";
    Random random = new Random();
    StringBuilder sb = new StringBuilder();

    for (int i = 0; i < length; i++) {
        sb.append(numbers.charAt(random.nextInt(numbers.length())));
    }

    return sb.toString();
}

System.out.println("OTP နံပါတ်: " + generateOTP(6));
```

အရေးကြီးသော Note များ

1. java.util.Random က **pseudo-random** ဖြစ်ပါတယ်
   (တစ်ခါတစ်ရံတူညီတဲ့အစီအစဉ်အတိုင်းထုတ်ပေးနိုင်ပါတယ်)
2. လုံခြုံရေးအရအရေးကြီးတဲ့အခါမျိုးမှာ java.security.SecureRandom ကိုသုံးပါ
3. Math.random() က double တန်ဖိုးပဲထုတ်ပေးပါတယ်
4. Multi-threaded application တွေမှာ ThreadLocalRandom ကိုသုံးပါ

အချုပ်ဆိုရသော် → Java မှာ ကျပန်းဂဏန်းနဲ့စာသားများကို အလွယ်တကူထုတ်ယူနိုင်ပြီး အမျိုးမျိုးသော အခြေအနေများအတွက် သင့်တော်စွာ အသုံးပြုနိုင်ပါတယ်!

💻 Java Console and Database

1. Database Connection

Code:

```
import java.sql.*;

public class DBConnection {
    private static final String URL = "jdbc:mysql://localhost:3306/mydatabase";
    private static final String USER = "root";
    private static final String PASSWORD = "";

    public static Connection getConnection() throws SQLException {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            return DriverManager.getConnection(URL, USER, PASSWORD);
        } catch (ClassNotFoundException e) {
            throw new SQLException("MySQL Driver not found", e);
        }
    }
}
```

2. Data Insert (အချက်အလက်ထည့်သွင်းခြင်း)

Code:

```
public void insertData(String name, int age) {
    String sql = "INSERT INTO users (name, age) VALUES (?, ?)";

    try (Connection conn = DBConnection.getConnection();
         PreparedStatement pstmt = conn.prepareStatement(sql)) {

        pstmt.setString(1, name);
        pstmt.setInt(2, age);
        pstmt.executeUpdate();

        System.out.println("အချက်အလက် ထည့်သွင်းပြီးပါပြီ");
    } catch (SQLException e) {
        System.out.println("အချက်အလက် ထည့်သွင်းရာတွင် အမှားတစ်ခုဖြစ်ပေါ် ခဲ့ပါသည်: " + e.getMessage());
    }
}
```

3. Data Reading (အချက်အလက်ဖတ်ခြင်း)

Code:

```
public void readData() {
    String sql = "SELECT * FROM users";

    try (Connection conn = DBConnection.getConnection();
         Statement stmt = conn.createStatement();
```

```
      ResultSet rs = stmt.executeQuery(sql)) {

     System.out.println("ID\tName\tAge");
     System.out.println("-------------------");

     while (rs.next()) {
        int id = rs.getInt("id");
        String name = rs.getString("name");
        int age = rs.getInt("age");

        System.out.println(id + "\t" + name + "\t" + age);
     }
   } catch (SQLException e) {
     System.out.println("အချက်အလက် ဖတ်ရာတွင် အမှားတစ်ခုဖြစ်ပေါ် ခဲ့ပါသည်: " + e.getMessage());
   }
}
```

4. Data Update (အချက်အလက်ပြင်ဆင်ခြင်း)

Code:

```
public void updateData(int id, String newName, int newAge) {
   String sql = "UPDATE users SET name = ?, age = ? WHERE id = ?";

   try (Connection conn = DBConnection.getConnection();
       PreparedStatement pstmt = conn.prepareStatement(sql)) {

      pstmt.setString(1, newName);
      pstmt.setInt(2, newAge);
      pstmt.setInt(3, id);

      int affectedRows = pstmt.executeUpdate();

      if (affectedRows > 0) {
         System.out.println("အချက်အလက် ပြင်ဆင်ပြီးပါပြီ");
      } else {
         System.out.println("ပြင်ဆင်ရန် အချက်အလက် မတွေ့ပါ");
      }
   } catch (SQLException e) {
      System.out.println("အချက်အလက် ပြင်ဆင်ရာတွင် အမှားတစ်ခုဖြစ်ပေါ် ခဲ့ပါသည်: " + e.getMessage());
   }
}
```

5. Data Delete (အချက်အလက်ဖျက်ခြင်း)

Code:

```
public void deleteData(int id) {
   String sql = "DELETE FROM users WHERE id = ?";

   try (Connection conn = DBConnection.getConnection();
       PreparedStatement pstmt = conn.prepareStatement(sql)) {

      pstmt.setInt(1, id);
```

```
        int affectedRows = pstmt.executeUpdate();

        if (affectedRows > 0) {
            System.out.println("အချက်အလက် ဖျက်ပြီးပါပြီ");
        } else {
            System.out.println("ဖျက်ရန် အချက်အလက် မတွေ့ပါ");
        }
    } catch (SQLException e) {
        System.out.println("အချက်အလက် ဖျက်ရာတွင် အမှားတစ်ခုဖြစ်ပေါ် ခဲ့ပါသည်: " + e.getMessage());
    }
}
```

Console Application နဲ့ ပေါင်းစပ်အသုံးပြုနည်း

Code:

```
import java.util.Scanner;

public class MainApp {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        DatabaseOperations dbOps = new DatabaseOperations();

        while (true) {
            System.out.println("\n1. အချက်အလက်ထည့်သွင်းမယ်");
            System.out.println("2. အချက်အလက်ဖတ်မယ်");
            System.out.println("3. အချက်အလက်ပြင်ဆင်မယ်");
            System.out.println("4. အချက်အလက်ဖျက်မယ်");
            System.out.println("5. ထွက်မယ်");
            System.out.print("လုပ်ဆောင်ချက် ရွေးချယ်ပါ (1-5): ");

            int choice = scanner.nextInt();
            scanner.nextLine();  // Clear buffer

            switch (choice) {
                case 1:
                    System.out.print("နာမည်ထည့်ပါ: ");
                    String name = scanner.nextLine();
                    System.out.print("အသက်ထည့်ပါ: ");
                    int age = scanner.nextInt();
                    dbOps.insertData(name, age);
                    break;

                case 2:
                    dbOps.readData();
                    break;

                case 3:
                    System.out.print("ပြင်ဆင်မည့် ID ထည့်ပါ: ");
```

```
                int updateId = scanner.nextInt();
                scanner.nextLine();
                System.out.print("အသစ်နာမည်ထည့်ပါ: ");
                String newName = scanner.nextLine();
                System.out.print("အသစ်အသက်ထည့်ပါ: ");
                int newAge = scanner.nextInt();
                dbOps.updateData(updateId, newName, newAge);
                break;

            case 4:
                System.out.print("ဖျက်မည့် ID ထည့်ပါ: ");
                int deleteId = scanner.nextInt();
                dbOps.deleteData(deleteId);
                break;

            case 5:
                System.out.println("ပရိုဂရမ်မှ ထွက်ပါပြီ");
                System.exit(0);

            default:
                System.out.println("မှားယွင်းသော ရွေးချယ်မှု");
        }
    }
}
```

အရေးကြီးသော Note များ

1. **JDBC Driver** ထည့်သွင်းရန် လိုအပ်ပါတယ် (MySQL အတွက် mysql-connector-java.jar)
2. try-with-resources ကို အသုံးပြုထားပါတယ် (Connection, Statement, ResultSet တို့ကို အလိုအလျောက် close လုပ်ပေးမယ်)
3. SQL Injection ကာကွယ်ရန် PreparedStatement ကို အသုံးပြုထားပါတယ်
4. မြန်မာဘာသာဖြင့် Error Message များကို ရှင်းလင်းစွာ ဖော်ပြထားပါတယ်

အချုပ်ဆိုရသော် → Java Console Application နဲ့ Database ကို အခြေခံအားဖြင့် ဤနည်းလမ်းဖြင့် ချိတ်ဆက်အသုံးပြုနိုင်ပါတယ်။

💻 Java Console CRUD Application with SQLite3

1. Project Structure

Structure:

```
src/
├── MainApp.java           (Main Application)
├── DatabaseConnection.java  (Database Connection)
├── model/
│   └── Product.java       (Product Model)
└── service/
    └── ProductService.java  (CRUD Operations)
```

2. Database Connection (DatabaseConnection.java)

Code:

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DatabaseConnection {
    private static final String URL = "jdbc:sqlite:products.db";

    public static Connection getConnection() throws SQLException {
        try {
            Class.forName("org.sqlite.JDBC");
            return DriverManager.getConnection(URL);
        } catch (ClassNotFoundException e) {
            throw new SQLException("SQLite JDBC Driver not found");
        }
    }

    public static void initializeDatabase() {
        String sql = "CREATE TABLE IF NOT EXISTS products (" +
                     "id INTEGER PRIMARY KEY AUTOINCREMENT," +
                     "name TEXT NOT NULL," +
                     "price REAL NOT NULL)";

        try (Connection conn = getConnection();
             var stmt = conn.createStatement()) {
            stmt.execute(sql);
        } catch (SQLException e) {
            System.out.println("Database initialization failed: " + e.getMessage());
        }
    }
}
```

3. Product Model (Product.java)

Code:

```
public class Product {
    private int id;
    private String name;
    private double price;

    public Product() {}

    public Product(String name, double price) {
        this.name = name;
        this.price = price;
    }

    // Getters and Setters
    public int getId() { return id; }
    public void setId(int id) { this.id = id; }

    public String getName() { return name; }
    public void setName(String name) { this.name = name; }

    public double getPrice() { return price; }
    public void setPrice(double price) { this.price = price; }

    @Override
    public String toString() {
        return String.format("%d\t%s\t%.2f", id, name, price);
    }
}
```

4. Product Service (ProductService.java)

Code:

```
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class ProductService {
    // Create
    public void addProduct(Product product) {
        String sql = "INSERT INTO products(name, price) VALUES(?, ?)";

        try (Connection conn = DatabaseConnection.getConnection();
             PreparedStatement pstmt = conn.prepareStatement(sql, Statement.RETURN_GENERATED_KEYS))
{

            pstmt.setString(1, product.getName());
            pstmt.setDouble(2, product.getPrice());
            pstmt.executeUpdate();

            try (ResultSet rs = pstmt.getGeneratedKeys()) {
                if (rs.next()) {
                    product.setId(rs.getInt(1));
                }
            }
            System.out.println("ထုတ်ကုန်ထည့်သွင်းပြီးပါပြီ");
        } catch (SQLException e) {
```

```
            System.out.println("ထုတ်ကုန်ထည့်သွင်းရာတွင် အမှားတစ်ခုဖြစ်ပေါ် ခဲ့ပါသည်: " + e.getMessage());
        }
    }

    // Read All
    public List<Product> getAllProducts() {
        List<Product> products = new ArrayList<>();
        String sql = "SELECT * FROM products";

        try (Connection conn = DatabaseConnection.getConnection();
             Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery(sql)) {

            while (rs.next()) {
                Product product = new Product();
                product.setId(rs.getInt("id"));
                product.setName(rs.getString("name"));
                product.setPrice(rs.getDouble("price"));
                products.add(product);
            }
        } catch (SQLException e) {
            System.out.println("ထုတ်ကုန်များရယူရာတွင် အမှားတစ်ခုဖြစ်ပေါ် ခဲ့ပါသည်: " + e.getMessage());
        }
        return products;
    }

    // Update
    public void updateProduct(Product product) {
        String sql = "UPDATE products SET name = ?, price = ? WHERE id = ?";

        try (Connection conn = DatabaseConnection.getConnection();
             PreparedStatement pstmt = conn.prepareStatement(sql)) {

            pstmt.setString(1, product.getName());
            pstmt.setDouble(2, product.getPrice());
            pstmt.setInt(3, product.getId());

            int affectedRows = pstmt.executeUpdate();
            if (affectedRows > 0) {
                System.out.println("ထုတ်ကုန်ပြင်ဆင်ပြီးပါပြီ");
            } else {
                System.out.println("ပြင်ဆင်ရန် ထုတ်ကုန်မတွေ့ပါ");
            }
        } catch (SQLException e) {
            System.out.println("ထုတ်ကုန်ပြင်ဆင်ရာတွင် အမှားတစ်ခုဖြစ်ပေါ် ခဲ့ပါသည်: " + e.getMessage());
        }
    }

    // Delete
    public void deleteProduct(int id) {
        String sql = "DELETE FROM products WHERE id = ?";

        try (Connection conn = DatabaseConnection.getConnection();
             PreparedStatement pstmt = conn.prepareStatement(sql)) {

            pstmt.setInt(1, id);
```

```
            int affectedRows = pstmt.executeUpdate();

            if (affectedRows > 0) {
                System.out.println("ထုတ်ကုန်ဖျက်ပြီးပါပြီ");
            } else {
                System.out.println("ဖျက်ရန် ထုတ်ကုန်မတွေ့ပါ");
            }
        } catch (SQLException e) {
            System.out.println("ထုတ်ကုန်ဖျက်ရာတွင် အမှားတစ်ခုဖြစ်ပေါ် ခဲ့ပါသည်: " + e.getMessage());
        }
    }
}
```

5. Main Application (MainApp.java)

Code:

```
import java.util.List;
import java.util.Scanner;

public class MainApp {
    private static final Scanner scanner = new Scanner(System.in);
    private static final ProductService productService = new ProductService();

    public static void main(String[] args) {
        DatabaseConnection.initializeDatabase();

        while (true) {
            System.out.println("\n=== ထုတ်ကုန်စီမံခန့်ခွဲမှုစနစ် ===");
            System.out.println("1. ထုတ်ကုန်အသစ်ထည့်မယ်");
            System.out.println("2. ထုတ်ကုန်အားလုံးကြည့်မယ်");
            System.out.println("3. ထုတ်ကုန်ပြင်ဆင်မယ်");
            System.out.println("4. ထုတ်ကုန်ဖျက်မယ်");
            System.out.println("5. ထွက်မယ်");
            System.out.print("လုပ်ဆောင်ချက် ရွေးချယ်ပါ (1-5): ");

            int choice = scanner.nextInt();
            scanner.nextLine();  // Clear buffer

            switch (choice) {
                case 1 -> addProduct();
                case 2 -> showAllProducts();
                case 3 -> updateProduct();
                case 4 -> deleteProduct();
                case 5 -> {
                    System.out.println("ကျေးဇူးတင်ပါတယ်!");
                    System.exit(0);
                }
                default -> System.out.println("မှားယွင်းသော ရွေးချယ်မှု");
            }
```

```
        }
    }

    private static void addProduct() {
        System.out.print("ထုတ်ကုန်နာမည်ထည့်ပါ: ");
        String name = scanner.nextLine();
        System.out.print("ထုတ်ကုန်ဈေးနှုန်းထည့်ပါ: ");
        double price = scanner.nextDouble();
        scanner.nextLine();

        Product product = new Product(name, price);
        productService.addProduct(product);
    }

    private static void showAllProducts() {
        List<Product> products = productService.getAllProducts();
        System.out.println("\nID\tName\tPrice");
        System.out.println("----------------------");
        products.forEach(System.out::println);
    }

    private static void updateProduct() {
        showAllProducts();
        System.out.print("\nပြင်ဆင်မည့် ထုတ်ကုန် ID ထည့်ပါ: ");
        int id = scanner.nextInt();
        scanner.nextLine();

        System.out.print("အသစ်နာမည်ထည့်ပါ: ");
        String newName = scanner.nextLine();
        System.out.print("အသစ်ဈေးနှုန်းထည့်ပါ: ");
        double newPrice = scanner.nextDouble();
        scanner.nextLine();

        Product product = new Product(newName, newPrice);
        product.setId(id);
        productService.updateProduct(product);
    }

    private static void deleteProduct() {
        showAllProducts();
        System.out.print("\nဖျက်မည့် ထုတ်ကုန် ID ထည့်ပါ: ");
        int id = scanner.nextInt();
        scanner.nextLine();

        productService.deleteProduct(id);
    }
}
```

အသုံးပြုရန်လိုအပ်သော Dependency

SQLite JDBC Driver ကို pom.xml (Maven) တွင် ထည့်သွင်းရန်:

Code:

```
<dependency>
  <groupId>org.xerial</groupId>
  <artifactId>sqlite-jdbc</artifactId>
  <version>3.42.0.0</version>
</dependency>
```

Run HTML

အရေးကြီးသော အချက်များ

1. SQLite Database File (products.db) ကို Project Root Directory တွင် အလိုအလျောက်ဖန်တီးပေးပါမယ်
2. try-with-resources ကို အသုံးပြုထားပါတယ် (Resource Leak မဖြစ်စေရန်)
3. မြန်မာဘာသာဖြင့် User Interface နဲ့ Error Messages များကို ရှင်းလင်းစွာ ဖော်ပြထားပါတယ်
4. Console မှ Menu-based Interface ဖြင့် အသုံးပြုနိုင်ပါတယ်

ဤ CRUD Application တွင် SQLite Database နဲ့ Java Console Interface ကို ပေါင်းစပ်အသုံးပြုထားပါတယ်။ ထုတ်ကုန်များကို ထည့်ခြင်း၊ ကြည့်ရှုခြင်း၊ ပြင်ဆင်ခြင်း၊ ဖျက်ခြင်း လုပ်ဆောင်နိုင်ပါတယ်။

Mini Projects

1. Yoma Bank ATM Application
2. Yangon Tea House POS Application
3. Bus Ticket Application
4. Mini Chat Application

💻 Yoma Bank ATM Application

Application Features:

1. **BankAccount Interface** - deposit, withdraw, getBalance method တွေပါဝင်ပါတယ်။
2. **Lambda Expressions အသုံးပြုမှု**:
   - deposit နှင့် withdraw operation တွေအတွက် Consumer functional interface ကိုသုံးထားပါတယ်။
   - checkBalance operation အတွက် Function functional interface ကိုသုံးထားပါတယ်။
3. **Menu System**:
   - ငွေထုတ်မယ် (Withdrawal)
   - ငွေထည့်မယ် (Deposit)
   - လက်ရှိအကောင့်ငွေစစ်မယ် (Checking Account Balance)
   - ချွေတာငွေအကောင့်စစ်မယ် (Savings Account Balance)
   - ထွက်မယ် (Exit)
4. **အကောင့်နှစ်မျိုး**:
   - လက်ရှိအကောင့် (Checking Account) - အစပိုင်းတွင် 100,000 ကျပ်ဖြင့်စတင်ထားပါတယ်။
   - ချွေတာငွေအကောင့် (Savings Account) - အစပိုင်းတွင် 500,000 ကျပ်ဖြင့်စတင်ထားပါတယ်။
5. **အကာအကွယ်**:
   - လက်ကျန်ငွေထက်ပိုပြီးငွေထုတ်ယူလျှင် "လက်ကျန်ငွေ မလုံလောက်ပါ" ဟုသတိပေးပါမယ်။
   - မှားယွင်းသောရွေးချယ်မှုများအတွက် error message ပြသပါမယ်။

YomaBankATM.java

Code:

```
import java.util.Scanner;
import java.util.function.Consumer;
import java.util.function.Function;

// Bank Interface with required methods
interface BankAccount {
   void deposit(double amount);
   void withdraw(double amount);
   double getBalance();
}

public class YomaBankATM {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);

      // Create accounts using lambda expressions
      BankAccount checkingAccount = new BankAccount() {
         private double balance = 100000; // Initial balance

         @Override
         public void deposit(double amount) {
            balance += amount;
         }

         @Override
         public void withdraw(double amount) {
            if(amount <= balance) {
               balance -= amount;
            } else {
               System.out.println("လက်ကျန်ငွေ မလုံလောက်ပါ။");
            }
         }

         @Override
         public double getBalance() {
            return balance;
         }
      };

      BankAccount savingsAccount = new BankAccount() {
         private double balance = 500000; // Initial balance

         @Override
         public void deposit(double amount) {
            balance += amount;
         }

         @Override
         public void withdraw(double amount) {
            if(amount <= balance) {
               balance -= amount;
            } else {
```

```
            System.out.println("လက်ကျန်ငွေ မလုံလောက်ပါ။");
        }
    }

    @Override
    public double getBalance() {
        return balance;
    }
};

// Lambda expressions for common operations
Consumer<Double> deposit = amount -> {
    System.out.println(amount + " ကျပ် ငွေထည့်ပြီးပါပြီ။");
};

Consumer<Double> withdraw = amount -> {
    System.out.println(amount + " ကျပ် ငွေထုတ်ပြီးပါပြီ။");
};

Function<BankAccount, String> checkBalance = account -> {
    return "လက်ကျန်ငွေ: " + account.getBalance() + " ကျပ်";
};

boolean running = true;

while(running) {
    System.out.println("\n===== Yoma Bank ATM Menu =====");
    System.out.println("1. ငွေထုတ်မယ်");
    System.out.println("2. ငွေထည့်မယ်");
    System.out.println("3. လက်ရှိအကောင့်ငွေစစ်မယ်");
    System.out.println("4. ချွေတာငွေအကောင့်စစ်မယ်");
    System.out.println("5. ထွက်မယ်");
    System.out.print("ရွေးချယ်မှုကိုရိုက်ထည့်ပါ: ");

    int choice = scanner.nextInt();
    double amount;

    switch(choice) {
        case 1: // Withdrawal
            System.out.print("ထုတ်ယူမည့်ငွေပမာဏကိုရိုက်ထည့်ပါ: ");
            amount = scanner.nextDouble();

            System.out.println("1. လက်ရှိအကောင့်");
            System.out.println("2. ချွေတာငွေအကောင့်");
            System.out.print("ရွေးချယ်မှုကိုရိုက်ထည့်ပါ: ");
            int accType = scanner.nextInt();

            if(accType == 1) {
                checkingAccount.withdraw(amount);
                withdraw.accept(amount);
            } else if(accType == 2) {
```

```
                savingsAccount.withdraw(amount);
                withdraw.accept(amount);
            } else {
                System.out.println("မှားယွင်းသောရွေးချယ်မှုဖြစ်ပါတယ်။");
            }
            break;

        case 2: // Deposit
            System.out.print("ထည့်သွင်းမည့်ငွေပမာဏကိုရိုက်ထည့်ပါ: ");
            amount = scanner.nextDouble();

            System.out.println("1. လက်ရှိအကောင့်");
            System.out.println("2. ချွေတာငွေအကောင့်");
            System.out.print("ရွေးချယ်မှုကိုရိုက်ထည့်ပါ: ");
            accType = scanner.nextInt();

            if(accType == 1) {
                checkingAccount.deposit(amount);
                deposit.accept(amount);
            } else if(accType == 2) {
                savingsAccount.deposit(amount);
                deposit.accept(amount);
            } else {
                System.out.println("မှားယွင်းသောရွေးချယ်မှုဖြစ်ပါတယ်။");
            }
            break;

        case 3: // Check Checking Account Balance
            System.out.println(checkBalance.apply(checkingAccount));
            break;

        case 4: // Check Savings Account Balance
            System.out.println(checkBalance.apply(savingsAccount));
            break;

        case 5: // Exit
            running = false;
            System.out.println("Yoma Bank ATM ကိုအသုံးပြုတဲ့အတွက် ကျေးဇူးတင်ပါတယ်။");
            break;

        default:
            System.out.println("မှားယွင်းသောရွေးချယ်မှုဖြစ်ပါတယ်။ ကျေးဇူးပြု၍ ပြန်ရွေးချယ်ပါ။");
      }
    }

    scanner.close();
  }
}
```

💻 Yangon Tea House Restaurant POS System

Program Output Results

Test Case 1: Add New Menu Items

Console:

```
===== Yangon Tea House POS System =====
1. အစားအစာများကြည့်မယ်
2. အစားအစာအသစ်ထည့်မယ်
3. အစားအစာပြင်ဆင်မယ်
4. အစားအစာဖျက်မယ်
5. အမှာစာလုပ်မယ်
6. ထွက်မယ်
ရွေးချယ်မှုကိုရိုက်ထည့်ပါ: 2
အစားအစာအမည်: လက်ဖက်ရည်ကြမ်း
ဈေးနှုန်း: 500
အရေအတွက်: 100
လက်ဖက်ရည်ကြမ်း ကိုမီနူးထဲသို့ထည့်ပြီးပါပြီ။
```

Test Case 2: Place Order

Console:

```
===== Yangon Tea House POS System =====
1. အစားအစာများကြည့်မယ်
2. အစားအစာအသစ်ထည့်မယ်
3. အစားအစာပြင်ဆင်မယ်
4. အစားအစာဖျက်မယ်
5. အမှာစာလုပ်မယ်
6. ထွက်မယ်
ရွေးချယ်မှုကိုရိုက်ထည့်ပါ: 5

--- Menu Items ---
1. လက်ဖက်ရည်ကြမ်း - 500 ကျပ် (လက်ကျန်: 100)
မှာယူမည့်အစားအစာအမှတ်စဉ် (0 နှိပ်ပါက အမှာစာပြီးမည်): 1
```

```
အရေအတွက်: 2
မှာယူမည့်အစားအစာအမှတ်စဉ် (0 နှိပ်ပါက အမှာစာပြီးမည်): 0

--- အမှာစာအချက်အလက် ---
အစားအစာ ပြင်ဆင်နေပါတယ် - လက်ဖက်ရည်ကြမ်း
လက်ဖက်ရည်ကြမ်း x 2 = 1,000 ကျပ်
စုစုပေါင်း: 1,000 ကျပ်
ကျေးဇူးတင်ပါတယ်!
```

Test Case 3: Check Database File

Console:

```
1,လက်ဖက်ရည်ကြမ်း,500.0,98
```

Test Case 4: Update Menu Item

Console:

```
===== Yangon Tea House POS System =====
1. အစားအစာများကြည့်မယ်
2. အစားအစာအသစ်ထည့်မယ်
3. အစားအစာပြင်ဆင်မယ်
4. အစားအစာဖျက်မယ်
5. အမှာစာလုပ်မယ်
6. ထွက်မယ်
ရွေးချယ်မှုကိုရိုက်ထည့်ပါ: 3

--- Menu Items ---
1. လက်ဖက်ရည်ကြမ်း - 500 ကျပ် (လက်ကျန်: 98)
ပြင်ဆင်မည့်အစားအစာအမှတ်စဉ်: 1
အမည်အသစ် (လက်ဖက်ရည်ကြမ်း):
ဈေးနှုန်းအသစ် (500.0): 600
အရေအတွက်အသစ် (98):
အစားအစာကိုပြင်ဆင်ပြီးပါပြီ။
```

Test Case 5: View Updated Menu

Console:

```
===== Yangon Tea House POS System =====
1. အစားအစာများကြည့်မယ်
2. အစားအစာအသစ်ထည့်မယ်
3. အစားအစာပြင်ဆင်မယ်
```

```
4. အစားအစာဖျက်မယ်
5. အမှာစာလုပ်မယ်
6. ထွက်မယ်
ရွေးချယ်မှုကိုရိုက်ထည့်ပါ: 1

--- Menu Items ---
1. လက်ဖက်ရည်ကြမ်း - 600 ကျပ် (လက်ကျန်: 98)
```

System Features :

- ✓ **CRUD Operations**:
  - အစားအစာအသစ်များထည့်သွင်းနိုင်ခြင်း
  - အစားအစာများကိုဖတ်ရှုနိုင်ခြင်း
  - အစားအစာများကိုပြင်ဆင်နိုင်ခြင်း
  - အစားအစာများကိုဖျက်နိုင်ခြင်း
- ✓ **Order Processing**:
  - အမှာစာလုပ်နိုင်ခြင်း
  - လက်ကျန်ပမာဏကိုအလိုအလျောက်ပြင်ဆင်ပေးခြင်း
  - ငွေတောင်းခံလွှာထုတ်ပေးခြင်း
- ✓ **Data Persistence**:
  - database.txt ဖိုင်တွင် အချက်အလက်များကိုသိမ်းဆည်းခြင်း
  - Program ပြန်ဖွင့်တိုင်း အရင်အချက်အလက်များကိုပြန်လည်ရယူခြင်း
- ✓ **Error Handling**:
  - မရှိသောအစားအစာများအတွက်စစ်ဆေးခြင်း
  - လက်ကျန်မလုံလောက်မှုကိုစစ်ဆေးခြင်း
  - ဖိုင်အမှားများကိုကိုင်တွယ်ခြင်း

Project Structure

Structure:

```
yangon-tea-house-pos/
├── src/
│   ├── model/
│   │   ├── Item.java
│   │   └── MenuItem.java
│   ├── service/
│   │   ├── CRUDService.java
│   │   └── CRUDImplementation.java
│   └── MainApp.java
├── database.txt
└── README.md
```

1. Model Classes and Interfaces

MenuItem Interface

Code:

```
public interface MenuItem {
   String getName();
   double getPrice();
   void prepare();
}
```

Item Class

Code:

```
public class Item implements MenuItem {
   private int id;
   private String name;
   private double price;
   private int quantity;

   public Item(int id, String name, double price, int quantity) {
      this.id = id;
      this.name = name;
      this.price = price;
      this.quantity = quantity;
   }

   // Getters and Setters
   @Override
   public String getName() { return name; }

   @Override
   public double getPrice() { return price; }
```

```
    @Override
    public void prepare() {
        System.out.println(name + " ပြင်ဆင်နေပါတယ်...");
    }

    public int getId() { return id; }

    public int getQuantity() { return quantity; }

    public void setQuantity(int quantity) { this.quantity = quantity; }

    @Override
    public String toString() {
        return id + "," + name + "," + price + "," + quantity;
    }
}
```

CRUDService Interface

Code:

```
import java.util.List;

public interface CRUDService {
    void create(Item item);
    Item read(int id);
    void update(Item item);
    void delete(int id);
    List<Item> getAllItems();
    void saveToFile();
    void loadFromFile();
}
```

2. CRUD Implementation

CRUDImplementation Class

Code:

```
import java.io.*;
import java.util.ArrayList;
import java.util.List;

public class CRUDImplementation implements CRUDService {
    private static final String FILE_NAME = "database.txt";
    private List<Item> items = new ArrayList<>();

    public CRUDImplementation() {
        loadFromFile();
    }

    @Override
    public void create(Item item) {
        items.add(item);
        saveToFile();
    }
```

```
    @Override
    public Item read(int id) {
        return items.stream()
                .filter(item -> item.getId() == id)
                .findFirst()
                .orElse(null);
    }

    @Override
    public void update(Item updatedItem) {
        items.replaceAll(item ->
            item.getId() == updatedItem.getId() ? updatedItem : item);
        saveToFile();
    }

    @Override
    public void delete(int id) {
        items.removeIf(item -> item.getId() == id);
        saveToFile();
    }

    @Override
    public List<Item> getAllItems() {
        return new ArrayList<>(items);
    }

    @Override
    public void saveToFile() {
        try (PrintWriter writer = new PrintWriter(new FileWriter(FILE_NAME))) {
            items.forEach(item -> writer.println(item.toString()));
        } catch (IOException e) {
            System.err.println("ဖိုင်သိမ်းရာတွင် အမှားတစ်ခုဖြစ်ပေါ် ခဲ့ပါတယ်: " + e.getMessage());
        }
    }

    @Override
    public void loadFromFile() {
        items.clear();
        try (BufferedReader reader = new BufferedReader(new FileReader(FILE_NAME))) {
            String line;
            while ((line = reader.readLine()) != null) {
                String[] parts = line.split(",");
                if (parts.length == 4) {
                    int id = Integer.parseInt(parts[0]);
                    String name = parts[1];
                    double price = Double.parseDouble(parts[2]);
                    int quantity = Integer.parseInt(parts[3]);
                    items.add(new Item(id, name, price, quantity));
                }
            }
        } catch (IOException e) {
            System.err.println("ဖိုင်ဖတ်ရာတွင် အမှားတစ်ခုဖြစ်ပေါ် ခဲ့ပါတယ်: " + e.getMessage());
        }
    }
}
```

3. Main Application

MainApp Class

Code:

```
import java.util.*;
import java.util.function.*;

public class MainApp {
    private static final Scanner scanner = new Scanner(System.in);
    private static final CRUDService crudService = new CRUDImplementation();

    public static void main(String[] args) {
        boolean running = true;

        // Lambda expressions for common operations
        Consumer<String> printHeader = title ->
            System.out.println("\n===== " + title + " =====");

        BiConsumer<MenuItem, Integer> prepareItem = (item, quantity) -> {
            System.out.println(quantity + " x " + item.getName() + " ပြင်ဆင်နေပါတယ်...");
            item.prepare();
        };

        while(running) {
            printHeader.accept("Yangon Tea House POS System");
            System.out.println("1. အစားအစာများကြည့်မယ်");
            System.out.println("2. အစားအစာအသစ်ထည့်မယ်");
            System.out.println("3. အစားအစာပြင်ဆင်မယ်");
            System.out.println("4. အစားအစာဖျက်မယ်");
            System.out.println("5. အမှာစာလုပ်မယ်");
            System.out.println("6. ထွက်မယ်");
            System.out.print("ရွေးချယ်မှုကိုရိုက်ထည့်ပါ: ");

            int choice = scanner.nextInt();
            scanner.nextLine(); // Consume newline

            switch(choice) {
                case 1: // View Menu Items
                    displayMenuItems();
                    break;

                case 2: // Add New Menu Item
                    addNewMenuItem();
                    break;

                case 3: // Update Menu Item
                    updateMenuItem();
```

```
            break;

        case 4: // Delete Menu Item
            deleteMenuItem();
            break;

        case 5: // Place Order
            placeOrder(prepareItem);
            break;

        case 6: // Exit
            running = false;
            System.out.println("POS System ကိုပိတ်လိုက်ပါတယ်။");
            break;

        default:
            System.out.println("မှားယွင်းသောရွေးချယ်မှုဖြစ်ပါတယ်။");
      }
    }
}

private static void displayMenuItems() {
    System.out.println("\n--- Menu Items ---");
    crudService.getAllItems().forEach(item ->
        System.out.printf("%d. %s - %,.0f ကျပ် (လက်ကျန်: %d)%n",
            item.getId(), item.getName(), item.getPrice(), item.getQuantity()));
}

private static void addNewMenuItem() {
    System.out.print("အစားအစာအမည်: ");
    String name = scanner.nextLine();

    System.out.print("ဈေးနှုန်း: ");
    double price = scanner.nextDouble();

    System.out.print("အရေအတွက်: ");
    int quantity = scanner.nextInt();

    int newId = crudService.getAllItems().stream()
            .mapToInt(Item::getId)
            .max()
            .orElse(0) + 1;

    crudService.create(new Item(newId, name, price, quantity));
    System.out.println(name + " ကိုမီနူးထဲသို့ထည့်ပြီးပါပြီ။");
}

private static void updateMenuItem() {
    displayMenuItems();
    System.out.print("ပြင်ဆင်မည့်အစားအစာအမှတ်စဉ်: ");
    int id = scanner.nextInt();
    scanner.nextLine();

    Item item = crudService.read(id);
```

```
        if (item == null) {
            System.out.println("အစားအစာမတွေ့ပါ။");
            return;
        }

        System.out.print("အမည်အသစ် (" + item.getName() + "): ");
        String name = scanner.nextLine();
        if (!name.isEmpty()) item = new Item(item.getId(), name, item.getPrice(), item.getQuantity());

        System.out.print("ဈေးနှုန်းအသစ် (" + item.getPrice() + "): ");
        String priceInput = scanner.nextLine();
        if (!priceInput.isEmpty()) {
            double price = Double.parseDouble(priceInput);
            item = new Item(item.getId(), item.getName(), price, item.getQuantity());
        }

        System.out.print("အရေအတွက်အသစ် (" + item.getQuantity() + "): ");
        String quantityInput = scanner.nextLine();
        if (!quantityInput.isEmpty()) {
            int quantity = Integer.parseInt(quantityInput);
            item = new Item(item.getId(), item.getName(), item.getPrice(), quantity);
        }

        crudService.update(item);
        System.out.println("အစားအစာကိုပြင်ဆင်ပြီးပါပြီ။");
    }

    private static void deleteMenuItem() {
        displayMenuItems();
        System.out.print("ဖျက်မည့်အစားအစာအမှတ်စဉ်: ");
        int id = scanner.nextInt();

        Item item = crudService.read(id);
        if (item != null) {
            crudService.delete(id);
            System.out.println(item.getName() + " ကိုဖျက်ပြီးပါပြီ။");
        } else {
            System.out.println("အစားအစာမတွေ့ပါ။");
        }
    }

    private static void placeOrder(BiConsumer<MenuItem, Integer> prepareItem) {
        Map<Item, Integer> order = new HashMap<>();
        boolean ordering = true;

        while (ordering) {
            displayMenuItems();
            System.out.print("မှာယူမည့်အစားအစာအမှတ်စဉ် (0 နှိပ်ပါက အမှာစာပြီးမည်): ");
            int id = scanner.nextInt();

            if (id == 0) {
                ordering = false;
            } else {
                Item item = crudService.read(id);
```

```
            if (item != null) {
                System.out.print("အရေအတွက်: ");
                int quantity = scanner.nextInt();

                if (quantity <= item.getQuantity()) {
                    order.merge(item, quantity, Integer::sum);
                    item.setQuantity(item.getQuantity() - quantity);
                    crudService.update(item);
                } else {
                    System.out.println("လက်ကျန်မလုံလောက်ပါ။");
                }
            } else {
                System.out.println("အစားအစာမတွေ့ပါ။");
            }
        }
    }

    if (!order.isEmpty()) {
        System.out.println("\n--- အမှာစာအချက်အလက် ---");
        double total = 0;

        for (Map.Entry<Item, Integer> entry : order.entrySet()) {
            Item item = entry.getKey();
            int quantity = entry.getValue();
            double itemTotal = item.getPrice() * quantity;

            prepareItem.accept(item, quantity);
            System.out.printf("%s x %d = %,.0f ကျပ်%n",
                item.getName( ), quantity, itemTotal);

            total += itemTotal;
        }

        System.out.printf("စုစုပေါင်း: %,.0f ကျပ်%n", total);
        System.out.println("ကျေးဇူးတင်ပါတယ်!");
    }
  }
}
```

💻 Bus Ticket Application

System Features :

- ✓ **CRUD Operations**:
  - လက်မှတ်(Ticket) အသစ်များထည့်သွင်းနိုင်ခြင်း
  - Ticket Reservation Record များကိုဖတ်ရှုနိုင်ခြင်း
  - Ticket Reservation Record များကိုပြင်ဆင်နိုင်ခြင်း
  - Ticket Reservation Record များကိုဖျက်နိုင်ခြင်း
- ✓ **Data Persistence**:
  - Bustickets.db ဖိုင်တွင် အချက်အလက်များကိုသိမ်းဆည်းခြင်း
  - Program ပြန်ဖွင့်တိုင်း အရင်အချက်အလက်များကို database မှပြန်လည်ရယူခြင်း
- ✓ **Error Handling**:
  - Database Connection ရမရ စစ်ဆေးခြင်း
  - Reservation Edit လုပ်တဲ့အခါ empty input တွေကို စစ်ဆေးခြင်း

Project Structure

Tree:

```
BusTicketApp/
├─── src/
│    ├─── model/
│    │    └─── Ticket.java
│    ├─── services/
│    │    ├─── BookingService.java
│    │    └─── IBookingService.java
│    └─── MainApp.java
├─── database/
│    └─── bustickets.db
└─── lib/
     └─── sqlite-jdbc-3.36.0.3.jar
```

Implementation

1. Ticket Model (src/model/Ticket.java)

Code:

```
package model;

public class Ticket {
    private int id;
    private int seatNumber;
    private String passengerName;
    private String source;
    private String destination;
    private double fee;

    public Ticket(int id, int seatNumber, String passengerName, String sour
ce, String destination, double fee) {
        this.id = id;
        this.seatNumber = seatNumber;
        this.passengerName = passengerName;
        this.source = source;
        this.destination = destination;
        this.fee = fee;
    }

    // Getters
    public int getId() { return id; }
    public int getSeatNumber() { return seatNumber; }
```

```
    public String getPassengerName() { return passengerName; }
    public String getSource() { return source; }
    public String getDestination() { return destination; }
    public double getFee() { return fee; }

    // Setters
    public void setSeatNumber(int seatNumber) { this.seatNumber = seatNumbe
r; }
    public void setPassengerName(String passengerName) { this.passengerName
= passengerName; }
    public void setSource(String source) { this.source = source; }
    public void setDestination(String destination) { this.destination = des
tination; }
    public void setFee(double fee) { this.fee = fee; }

    @Override
    public String toString() {
        return String.format("Ticket ID: %d\nSeat Number: %d\nPassenger: %s
\nRoute: %s to %s\nFee: $%.2f",
                id, seatNumber, passengerName, source, destination, fee);
    }
}
```

2. Booking Service Interface (src/services/IBookingService.java)

Code:

```
package services;

import model.Ticket;
import java.util.List;

public interface IBookingService {
    void bookTicket(Ticket ticket);
    List<Ticket> getAllTickets();
    Ticket getTicketById(int id);
    boolean updateTicket(Ticket ticket);
    boolean deleteTicket(int id);
    void printTicket(int ticketId);
}
```

3. Booking Service Implementation (src/services/BookingService.java)

Code:

```
package services;
```

```
import model.Ticket;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class BookingService implements IBookingService {
    private static final String DB_URL = "jdbc:sqlite:database/bustickets.d
b";

    public BookingService() {
        initializeDatabase();
    }

    private void initializeDatabase() {
        try (Connection conn = DriverManager.getConnection(DB_URL);
             Statement stmt = conn.createStatement()) {

            String sql = "CREATE TABLE IF NOT EXISTS tickets (" +
                         "id INTEGER PRIMARY KEY AUTOINCREMENT," +
                         "seat_number INTEGER NOT NULL UNIQUE," +
                         "passenger_name TEXT NOT NULL," +
                         "source TEXT NOT NULL," +
                         "destination TEXT NOT NULL," +
                         "fee REAL NOT NULL)";
            stmt.execute(sql);

        } catch (SQLException e) {
            System.out.println("Database initialization error: " + e.getMes
sage());
        }
    }

    @Override
    public void bookTicket(Ticket ticket) {
        try (Connection conn = DriverManager.getConnection(DB_URL);
             PreparedStatement pstmt = conn.prepareStatement(
                     "INSERT INTO tickets (seat_number, passenger_name, sou
rce, destination, fee) VALUES (?, ?, ?, ?, ?)")) {

            pstmt.setInt(1, ticket.getSeatNumber());
            pstmt.setString(2, ticket.getPassengerName());
            pstmt.setString(3, ticket.getSource());
            pstmt.setString(4, ticket.getDestination());
            pstmt.setDouble(5, ticket.getFee());

            pstmt.executeUpdate();
```

```
            System.out.println("Ticket booked successfully!");

        } catch (SQLException e) {
            System.out.println("Error booking ticket: " + e.getMessage());
        }
    }

    @Override
    public List<Ticket> getAllTickets() {
        List<Ticket> tickets = new ArrayList<>();

        try (Connection conn = DriverManager.getConnection(DB_URL);
             Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery("SELECT * FROM tickets")) {

            while (rs.next()) {
                Ticket ticket = new Ticket(
                    rs.getInt("id"),
                    rs.getInt("seat_number"),
                    rs.getString("passenger_name"),
                    rs.getString("source"),
                    rs.getString("destination"),
                    rs.getDouble("fee")
                );
                tickets.add(ticket);
            }

        } catch (SQLException e) {
            System.out.println("Error retrieving tickets: " + e.getMessage(
));
        }

        return tickets;
    }

    @Override
    public Ticket getTicketById(int id) {
        try (Connection conn = DriverManager.getConnection(DB_URL);
             PreparedStatement pstmt = conn.prepareStatement(
                     "SELECT * FROM tickets WHERE id = ?")) {

            pstmt.setInt(1, id);
            ResultSet rs = pstmt.executeQuery();

            if (rs.next()) {
                return new Ticket(
```

```
                    rs.getInt("id"),
                    rs.getInt("seat_number"),
                    rs.getString("passenger_name"),
                    rs.getString("source"),
                    rs.getString("destination"),
                    rs.getDouble("fee")
                );
            }

        } catch (SQLException e) {
            System.out.println("Error retrieving ticket: " + e.getMessage()
);
        }

        return null;
    }

    @Override
    public boolean updateTicket(Ticket ticket) {
        try (Connection conn = DriverManager.getConnection(DB_URL);
             PreparedStatement pstmt = conn.prepareStatement(
                     "UPDATE tickets SET seat_number = ?, passenger_name =
?, source = ?, destination = ?, fee = ? WHERE id = ?")) {

            pstmt.setInt(1, ticket.getSeatNumber());
            pstmt.setString(2, ticket.getPassengerName());
            pstmt.setString(3, ticket.getSource());
            pstmt.setString(4, ticket.getDestination());
            pstmt.setDouble(5, ticket.getFee());
            pstmt.setInt(6, ticket.getId());

            int rowsAffected = pstmt.executeUpdate();
            return rowsAffected > 0;

        } catch (SQLException e) {
            System.out.println("Error updating ticket: " + e.getMessage());
            return false;
        }
    }

    @Override
    public boolean deleteTicket(int id) {
        try (Connection conn = DriverManager.getConnection(DB_URL);
             PreparedStatement pstmt = conn.prepareStatement(
                     "DELETE FROM tickets WHERE id = ?")) {
```

```
            pstmt.setInt(1, id);
            int rowsAffected = pstmt.executeUpdate();
            return rowsAffected > 0;

        } catch (SQLException e) {
            System.out.println("Error deleting ticket: " + e.getMessage());
            return false;
        }
    }

    @Override
    public void printTicket(int ticketId) {
        Ticket ticket = getTicketById(ticketId);
        if (ticket != null) {
            System.out.println("\n=== BUS TICKET ===");
            System.out.println(ticket);
            System.out.println("==================");
        } else {
            System.out.println("Ticket with ID " + ticketId + " not found."
);
        }
    }
}
```

4. Main Application (src/MainApp.java)

Code:

```
import services.BookingService;
import services.IBookingService;
import model.Ticket;
import java.util.List;
import java.util.Scanner;

public class MainApp {
    public static void main(String[] args) {
        IBookingService bookingService = new BookingService();
        Scanner sc = new Scanner(System.in);
        boolean running = true;

        while (running) {
            System.out.println("\n=== Bus Ticket Reservation System ===");
            System.out.println("1. Book a Seat");
            System.out.println("2. View Reservations");
            System.out.println("3. Edit Reservations");
            System.out.println("4. Print a Ticket");
            System.out.println("------------------");
```

```
            System.out.println("5. Exit Program");
            System.out.println("------------------");
            System.out.print("Enter your choice: ");

            int choice = scanner.nextInt();
            sc.nextLine(); // Consume newline

            switch (choice) {
                case 1:
                    bookSeat(bookingService, sc);
                    break;
                case 2:
                    viewReservations(bookingService);
                    break;
                case 3:
                    editReservation(bookingService, sc);
                    break;
                case 4:
                    printTicket(bookingService, sc);
                    break;
                case 5:
                    running = false;
                    System.out.println("Exiting program. Thank you!");
                    break;
                default:
                    System.out.println("Invalid choice. Please try again.")
;
            }
        }

        sc.close();
    }

    private static void bookSeat(IBookingService service, Scanner sc) {
        System.out.println("\n--- Book a Seat ---");

        System.out.print("Enter seat number: ");
        int seatNumber = sc.nextInt();
        sc.nextLine(); // Consume newline

        System.out.print("Enter passenger name: ");
        String passengerName = sc.nextLine();

        System.out.print("Enter source: ");
        String source = sc.nextLine();
```

```
        System.out.print("Enter destination: ");
        String destination = sc.nextLine();

        System.out.print("Enter fee: ");
        double fee = sc.nextDouble();
        sc.nextLine(); // Consume newline

        Ticket ticket = new Ticket(0, seatNumber, passengerName, source, de
stination, fee);
        service.bookTicket(ticket);
    }

    private static void viewReservations(IBookingService service) {
        System.out.println("\n--- All Reservations ---");
        List<Ticket> tickets = service.getAllTickets();

        if (tickets.isEmpty()) {
            System.out.println("No reservations found.");
        } else {
            for (Ticket ticket : tickets) {
                System.out.println(ticket);
                System.out.println("----------------------");
            }
        }
    }

    private static void editReservation(IBookingService service, Scanner sc
) {
        System.out.println("\n--- Edit Reservation ---");
        System.out.print("Enter ticket ID to edit: ");
        int id = sc.nextInt();
        sc.nextLine(); // Consume newline

        Ticket ticket = service.getTicketById(id);
        if (ticket == null) {
            System.out.println("Ticket with ID " + id + " not found.");
            return;
        }

        System.out.println("\nCurrent details:");
        System.out.println(ticket);

        System.out.println("\nEnter new details (leave blank to keep curren
t value):");

        System.out.print("Seat number (" + ticket.getSeatNumber() + "): ");
```

```
        String seatInput = sc.nextLine();
        if (!seatInput.isEmpty()) {
            ticket.setSeatNumber(Integer.parseInt(seatInput));
        }

        System.out.print("Passenger name (" + ticket.getPassengerName() + "
): ");
        String passengerName = sc.nextLine();
        if (!passengerName.isEmpty()) {
            ticket.setPassengerName(passengerName);
        }

        System.out.print("Source (" + ticket.getSource() + "): ");
        String source = sc.nextLine();
        if (!source.isEmpty()) {
            ticket.setSource(source);
        }

        System.out.print("Destination (" + ticket.getDestination() + "): ")
;
        String destination = sc.nextLine();
        if (!destination.isEmpty()) {
            ticket.setDestination(destination);
        }

        System.out.print("Fee (" + ticket.getFee() + "): ");
        String feeInput = sc.nextLine();
        if (!feeInput.isEmpty()) {
            ticket.setFee(Double.parseDouble(feeInput));
        }

        boolean updated = service.updateTicket(ticket);
        if (updated) {
            System.out.println("Reservation updated successfully!");
        } else {
            System.out.println("Failed to update reservation.");
        }
    }

    private static void printTicket(IBookingService service, Scanner sc) {
        System.out.println("\n--- Print Ticket ---");
        System.out.print("Enter ticket ID to print: ");
        int ticketId = sc.nextInt();
        sc.nextLine(); // Consume newline

        service.printTicket(ticketId);
```

```
    }
}
```

Testing Results

Sample Run 1: Booking a Seat

Console:

```
=== Bus Ticket Reservation System ===
1. Book a Seat
2. View Reservations
3. Edit Reservations
4. Print a Ticket
5. Exit Program
Enter your choice: 1

--- Book a Seat ---
Enter seat number: 12
Enter passenger name: John Doe
Enter source: Yangon
Enter destination: Mandalay
Enter fee: 15000
Ticket booked successfully!
```

Sample Run 2: Viewing Reservations

Console:

```
=== Bus Ticket Reservation System ===
1. Book a Seat
2. View Reservations
3. Edit Reservations
4. Print a Ticket
5. Exit Program
Enter your choice: 2

--- All Reservations ---
Ticket ID: 1
Seat Number: 12
Passenger: John Doe
Route: Yangon to Mandalay
Fee: $15000.00
---------------------
```

Sample Run 3: Editing a Reservation

Console:

```
=== Bus Ticket Reservation System ===
1. Book a Seat
2. View Reservations
3. Edit Reservations
4. Print a Ticket
5. Exit Program
Enter your choice: 3

--- Edit Reservation ---
Enter ticket ID to edit: 1

Current details:
Ticket ID: 1
Seat Number: 12
Passenger: John Doe
Route: Yangon to Mandalay
Fee: $15000.00

Enter new details (leave blank to keep current value):
Seat number (12): 14
Passenger name (John Doe): John William Doe
Source (Yangon):
Destination (Mandalay):
Fee (15000.0): 16000
Reservation updated successfully!
```

Sample Run 4: Printing a Ticket

Console:

```
=== Bus Ticket Reservation System ===
1. Book a Seat
2. View Reservations
3. Edit Reservations
4. Print a Ticket
5. Exit Program
Enter your choice: 4

--- Print Ticket ---
Enter ticket ID to print: 1

=== BUS TICKET ===
Ticket ID: 1
```

```
Seat Number: 14
Passenger: John William Doe
Route: Yangon to Mandalay
Fee: $16000.00
=================
```

💻 Mini Chat Application

Console Chat Application

Java Console အသုံးပြုပြီး Chat Application တစ်ခုရေးနည်းကို ရှင်းပြပေးပါမယ်။

အခြေခံအလုပ်လုပ်ပုံ

1. **Client-Server Model** အသုံးပြုပါမယ်
2. **Socket Programming** နည်းပညာကို အခြေခံပါမယ်
3. Multi-threading သုံးပြီး multiple clients တစ်ပြိုင်နက်ချိတ်ဆက်နိုင်ပါမယ်

Project Structure

Tree:

```
JavaChatApp/
├── src/
│   ├── server/
│   │   ├── ChatServer.java
│   │   └── ClientHandler.java
│   └── client/
│       └── ChatClient.java
└── README.md
```

1. Server Side Implementation

ChatServer.java

Code:

```java
package server;

import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.ArrayList;
import java.util.List;

public class ChatServer {
    private static final int PORT = 12345;
```

```
    private static List<ClientHandler> clients = new ArrayList<>();

    public static void main(String[] args) {
        try (ServerSocket serverSocket = new ServerSocket(PORT)) {
            System.out.println("Chat Server is running on port " + PORT);

            while (true) {
                Socket clientSocket = serverSocket.accept();
                System.out.println("New client connected");

                ClientHandler clientThread = new ClientHandler(clientSocket
, clients);
                clients.add(clientThread);
                new Thread(clientThread).start();
            }
        } catch (IOException e) {
            System.out.println("Server exception: " + e.getMessage());
        }
    }
}
```

ClientHandler.java

Code:

```
package server;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;
import java.util.List;

public class ClientHandler implements Runnable {
    private Socket clientSocket;
    private List<ClientHandler> clients;
    private PrintWriter out;
    private String clientName;

    public ClientHandler(Socket socket, List<ClientHandler> clients) {
        this.clientSocket = socket;
        this.clients = clients;
    }

    @Override
```

```
    public void run() {
        try {
            out = new PrintWriter(clientSocket.getOutputStream(), true);
            BufferedReader in = new BufferedReader(new InputStreamReader(cl
ientSocket.getInputStream()));

            // Get client name
            out.println("Enter your name:");
            clientName = in.readLine();
            broadcast(clientName + " has joined the chat!");

            String inputLine;
            while ((inputLine = in.readLine()) != null) {
                if ("/quit".equalsIgnoreCase(inputLine)) {
                    break;
                }
                broadcast(clientName + ": " + inputLine);
            }
        } catch (IOException e) {
            System.out.println("Error in ClientHandler: " + e.getMessage())
;
        } finally {
            try {
                clientSocket.close();
            } catch (IOException e) {
                System.out.println("Error closing socket: " + e.getMessage(
));
            }
            clients.remove(this);
            broadcast(clientName + " has left the chat.");
        }
    }

    private void broadcast(String message) {
        for (ClientHandler client : clients) {
            client.out.println(message);
        }
    }
}
```

2. Client Side Implementation

ChatClient.java

Code:

```
package client;
```

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;

public class ChatClient {
    private static final String SERVER_ADDRESS = "localhost";
    private static final int SERVER_PORT = 12345;

    public static void main(String[] args) {
        try (Socket socket = new Socket(SERVER_ADDRESS, SERVER_PORT);
             BufferedReader in = new BufferedReader(new InputStreamReader(s
ocket.getInputStream()));
             PrintWriter out = new PrintWriter(socket.getOutputStream(), tr
ue);
             BufferedReader stdIn = new BufferedReader(new InputStreamReade
r(System.in))) {

            // Thread for reading server messages
            new Thread(() -> {
                try {
                    String serverMessage;
                    while ((serverMessage = in.readLine()) != null) {
                        System.out.println(serverMessage);
                    }
                } catch (IOException e) {
                    System.out.println("Disconnected from server");
                }
            }).start();

            // Read user input and send to server
            String userInput;
            while ((userInput = stdIn.readLine()) != null) {
                out.println(userInput);
                if ("/quit".equalsIgnoreCase(userInput)) {
                    break;
                }
            }
        } catch (IOException e) {
            System.out.println("Error connecting to server: " + e.getMessag
e());
        }
    }
}
```

အသုံးပြုနည်း

Server စတင်ဖို့:

```
Command:

java server.ChatServer
```

Client တစ်ခုစတင်ဖို့ (မတူညီတဲ့ terminal မှာ):

```
Command:

java client.ChatClient
```

1. **Client အသစ်တွေထပ်ထည့်ဖို့**:
    - အပေါ်ကအတိုင်း client အသစ်တွေ run ပါ

Features

1. Multiple clients တွေတစ်ပြိုင်နက် chat လုပ်နိုင်ခြင်း
2. Client တစ်ခုချင်းစီကို name တောင်းပြီး identify လုပ်နိုင်ခြင်း
3. /quit command နဲ့ chat ကနေထွက်နိုင်ခြင်း
4. Real-time message broadcasting

Testing Results

Server Console

```
Console:
Chat Server is running on port 12345
New client connected
New client connected
```

Client 1 Console

Console:
```
Enter your name:
Alice
Alice has joined the chat!
Bob has joined the chat!
Bob: Hello everyone!
Alice: Hi Bob!
Bob has left the chat.
```

Client 2 Console

Console:
```
Enter your name:
Bob
Bob has joined the chat!
Alice: Hi Bob!
Bob: Goodbye!
/quit
```

ဖြည့်စွက်နိုင်တဲ့ Features

1. Private messaging (ဥပမာ `/msg Bob Hello`)
2. User list ကြည့်နိုင်တဲ့ command (ဥပမာ `/users`)
3. Chat room အမျိုးမျိုး ဖန်တီးနိုင်ခြင်း
4. Message history သိမ်းဆည်းခြင်း

ဒီ application ကို ပိုမိုကောင်းမွန်အောင် ဆက်လက် develop လုပ်နိုင်ပါတယ်။ လိုအပ်တဲ့ features တွေထပ်ထည့်ပြီး customize လုပ်နိုင်ပါတယ်။

Mini Games

1. Tic Tac Toe game
2. Snake Game

💻 Tic Tac Toe Game

Game Features:

1. **ဖွဲ့စည်းပုံ**:
    - 3x3 matrix အရွယ်ရှိတဲ့ char array တစ်ခုဖြင့်ဖော်ပြထားပါတယ်
    - '-' ကိုဗလာနေရာအဖြစ်သတ်မှတ်ထားပါတယ်
    - Player 1 အတွက် 'X' နှင့် Player 2 အတွက် 'O' အဖြစ်သတ်မှတ်ထားပါတယ်
2. **အဓိကလုပ်ဆောင်ချက်များ**:
    - printBoard() - လက်ရှိဘုတ်အခြေအနေကိုပြသပေးပါတယ်
    - play() - ကစားသမားရဲ့ move ကိုလက်ခံပြီးစစ်ဆေးပေးပါတယ်
    - checkWinner() - အနိုင်ရသူရှိမရှိစစ်ဆေးပေးပါတယ်
    - checkDraw() - ဘုတ်ပြည့်နေပြီလားစစ်ဆေးပေးပါတယ်
    - winningMessage() - အနိုင်ရ Message
    - drawMessage() - သရေကျ Message
3. **ဂိမ်းစည်းမျဉ်းများ**:
    - X ကစားသမားကိုအရင်စခိုင်းပါတယ်
    - ကစားသမားတစ်ဦးစီကိုသူတို့ရဲ့ move ရွေးခွင့်ပေးပါတယ်
    - တစ်စုံတစ်ဦးအနိုင်ရပါက သို့မဟုတ် ဘုတ်ပြည့်သွားပါက ဂိမ်းပြီးဆုံးပါတယ်
4. **အသုံးပြုပုံ**:
    - ပရိုဂရမ်ကို run ပါက ဂိမ်းစတင်ပါမယ်
    - ကစားသမားတစ်ဦးစီကို သူတို့ရဲ့ move အတွက် row နှင့် column ရွေးခိုင်းပါမယ် (1-9)
    - ဘုတ်ကိုအဆုံးတိုင်ပြသပေးပြီး ရလဒ်ကိုကြေညာပါမယ်

Code:

```
package games;

import java.util.Scanner;

public class TicTacToe {

	public static char b[]= {'-','-','-','-','-','-','-','-','-'
','-'};
	public static char symbol;
	public static Scanner sc=new Scanner(System.in);


	public static void main(String[] args) {
		int count=0;
		int player=0;

		boolean gameOver=false;
		while(!gameOver) {
			printBoard();
			player=count%2==0 ? 1 : 2;

			if(play(player)==false) {
				System.out.println("Invalid input...");
				count--;
			}

			if(checkWinner(player)==1 || checkWinner(player)==2)
			{
				printBoard();
				winningMessage(player);
				gameOver=true;
			}


			if(checkDraw()) {
				printBoard();
				drawMessage();
				gameOver=true;
			}

			count++;
		}

	}
```

```
private static boolean checkDraw() {
	boolean res=false;

	if(   b[0]!='-' &&
		b[1]!='-' &&
		b[2]!='-' &&

		b[3]!='-' &&
		b[4]!='-' &&
		b[5]!='-' &&

		b[6]!='-' &&
		b[7]!='-' &&
		b[8]!='-'

			)
		res=true;

	return res;
}

private static void drawMessage() {
	System.out.println("Opps!! No more moves... ");
	System.out.println("-------------");
	System.out.println(" It is a Draw ");
	System.out.println("-------------");
	System.out.println(" Game Over ");
	System.out.println(" ----------- ");


}

private static void winningMessage(int player) {
	System.out.println("Congratulations... ");
	System.out.println("-------------");
	System.out.println(" Player "+player+" Won!! ");
	System.out.println("-------------");
	System.out.println(" Game Over ");
	System.out.println(" ----------- ");

}

private static int checkWinner(int player) {
	int res=0;
	switch(player) {
	case 1:if( b[0]=='X' && b[1]=='X' && b[2]=='X' ||
			b[3]=='X' && b[4]=='X' && b[5]=='X' ||
			b[6]=='X' && b[7]=='X' && b[8]=='X' ||
```

```
                        b[0]=='X' && b[3]=='X' && b[6]=='X' ||
                        b[1]=='X' && b[4]=='X' && b[7]=='X' ||
                        b[2]=='X' && b[5]=='X' && b[8]=='X' ||

                        b[0]=='X' && b[4]=='X' && b[8]=='X' ||
                        b[2]=='X' && b[4]=='X' && b[6]=='X'

                    )
                res=1;
            break;
            case 2:if( b[0]=='O' && b[1]=='O' && b[2]=='O' ||
                        b[3]=='O' && b[4]=='O' && b[5]=='O' ||
                        b[6]=='O' && b[7]=='O' && b[8]=='O' ||

                        b[0]=='O' && b[3]=='O' && b[6]=='O' ||
                        b[1]=='O' && b[4]=='O' && b[7]=='O' ||
                        b[2]=='O' && b[5]=='O' && b[8]=='O' ||

                        b[0]=='O' && b[4]=='O' && b[8]=='O' ||
                        b[2]=='O' && b[4]=='O' && b[6]=='O'

                )
            res=2;
        break;

            }
            return res;
        }

        private static boolean play(int player) {
            boolean res=true;

            symbol=player==1 ? 'X':'O';
            System.out.println("Player "+player+" Turn ... ");
            int input=getInput();

            if(input==1 && b[0]=='-') {
                b[0]=symbol;
            }
            else if(input==2 && b[1]=='-') {
                b[1]=symbol;
            }
            else if(input==3 && b[2]=='-') {
                b[2]=symbol;
            }
            else if(input==4 && b[3]=='-') {
                b[3]=symbol;
            }
```

```
            else if(input==5 && b[4]=='-') {
                  b[4]=symbol;
            }
            else if(input==6 && b[5]=='-' ) {
                  b[5]=symbol;
            }
            else if(input==7 && b[6]=='-') {
                  b[6]=symbol;
            }
            else if(input==8 && b[7]=='-') {
                  b[7]=symbol;
            }
            else if(input==9 && b[8]=='-') {
                  b[8]=symbol;
            }
            else {
                  res=false;
            }

            return res;

      }

      private static int getInput() {
            System.out.println("Enter Input[1-9]:");
            return sc.nextInt();
      }

      private static void printBoard() {
            // TODO Auto-generated method stub

            System.out.println(" Tic Tac Toe Game  ");
            System.out.println("Player 1 = X, Player 2 = O ");
            System.out.println(" ----------------------");
            System.out.println("  "+b[0]+"  |  "+b[1]+"  |  "+b[2]+"
");
            System.out.println(" ----------------------");
            System.out.println("  "+b[3]+"  |  "+b[4]+"  |  "+b[5]+"
");
            System.out.println(" ----------------------");
            System.out.println("  "+b[6]+"  |  "+b[7]+"  |  "+b[8]+"
");
            System.out.println(" ----------------------");


      }

}
```

Game Output:

```
Tic Tac Toe Game
Player 1 = X, Player 2 = O
 ---------------------
  -  |  -  |  -
 ---------------------
  -  |  -  |  -
 ---------------------
  -  |  -  |  -
 ---------------------
Player 1 Turn ...
Enter Input[1-9]:
1
 Tic Tac Toe Game
Player 1 = X, Player 2 = O
 ---------------------
  X  |  -  |  -
 ---------------------
  -  |  -  |  -
 ---------------------
  -  |  -  |  -
 ---------------------
Player 2 Turn ...
Enter Input[1-9]:
2
 Tic Tac Toe Game
Player 1 = X, Player 2 = O
 ---------------------
  X  |  O  |  -
 ---------------------
  -  |  -  |  -
 ---------------------
  -  |  -  |  -
 ---------------------
Player 1 Turn ...
Enter Input[1-9]:
3
 Tic Tac Toe Game
Player 1 = X, Player 2 = O
 ---------------------
  X  |  O  |  X
 ---------------------
  -  |  -  |  -
 ---------------------
  -  |  -  |  -
```

```
 ---------------------
Player 2 Turn ...
Enter Input[1-9]:
```

💻 Snake Game

Game Features:

1. **ဂိမ်းဘုတ် (Board)**
    - WIDTH = 20, HEIGHT = 16 ဖြင့် 20x16 ဂရစ်ကွက်ကို ဖန်တီးထားပါတယ်။
    - Q = မြွေခေါင်း , o = မြွေကိုယ်, @ = အစာ။
2. **မြွေနဲ့ အစာ**
    - မြွေကို game board လယ်တည့်တည့် မှာ ချထားပါမယ်။
    - အစာကို ကျပန်းနေရာ မှာ ထုတ်ပေးပါမယ်။
3. **ဂိမ်းလုပ်ဆောင်ချက်များ**
    - clearScreen() : game board ကို refresh လုပ်ပါမယ်။
    - update() : လှုပ်ရှားမှုတွေကို အမြဲ update လုပ်ပါမယ်။
    - draw() : ဂိမ်းဘုတ်၊ အစာ နဲ့ မြွေ တွေကို ဆွဲပါမယ်။
4. **ထိန်းချုပ်မှု**
    - **W** : အထက်, **S** : အောက်, **A** : ဘယ်, **D** : ညာ, **Q** : ထွက်။

Code:

```
package games;


import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;
import java.io.IOException;

import javax.swing.JFrame;



public class SnakeGame extends JFrame {

	public static boolean gameOver=false;
	public static int width=20;
	public static int height=16;
	public static int score=0;
	public static char dir='0';

	public SnakeGame(){

		this.setTitle("Snake Game ");
		this.setLayout(null);
		this.setDefaultCloseOperation(EXIT_ON_CLOSE);

		this.setSize(400,100);
		this.addKeyListener(new KeyAdapter() {

			public void keyPressed(KeyEvent e) {
				dir=e.getKeyChar();
			}
		});

		this.setVisible(true);

	}

	public static void main(String[] args) throws
InterruptedException, IOException {
		// TODO Auto-generated method stub
		new SnakeGame1();

		Snake s=new Snake(width,height);
		Board b=new Board(width,height);
		Food f=new Food(width,height);
```

```
            while(!gameOver) {

                  clearScreen();
                  update(s,b,f,dir,width,height);
                  draw(s,b,f,gameOver);

            }

      }



      private static void update(Snake s, Board b, Food f, char dir,
int width, int height) {
            // TODO Auto-generated method stub
            if(s.L>0) {

                  //swapping new head position and tails position
                  for(int i=s.L-1;i>=1;i--) {

                        s.tailX[i]=s.tailX[i-1];
                        s.tailY[i]=s.tailY[i-1];
                  }

                  s.tailX[0]=s.x;
                  s.tailY[0]=s.y;
            }

            switch(dir) {

            case 'a':s.x--;break;
            case 'd':s.x++;break;
            case 'w':s.y--;break;
            case 's':s.y++;break;

            }


            if(s.x==f.x && s.y==f.y) {

                  s.eatFood();
                  f.reset(width, height, s);
                  score=score+10;

            }

      }
```

```
    private static void draw(Snake s, Board b, Food f, boolean
gameOver) {
          // top bar
          for(int i=0;i<width+2;i++)
              System.out.printf("#");
          System.out.printf("\n");

          //body drawing
          for(int i=0;i<height;i++) {
               for(int j=0;j<width+2;j++) {

                    if(j==0) {
                         System.out.print("#");
                    }
                    else if(j==width+1) {
                         System.out.print("#");
                    }
                    else if(s.x==j && s.y==i) {
                         System.out.print("Q");
                    }
                    else if(f.x==j && f.y==i) {
                         System.out.print("F");
                    }
                    else {

                         boolean isTail=false;

                         for(int k=0;k<s.L;k++) {
                              if(s.tailX[k]==j && s.tailY[k]==i)
{
                                   System.out.print("O");
                                   isTail=true;
                              }
                         }

                         if(!isTail) {
                              System.out.print(" ");
                         }

                    }

               }
               System.out.println();

          }
```

```
            // bottom bar
            for(int i=0;i<width+2;i++)
                  System.out.printf("#");
            System.out.printf("\n");

            System.out.println(" ================  ");
            System.out.println("Score :"+score);
            System.out.println("======================");



      }



      private static void clearScreen() throws InterruptedException,
IOException {
            // TODO Auto-generated method stub
            new
ProcessBuilder("cmd","/c","cls").inheritIO().start().waitFor();
      }

}

class Snake{

      int x;
      int y;
      int L;
      int[] tailX,tailY;

      public Snake(int width,int height) {
            this.x=width/2;
            this.y=height/2;
            this.L=0;
            this.tailX=new int[width*height];
            this.tailY=new int[width*height];
      }

      public void eatFood() {
            this.L++;
      }

}

class Board{
      int width;
      int height;
      public Board(int width,int height) {
```

```
            this.width=width;
            this.height=height;
      }

}

class Food{
      int x;
      int y;

      public Food(int width,int height) {
            this.x=(int) (Math.random() * width+1);
            this.y=(int) (Math.random() * height);

      }

      public void reset(int width,int height,Snake s) {
            this.x=(int) (Math.random()*width+1);
            this.y=(int) (Math.random()*height);

            if ((this.x == s.x) && (this.y == s.y)) {
                  s.eatFood();
                  reset(width, height, s);
            }

      }

}
```

References

1. https://www.w3schools.com/java/
2. Head First Java 3rd Edition 2023 by Sierra, Kathy Bates, Bert Gee
3. https://www.tutorialspoint.com/java
4. https://openai.com/index/chatgpt
5. https://chat.deepseek.com
6. Andrew Wellings. Concurrent and Real-Time Programming in Java. John Wiley & Sons, 2004.
7. Joshua Bloch. Effective Java Programming Language Guide. Addison–Wesley, 2001

■ စားရေးသူ၏ ကိုယ်ရေးအကျဉ်း

ဤစာအုပ်ကိုရေးသားပြုစုသူကတော့ ကျွန်တော် ဆရာတင်မိုင်ဇော် ဖြစ်ပါတယ်။ ကျွန်တော်က မြစ်ကြီးနားမြို့အထက (၁) မှာ အထက်တန်းကို ၁၉၉၄ မှာ အောင်မြင်ခဲ့ပါတယ်။ ပြီးတော့ ကျွန်တော် ဖိလိပိုင်နိုင်ငံ University of the Philippines (Los Banos) မှာ B.Sc Computer Science ကိုဆက်လက်ပညာသင်ယူခဲ့ပါတယ်။ ၂၀၀၄ မှာ B.Sc Computer Science နဲ့ ဘွဲ့ရကျောင်းပြီးခဲ့ပါတယ်။ ၂၀၀၅ မှာ မြန်မာပြည်ပြန်လာပြီး မြစ်ကြီးနား ဇာတိမြို့မှာ Northern City Computer Training Center ကွန်ပျူတာသင်တန်းကျောင်းကို စတင်တည်ထောင်ဖွင့်လှစ်ခဲ့ပါတယ်။ လေ့လာဆည်းပူးခဲ့တဲ့ programming IT ဘာသာရပ်တွေကို သင်ကြားပို့ချခဲ့တာ ဖြစ်ပါတယ်။ ၂၀၀၉ မှာ မလေးရှားနိုင်ငံ Kualalumpur မှာ Zepto IT Solution လို့ခေါ်တဲ့ အိုင်တီ Company မှာ Software Developer (programmer) အဖြစ် ၃ နှစ်တာ အလုပ်လုပ်ခဲ့ပါတယ်။ ကျန်းမာရေးအခြေနေကြောင့် ရန်ကုန်မြို့ကို ပြန်လာပြီး ဆေးကုသရင်း ရန်ကုန်မြို့မှာပဲ ValueStar Computer Institute မှာ ၆ နှစ်တာ IT Lecturer အနေနဲ့ရော IT Department Head အနေနဲ့ရော ဆက်လက်ခြေချခဲ့ပါတယ်။ ၂၀၁၇ မှာ ကိုယ်ပိုင် အိုင်တီသင်တန်းကျောင်းအဖြစ် Northern City Center နာမည်နဲ့ အရင်က မြစ်ကြီးနားမှာ တည်ထောင်ခဲ့ဖူး တဲ့ နာမည်ကို ပြန်လည်အသုံးပြုကာ ဖွင့်လှစ်ထားပြီး ၂၀၂၅ လက်ရှိအချိန်အထိ သင်တန်းကျောင်းကို ဦးစီးလုပ်ကိုင်နေဆဲ ဖြစ်ပါတယ်။

■ စာရေးသူ၏ အိုင်တီအတွေ့အကြုံ မှတ်တမ်းများ

Popular Web projects –

- Japan Used Car Sales & Show room
  https://www.sbtjapan.com/sbt-myanmar/
- Myanmar Traditional Boxing
  https://www.myanmartraditionalboxing.com.mm
- Myanmar Agriculture Machinery & Products
  https://tptyeeshinn.com.mm
- Real Estate
  https://www.saikhungnoung.com
- Malaysia Minimart
  https://familystore.com.my
- China Products & Fashion Sales
  https://youhome.space/
- Online Payment System
  https://ucapay.com.mm

Popular Point of Sales Application Projects –

- Malaysia Family Store Desktop Application and Mobile Application
- Myitkyina iGu Café & Gallery Desktop Application
- Myitkyina Hospital Blood Bank Desktop Application
- Myitkyina Fuji Food & Drinks Desktop Application
- Yangon Joyzone Stock Controller Desktop Application and Mobile Application
- Yangon Malihka Restaurant Desktop Application and Mobile Application
- Yangon Yuri Café Mobile Application and Mobile Application
- Yangon Gracevines Agarwood Desktop Application and Mobile Application

■ Documentary Photos

ပျော်ရွှင် ချမ်းမြေ့ကြပါစေ။

Northern City