Java - Interns Training

1.sum program

public class add

{

public static void main(String args[])

{

Scanner sc=new Scanner(System.in);

int sum=0,a=-1;

while(a!=0) {

System.out.println("enter the number or enter 0 to exit");

a = sc.nextInt(); sum += a;

}

System.out.println("sum="+sum);

}

}

2.Features of java

1. [Simple](https://www.javatpoint.com/features-of-java#Simple)
2. [Object-Oriented](https://www.javatpoint.com/features-of-java#Object-Oriented)
3. [Portable](https://www.javatpoint.com/features-of-java#Portable)
4. [Platform independent](https://www.javatpoint.com/features-of-java#Platform-independent)-write once run anywhere
5. [Secured](https://www.javatpoint.com/features-of-java#Secured)
6. [Robust](https://www.javatpoint.com/features-of-java#Robust)-strong memory management.
7. [Architecture neutral](https://www.javatpoint.com/features-of-java#Architecture-neutral)-no implementation dependent features, for example, the size of primitive types is fixed.
8. [Interpreted](https://www.javatpoint.com/features-of-java#Interpreted)
9. [High Performance](https://www.javatpoint.com/features-of-java#High-Performance)
10. [Multithreaded](https://www.javatpoint.com/features-of-java#Multithreaded)
11. [Distributed](https://www.javatpoint.com/features-of-java#Distributed)- run in all or multiple systems
12. [Dynamic](https://www.javatpoint.com/features-of-java#Dynamic)-load class at run time

3.usage of jdk,jvm,jre,jit

1.JDK

JDK is a software development kit used to develop Java applications. It includes tools needed for developing, compiling, debugging, and running Java programs.

🔹 **Components of JDK:**

* **JRE (Java Runtime Environment):** Provides the necessary libraries and environment to run Java applications.
* **Java Compiler (javac):** Converts Java source code (.java files) into bytecode (.class files).
* **Debugger (jdb):** Used for debugging Java programs.
* **JavaDoc:** Tool for generating documentation from Java source code comments.

2.JVM

JVM is an abstract machine that provides a runtime environment for executing Java programs. It converts Java bytecode into machine-specific code.

**🔹 JVM Architecture:**

1. **ClassLoader: Loads Java classes dynamically.**
2. **Runtime Data Areas: Includes heap, method area, stack, and program counter register.**
3. **Execution Engine: Executes Java bytecode using an interpreter or JIT compiler.**
4. **Garbage Collector: Reclaims unused memory.**
5. **Multithreading: break the whole process into threads that run simultaneously.**

**🔹 Platform Independence:  
JVM makes Java platform-independent by allowing the same bytecode to run on different operating systems (Windows, Linux, macOS).**

3.JRE2

JRE is a software package that provides the necessary environment to run Java applications. It includes the JVM, libraries, and other components needed for execution.

🔹 **Components of JRE:**

* **JVM (Java Virtual Machine)** – Converts bytecode into machine code.
* **Core Libraries** – Predefined Java libraries (e.g., java.lang, java.util).
* **Supportive Files** – Configuration files, property files, etc.

4.JIT

🔹 **Types of JIT Compilation:**

* **Method-Based Compilation:** Compiles only frequently used methods.
* **Adaptive Optimization:** Analyzes code execution and optimizes accordingly.

4.Java Versions

1.Records

* Introduces record to simplify immutable objects.
* No need for getters, setters, constructors, toString(), equals(), hashCode()—all are autogenerated.
* They are final,used in data transfer objects(dto),not to set getters and setters,created for imutable objects.
* Syntax:

record Person(String name, int age) {}

2.Sealed classes

Sealed Classes provide controlled inheritance, allowing only specific subclasses to extend them.

🔹 They improve security, maintainability, and pattern matching in Java.

🔹 Best used when you want to define a fixed set of subclass types.

**Syntax:**

sealed interface Shape permits Circle, Rectangle, Square { }

final class Circle implements Shape { double radius; }

final class Rectangle implements Shape { double length, width; }

final class Square implements Shape { double side; }

3.Hidden class

* improves Performance for Runtime-Generated Classes
* Frameworks like Spring, Hibernate, and ByteBuddy generate dynamic classes at runtime.
* Hidden classes are not stored permanently, reducing memory overhead.
* Faster execution since they are optimized for short-lived objects.
* Reflection- creating and accesing objecs and class and methods dynamically
* They are **not accessible** using standard reflection (Class.forName() won't work).

Steps

1. **MethodHandles.lookup()** – Gets a Lookup object with permissions to create hidden classes. This is a api, present in java.lang.
2. **defineHiddenClass()** – Dynamically loads a class from bytecode and marks it as hidden.
3. **Creating an Instance** – Uses reflection to instantiate the hidden class.
4. **Invoking Methods** – Calls a method inside the hidden class dynamically.

4.pattern matching

Helps to reduce the redundancy-assigning the statements

Example

public class PatternMatchingInstanceOf {

public static void main(String[] args) {

Object obj = "Hello, Java!";

if (obj instanceof String str) { // No explicit casting needed

System.out.println("String length: " + str.length());

}

}

}

5.Reducing the length of the code

Java 8: Multi-line strings required \n and concatenation.

String html = "<html>\n" +

" <body>\n" +

" <h1>Hello</h1>\n" +

" </body>\n" +

"</html>";

Java 21: Uses text blocks (""").

String html = """

<html>

<body>

<h1>Hello</h1>

</body>

</html>

""";

6.Fast compilation

* Graal JIT is a high-performance Java compiler that: ✔ Is written in Java itself (unlike C1 & C2, which are in C++,in traditional versions).
* They are fast and efficient.

7.API Development

Java 21: Factory Methods for Immutable Collections

Java 21 simplifies immutable collections by introducing factory methods:

✔ List.of() → Creates an immutable list.

✔ Set.of() → Creates an immutable set.

✔ Map.of() → Creates an immutable map.

Java 21: Introducing ScopedValue for Safer Thread-Scoped Storage

ScopedValue is a new API in Java 21 that improves thread-local storage by automatically handling cleanup. It ensures better performance and memory safety.

✔ Scoped values are immutable, making them safer for parallel execution.

✔ Automatic cleanup after execution, preventing memory leaks.

✔ More efficient than ThreadLocal in high-performance applications.

8.security

* Jigsaw Module System (Encapsulation, better security)
* Break down the code and organise it
* 🔴 Avoid finalize(): It is slow, unreliable, and leads to memory/resource leaks.
* ✅ Use AutoCloseable: It ensures fast, predictable, and safe resource cleanup.

9.Reflections

Java Reflection is a *process of examining or modifying the run time behavior of a class at run time*.

In spring boot, the new keywords are not used to create instances of the class, instead, we use @component to create instances of the class that are required.

They degrade the security and performance of the code

5.Type Of Variable & Data Type & Naming Convertion in java with example code

1.Type Of Variables

1.local variable

Inside a method or block only

2.Instance variable-non-static

Within a class that is associated with an objects

3.Static variable

Within the class that is shared among all the objects

2.Data types

| **Primitive Data Types** | byte, short, int, long, float, double, char, boolean |  |  |
| --- | --- | --- | --- |

| **Non-Primitive Data Types** | String, Arrays, Classes, Interfaces |  |  |
| --- | --- | --- | --- |

3.Naming Conventions

1.Class name-Start with an uppercase letter, use PascalCase

class EmployeeDetails {}

2.variable and method name- Start with a lowercase letter, use camelCase

void calculateSalary() {}

3.constant name-Use uppercase letters with underscores

final int MAX\_LIMIT = 100;

**Example**

package programs;

import java.util.Scanner;

public class Name\_Convention {

public static void main(String args[]) {

Scanner sc = new Scanner(System.in);

System.out.println("enter the number");

int number= sc.nextInt();

int digits= countDigits(number);

System.out.println(digits);

}

public static int countDigits(int number)

{

int count=0;

while(number>0)

{

++count;

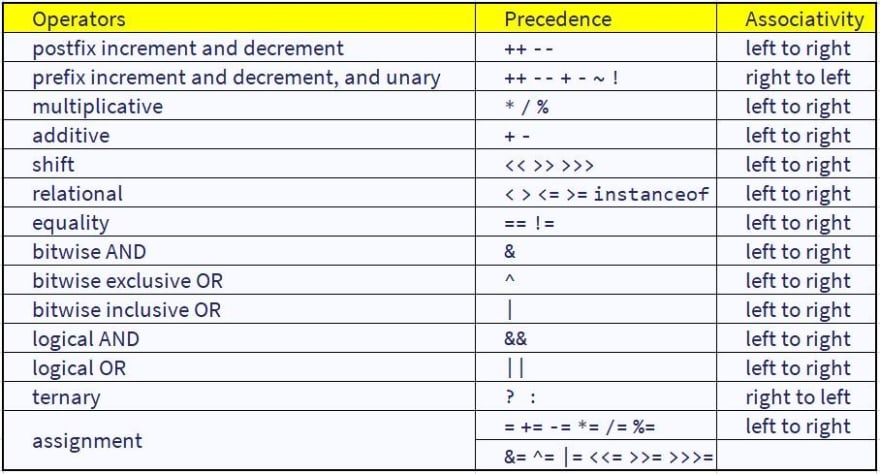
number/=10;

}

return count;

}

}

6.Java Operator Precedence with sample code

|  |
| --- |

|  |
| --- |

|  |
| --- |

package programs;

import java.util.Scanner;

public class operator\_precedence {

public static void main(String args[])

{

Scanner sc=new Scanner(System.in);

while(true) {

System.out.println("1.precedence evaluation 2.exit");

System.out.println("enter the choice");

String input = sc.nextLine().trim(); // Read input & trim spaces

int ch=-1;

try {

ch= Integer.parseInt(input); // Convert to integer

if (ch>= 1 && ch <= 2) {

if(ch==2)

break;

// Exit loop if valid

} else {

System.out.println("Error: Number must be between the given range");

}

} catch (NumberFormatException e) {

System.out.println("Error: Invalid input! Enter a number between the given range");

}

int a=0,b=0,c=0;

if(ch==1) {

while(true) {

System.out.println("enter the number1");

input = sc.nextLine().trim();

try {

a = Integer.parseInt(input);

if (a >= 1 && a <= 100) {

// Exit loop if valid

break;

} else {

System.out.println("Error: Number must be between the given range");

}

} catch (NumberFormatException e) {

System.out.println("Error: Invalid input! Enter a number between the given range");

}

}

while(true) {

System.out.println("enter the number2");

input = sc.nextLine().trim();

try {

b = Integer.parseInt(input);

if (b >= 1 && b <= 100) {

// Exit loop if valid

break;

} else {

System.out.println("Error: Number must be between the given range");

}

} catch (NumberFormatException e) {

System.out.println("Error: Invalid input! Enter a number between the given range");

}

}

while(true) {

System.out.println("enter the number3");

input = sc.nextLine().trim();

try {

c = Integer.parseInt(input);

if (c >= 1 && c <= 100) {

// Exit loop if valid

break;

} else {

System.out.println("Error: Number must be between the given range");

}

} catch (NumberFormatException e) {

System.out.println("Error: Invalid input! Enter a number between the given range");

}

}

// 1. Parentheses ()

int result1 = (a + b) \* c;

System.out.println("Parentheses: " + result1); // (10+5) \* 2 = 30

// 2. Unary Operators (++ -- ! ~)

int result2 = ++a - b--;

System.out.println("Unary: " + result2); // (11 - 5) = 6 (a=11, b=4)

// 3. Multiplicative (\* / %)

int result3 = a \* b / c % 3;

System.out.println("Multiplicative: " + result3); // (11 \* 4 / 2 % 3) = 1

// 4. Additive (+ -)

int result4 = a + b - c;

System.out.println("Additive: " + result4); // 11 + 4 - 2 = 13

// 5. Shift Operators (<< >> >>>)

int result5 = a << 1;

System.out.println("Bitwise Shift: " + result5); // 11 \* 2 = 22

// 6. Relational Operators (<, >, <=, >=)

boolean result6 = a > b;

System.out.println("Relational: " + result6); // 11 > 4 → true

// 7. Equality Operators (== !=)

boolean result7 = a != c;

System.out.println("Equality: " + result7); // 11 != 2 → true

// 8. Bitwise AND, XOR, OR (&, ^, |)

int result8 = a & b;

System.out.println("Bitwise AND: " + result8); // 11 & 4 → 0

int result9 = a ^ b;

System.out.println("Bitwise XOR: " + result9); // 11 ^ 4 → 15

int result10 = a | b;

System.out.println("Bitwise OR: " + result10); // 11 | 4 → 15

// 9. Logical AND (&&) and OR (||)

boolean result11 = (a > b) && (b < c);

System.out.println("Logical AND: " + result11); // (true && false) → false

boolean result12 = (a > b) || (b < c);

System.out.println("Logical OR: " + result12); // (true || false) → true

// 10. Ternary Operator (?:)

int result13 = (a > b) ? a : b;

System.out.println("Ternary: " + result13); // 11 (a > b is true)

}

}

}

}

**Tips**

**Input a number infinite times with valid number format**

while(true) {

System.out.println("enter the number3");

input = sc.nextLine().trim();

try {

c = Integer.parseInt(input);

if (c >= 1 && c <= 100) {

// Exit loop if valid

break;

} else {

System.out.println("Error: Number must be between the given range");

}

} catch (NumberFormatException e) {

System.out.println("Error: Invalid input! Enter a number between the given range");

}

}

7.N' Number of Student Marks List Generation in Table Format. Using Control Statements & Looping Statements

package programs;

import java.util.ArrayList;

import java.util.Scanner;

public class Student\_Marklist {

public static void main(String args[])

{

Scanner sc=new Scanner(System.in);

int n=0;

while(true)

{

System.out.println("enter the number of students");

String input = sc.nextLine().trim();

try {

n=Integer.parseInt(input);

if (n<=100&&n>=1) {

// Exit loop if valid

break;

} else {

System.out.println("Error: Number must be between the given range");

}

} catch (NumberFormatException e) {

System.out.println("Error: Invalid input! Enter a number between the given range");

}

}

ArrayList<Student> students=new ArrayList<>();

for(int i=0;i<n;i++)

{

Student s=new Student();

System.out.println("enter the details of student"+(i+1));

while(true)

{

System.out.println("name");

s.name = sc.nextLine().trim();

try {

if (s.name.matches("[A-Za-z]+( [A-Za-z]+)\*")) {

// Exit loop if valid

break;

} else {

System.out.println("Error: Number must be between the given range");

}

} catch (NumberFormatException e) {

System.out.println("Error: Invalid input! Enter a valid name ");

}

}

while(true)

{

System.out.println("subject 1 marks");

String input = sc.nextLine().trim();

try {

int mark=Integer.parseInt(input);

if (mark<=100&&mark>=1) {

// Exit loop if valid

s.sum+=mark;

s.marks.add(mark);

break;

} else {

System.out.println("Error: Number must be between the given range");

}

} catch (NumberFormatException e) {

System.out.println("Error: Invalid input! Enter a number between the given range");

}

}

while(true)

{

System.out.println("subject 2 marks");

String input = sc.nextLine().trim();

try {

int mark=Integer.parseInt(input);

if (mark<=100&&mark>=1) {

// Exit loop if valid

s.sum+=mark;

s.marks.add(mark);

break;

} else {

System.out.println("Error: Number must be between the given range");

}

} catch (NumberFormatException e) {

System.out.println("Error: Invalid input! Enter a number between the given range");

}

}

while(true)

{

System.out.println("subject 3 marks");

String input = sc.nextLine().trim();

try {

int mark=Integer.parseInt(input);

if (mark<=100&&mark>=1) {

// Exit loop if valid

s.sum+=mark;

s.marks.add(mark);

break;

} else {

System.out.println("Error: Number must be between the given range");

}

} catch (NumberFormatException e) {

System.out.println("Error: Invalid input! Enter a number between the given range");

}

}

s.average=s.sum/3;

students.add(s);

}

System.out.printf("%-30s %-10s %-10s %-10s %-10s %-10s%n",

"Name", "Subject1", "Subject2", "Subject3", "Total Marks","Average Marks");

for(int i=0;i<students.size();i++)

{

System.out.printf("%-30s %-10d %-10d %-10d %-10d %-10.2f%n",

students.get(i).name,

students.get(i).marks.get(0),

students.get(i).marks.get(1),

students.get(i).marks.get(2),

students.get(i).sum,

students.get(i).average);

}

}

public static class Student

{

String name;

ArrayList<Integer> marks=new ArrayList<>();

int sum;

double average;

}

}

8.fibonacci series coding with all validation

Used big integer

package programs;

import java.math.BigInteger;

import java.util.Scanner;

public class Fibanocciseries {

public static void main(String args[])

{

Scanner scanner = new Scanner(System.in);

int num;

// Input validation: Ensure the user enters a valid positive integer

while (true) {

System.out.print("Enter the number of Fibonacci terms: ");

if (scanner.hasNextInt()) {

num = scanner.nextInt();

if (num >= 0) {

break; // Valid input, exit loop

} else {

System.out.println("Error: Please enter a non-negative integer.");

}

} else {

System.out.println("Error: Invalid input. Please enter an integer.");

scanner.next(); // Consume invalid input

}

}

// Generate Fibonacci series

System.out.println("Fibonacci Series up to " + num + " terms:");

printFibonacci(num);

scanner.close();

}

// Function to print Fibonacci series using BigInteger

public static void printFibonacci(int n) {

if (n == 0) {

System.out.println("No terms to display.");

return;

}

BigInteger first = BigInteger.ZERO;

BigInteger second = BigInteger.ONE;

System.out.print(first); // Print first term

if (n > 1) {

System.out.print(", " + second); // Print second term

}

for (int i = 2; i < n; i++) {

BigInteger next = first.add(second);

System.out.print(", " + next);

first = second;

second = next;

}

System.out.println();

}

}

9.To Read Integer value stored in single variable and find following 1) Sum of Digits 2) Reverse Order 3) Find Biggest Number 4)Find Smallest Number 5)Find Average 6 )Finding Index position value 7)To Print String Format 8)Find the Mid elements 8)Find Amstrong Number 9)Total Number Of Counts 10)Particular Number Repeating Count 11)Replaceing Number

**Tips:**

1)int age;

while (true) {

try {

System.out.print("Enter your Age: ");

age= Integer.parseInt(sc.nextLine().trim());

break;

} catch (NumberFormatException e) {

System.out.println("enter valid choice number");

}

}

2) while (true) {

try {

System.out.println("1.sum 2.reverse 3.largest 4.smallest 5.average 6.find index 7.string format 8.mid element 9.Armstrong number" +

"10.total counts 11.number reptition 12.replacing number 13.exit");

System.out.println("enter the choice");

ch = Integer.parseInt(sc.nextLine().trim());

break;

} catch (NumberFormatException e) {

System.out.println("enter valid choice number");

}

}

3) while(true)

{

try {

System.out.println("enter the number");

num = Integer.parseInt(sc.nextLine().trim());

if(isValid(num,number))

break;

}

catch (NumberFormatException e)

{

System.out.println("enter valid number");

}

}

package programs;

import java.math.BigInteger;

import java.util.ArrayList;

import java.util.Scanner;

public class subprograms {

public static void main(String args[])

{

Scanner sc=new Scanner(System.in);

BigInteger num;

while (true) {

try {

System.out.println("Enter the number:");

// Read input directly as BigInteger

num = new BigInteger(sc.nextLine().trim()); // Trims spaces but still checks for empty input

break; // Exit loop if valid

} catch (NumberFormatException e) {

System.out.println("Enter a valid number (no spaces, no empty input):");

}

}

int ch=0;

while(true) {

while (true) {

try {

System.out.println("1.sum 2.reverse 3.largest 4.smallest 5.average 6.find index 7.string format 8.mid element 9.Armstrong number" +

"10.total counts 11.number reptition 12.replacing number 13.exit");

System.out.println("enter the choice");

ch = Integer.parseInt(sc.nextLine().trim());

break;

} catch (NumberFormatException e) {

System.out.println("enter valid choice number");

}

}

if (ch == 1)

System.out.println("sum:" + sumDigits(num));

else if (ch == 2)

System.out.println("reverse:" + reverseDigits(num));

else if (ch == 3) {

System.out.println("largest:" + largest(num));

}

else if(ch==4)

System.out.println("smallest:" + smallest(num));

else if(ch==5)

System.out.println("average:"+average(num));

else if(ch==6)

System.out.println("index position:"+findIndex(num));

else if (ch==7)

System.out.println("string format:"+stringFormat(num));

else if (ch==8)

System.out.println("mid element:"+midElement(num));

else if (ch==9)

System.out.println("armstrong:"+armstrong(num));

else if (ch==10)

System.out.println("total counts:"+totalCounts(num));

else if (ch==11)

System.out.println("number repeated times:"+repetition(num));

else if (ch==12)

System.out.println("After replacing:"+replacingNumber(num));

else if(ch==13)

break;

else

System.out.println("enter valid choice");

}

}

public static int sumDigits(BigInteger number)

{

BigInteger ten = BigInteger.TEN;

int sum = 0;

number = number.abs(); // Convert to positive for negative numbers

while (!number.equals(BigInteger.ZERO)) {

sum += number.mod(ten).intValue(); // Extract last digit

number = number.divide(ten); // Remove last digit

}

return sum;

}

public static BigInteger reverseDigits(BigInteger num)

{

boolean isNegative=false;

if(num.compareTo(BigInteger.ZERO)<0) {

isNegative = true;

num=num.multiply(BigInteger.valueOf(-1));

}

BigInteger rev=BigInteger.ZERO;

while(!num.equals(BigInteger.ZERO))

{

rev=rev.multiply(BigInteger.TEN).add(num.mod(BigInteger.TEN));

num=num.divide(BigInteger.TEN);

}

if (isNegative) {

rev = rev.multiply(BigInteger.valueOf(-1)); // Corrected syntax

}

return rev;

}

public static int largest(BigInteger number)

{

BigInteger ten = BigInteger.TEN;

int max = 0;

number = number.abs(); // Convert to positive for negative numbers

while (!number.equals(BigInteger.ZERO)) {

max=Math.max(number.mod(ten).intValue(),max);// Extract last digit

number = number.divide(ten); // Remove last digit

}

return max;

}

public static int smallest(BigInteger number)

{

BigInteger ten = BigInteger.TEN;

int min = Integer.MAX\_VALUE;

number = number.abs(); // Convert to positive for negative numbers

while (!number.equals(BigInteger.ZERO)) {

min=Math.min(number.mod(ten).intValue(),min);// Extract last digit

number = number.divide(ten); // Remove last digit

}

return min;

}

public static double average(BigInteger number)

{

BigInteger ten = BigInteger.TEN;

BigInteger num=number;

int sum = 0;

number = number.abs(); // Convert to positive for negative numbers

while (!number.equals(BigInteger.ZERO)) {

sum += number.mod(ten).intValue(); // Extract last digit

number = number.divide(ten); // Remove last digit

}

return sum/totalCounts(num);

}

public static int totalCounts(BigInteger number)

{

BigInteger ten = BigInteger.TEN;

int count=0;

number = number.abs(); // Convert to positive for negative numbers

while (!number.equals(BigInteger.ZERO)) {

count++; // Extract last digit

number = number.divide(ten); // Remove last digit

}

return count;

}

public static ArrayList<Integer> findIndex(BigInteger number)

{

ArrayList<Integer> list = new ArrayList<>();

Scanner sc=new Scanner(System.in);

int num=0;

while(list.isEmpty()) {

while(true)

{

try {

System.out.println("enter the number");

num = Integer.parseInt(sc.nextLine().trim());

if(isValid(num,number))

break;

}

catch (NumberFormatException e)

{

System.out.println("enter valid number");

}

}

BigInteger ten = BigInteger.TEN;

int totalcounts = totalCounts(number);

number = number.abs(); // Convert to positive for negative numbers

while (!number.equals(BigInteger.ZERO)) {

if (number.mod(ten).intValue() == num) {

list.add(totalcounts - 1);// Extract last digit

}

totalcounts--;

number = number.divide(ten); // Remove last digit

}

}

return list;

}

public static String stringFormat(BigInteger number)

{

BigInteger ten = BigInteger.TEN;

String str="";

number = number.abs(); // Convert to positive for negative numbers

while (!number.equals(BigInteger.ZERO)) {

str= word(number.mod(ten).intValue())+" "+str; // Extract last digit

number = number.divide(ten); // Remove last digit

}

return str;

}

public static String word(int n)

{

switch (n)

{

case 1:

return "one";

case 2:

return "two";

case 3:

return "three";

case 4:

return "four";

case 5:

return "five";

case 6:

return "six";

case 7:

return "seven";

case 8:

return "eight";

case 9:

return "nine";

default:

return "zero";

}

}

public static String armstrong(BigInteger number)

{

int power=totalCounts(number);

BigInteger num=number;

BigInteger ten = BigInteger.TEN;

BigInteger sum =BigInteger.ZERO;

number = number.abs(); // Convert to positive for negative numbers

while (!number.equals(BigInteger.ZERO)) {

BigInteger digit = number.mod(ten); // Extract last digit

sum = sum.add(digit.pow(power));

number = number.divide(ten); // Remove last digit

}

if(num.equals(sum))

return "yes";

else

return "no";

}

public static ArrayList<Integer> repetition(BigInteger number)

{

int[] arr=new int[10];

ArrayList<Integer> list=new ArrayList<>();

BigInteger ten = BigInteger.TEN;

number = number.abs(); // Convert to positive for negative numbers

while (!number.equals(BigInteger.ZERO)) {

arr[number.mod(ten).intValue()]++; // Extract last digit

number = number.divide(ten); // Remove last digit

}

int num;

Scanner sc=new Scanner(System.in);

while(list.isEmpty()) {

while(true)

{

try {

System.out.println("enter the number");

num = Integer.parseInt(sc.nextLine().trim());

break;

}

catch (NumberFormatException e)

{

System.out.println("enter valid number");

}

}

if(arr[num]!=0)

list.add(arr[num]);

else

System.out.println("enter the number present");

}

return list;

}

public static BigInteger replacingNumber(BigInteger number)

{

Scanner sc=new Scanner(System.in);

int num;

while(true)

{

try {

System.out.println("enter the number to replace");

num = Integer.parseInt(sc.nextLine().trim());

if(isValid(num,number))

break;

else

System.out.println("enter valid number present");

}

catch (NumberFormatException e)

{

System.out.println("enter valid number");

}

}

int anum;

while(true)

{

try {

System.out.println("enter the alternate number");

anum= Integer.parseInt(sc.nextLine().trim());

break;

}

catch (NumberFormatException e)

{

System.out.println("enter valid number");

}

}

String str=String.valueOf(number);

str= str.replace(String.valueOf(num),String.valueOf(anum));

BigInteger ans=new BigInteger(str);

return ans;

}

public static boolean isValid(int num,BigInteger number)

{

BigInteger ten = BigInteger.TEN;

int totalcounts = totalCounts(number);

number = number.abs(); // Convert to positive for negative numbers

while (!number.equals(BigInteger.ZERO)) {

if (number.mod(ten).intValue() == num) {

return true;

}

number = number.divide(ten); // Remove last digit

}

return false;

}

public static String midElement(BigInteger number)

{

String numStr = number.abs().toString();

int length = numStr.length();

// Calculate Mean (center of digit positions)

double mean = (length - 1) / 2.0;

// Find middle index based on mean

int middleIndex = (int) Math.round(mean);

// Determine middle digit(s)

String middleDigit;

if (length % 2 == 1) { // Odd-length number

middleDigit = String.valueOf(numStr.charAt(middleIndex));

} else { // Even-length number

middleDigit = numStr.substring(middleIndex - 1, middleIndex + 1);

}

return middleDigit;

}

}

10)To Read 'N' Number of value store in Single array variable, that array values copy to another array variable. Then Finding Sorting Number Acesndeing Oder & descending order, Finding the Repeating elements in that copy array

package programs;

import java.util.\*;

public class Array\_Operations {

static int n;

public static void main(String args[]){

Scanner sc=new Scanner(System.in);

while(true)

{

try {

System.out.println("enter the number");

n = Integer.parseInt(sc.nextLine().trim());

break;

}

catch (NumberFormatException e)

{

System.out.println("enter valid number");

}

}

int arr[]=new int[n];

for(int i=0;i<n;i++)

{

int element;

while(true)

{

try {

System.out.println("enter the number");

element=Integer.parseInt(sc.nextLine().trim());

break;

}

catch (NumberFormatException e)

{

System.out.println("enter valid number");

}

}

arr[i]=element;

}

int[] arrCopy=arr;

System.out.println("Copied Array:"+Arrays.toString(arrCopy));

sort(0,n-1,arr);

System.out.println("Ascending Order"+Arrays.toString(arr));

reverse(arr);

System.out.println("Descending Order:"+Arrays.toString(arr));

System.out.println("Duplicates:");

frequency(arr);

}

public static void sort(int start,int end, int arr[])

{

if(start>=end)

return;

int mid=start+(end-start)/2;

sort(start,mid,arr);

sort(mid+1,end,arr);

merge(start,mid,end,arr);

}

public static void merge(int start,int mid,int end,int[] arr)

{

int i=start,j=mid+1,k=start;

int temp[]=new int[n];

while(i<=mid&&j<=end)

{

if(arr[i]<arr[j])

{

temp[k++]=arr[i++];

}

else

temp[k++]=arr[j++];

}

while(i<=mid)

temp[k++]=arr[i++];

while(j<=end)

temp[k++]=arr[j++];

for(int l=start;l<=end;l++)

arr[l]=temp[l];

}

public static void reverse(int arr[])

{

int l=0,r=arr.length-1;

while(l<r)

{

int temp=arr[l];

arr[l]=arr[r];

arr[r]=temp;

l++;r--;

}

}

public static void frequency(int arr[])

{

HashMap<Integer,Integer> map=new HashMap<>();

for(int i=0;i<arr.length;i++)

{

if(!map.containsKey(arr[i]))

map.put(arr[i],1);

else

map.put(arr[i],map.get(arr[i])+1);

}

List<List<Integer>> list = new ArrayList<>();

for (Map.Entry<Integer, Integer> entry : map.entrySet()) {

ArrayList<Integer> row = new ArrayList<>();

row.add(entry.getKey()); // Store number first

row.add(entry.getValue()); // Store frequency second

list.add(row);

}

list.sort(Comparator.comparingInt(a -> a.get(0)));

// Printing the result

for (List<Integer> row : list) {

System.out.println(row.get(0) + " appears " + row.get(1) + " times.");

}

}

}

11)oops concepts

Abstraction and interface

package programs;

import java.util.Scanner;

// Interface Example

// ✅ Interface defining behavior

interface Drawable {

void draw();

// Abstract method (by default)

}

// ✅ Shape class (providing base logic)

abstract class Shape {

Shape()

{

System.out.println("abstract constructor call");

}

abstract double calculateArea();

// Abstract method

}

// ✅ Circle class implementing Drawable & extending Shape

class Circle extends Shape implements Drawable {

private double radius;

Circle(double radius) {

this.radius = radius;

}

@Override

double calculateArea() {

return Math.PI \* radius \* radius;

}

@Override

public void draw() {

System.out.println("Drawing Circle");

}

}

// ✅ Rectangle class implementing Drawable & extending Shape

class Rectangle extends Shape implements Drawable {

private double length, width;

Rectangle(double length, double width) {

this.length = length;

this.width = width;

}

@Override

double calculateArea() {

return length \* width;

}

@Override

public void draw() {

System.out.println("Drawing Rectangle");

}

}

// ✅ Main class to test interfaces

public class oops\_demo {

public static void main(String[] args) {

Drawable d1 = new Circle(7);

d1.draw();

System.out.println("Circle Area: " + ((Circle) d1).calculateArea()); // Type casting needed

Shape d2 = new Rectangle(5, 3);

System.out.println("Rectangle Area: " + d2.calculateArea());

}

}

12)Creating two class ( demo1, demo2),Inside a demo1 class creating three method for( leap year implementation, STDDEV method implementation, Display user Details (Name, Age, Address,Phone number,Dept)), Inside a demo2 class creating a two method for (Switch Case Own Implementation using Static Method), static Implementation for (static varibale, final varibale and Inc/Dec Operator implementation)

package programs;

import java.math.BigInteger;

import java.util.ArrayList;

import java.util.Scanner;

// ✅ Class 1: Demo1 (Contains three methods)

class Demo1 {

// ✅ 1. Leap Year Check

public void checkLeapYear(int year) {

if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0)) {

System.out.println(year + " is a Leap Year.");

} else {

System.out.println(year + " is NOT a Leap Year.");

}

}

// ✅ 2. Standard Deviation Calculation

public double calculateSTDDEV(double[] numbers) {

int n = numbers.length;

double sum = 0, mean, standardDeviation = 0;

// Calculate Mean

for (double num : numbers) {

sum += num;

}

mean = sum / n;

// Calculate Variance

for (double num : numbers) {

standardDeviation += Math.pow(num - mean, 2);

}

// Standard Deviation = sqrt(variance)

return Math.sqrt(standardDeviation / n);

}

// ✅ 3. Display User Details

public void displayUserDetails(ArrayList<Demo\_Class.user> list) {

System.out.printf("%-30s %-10s %-20s %-15s %-15s%n",

"Name", "Age", "Address", "Phone", "Department");

for (Demo\_Class.user user : list) {

System.out.printf("%-30s %-10d %-20s %-15s %-15s%n",

user.name, user.age, user.address, user.phone, user.dept);

}

}

}

// ✅ Class 2: Demo2 (Contains two methods)

class Demo2 {

// ✅ Static Variables

static int counter = 0; // Static variable

final static int MAX\_VALUE = 100; // Final variable (Cannot be changed)

// ✅ 1. Switch Case Implementation (Static Method)

public static void switchExample(int choice) {

switch (choice) {

case 1:

System.out.println("You selected Option 1: bad luck ");

break;

case 2:

System.out.println("You selected Option 2: bad luck");

break;

case 3:

System.out.println("You selected Option 3: good luck");

break;

case 4:

System.out.println("You selected Option 4: good luck");

break;

default:

System.out.println("Invalid choice! Please select a valid option.");

}

}

// ✅ 2. Static Implementation (Static Variable, Final Variable, Inc/Dec Operators)

public static void staticImplementation() {

System.out.println("\n--- Static & Final Variable Demonstration ---");

// Static variable manipulation

System.out.println("Initial Counter Value: " + counter);

counter++; // Increment operator

System.out.println("Counter after Increment: " + counter);

counter--; // Decrement operator

System.out.println("Counter after Decrement: " + counter);

// Final variable usage

System.out.println("Max Value (Final Variable): " + MAX\_VALUE);

// MAX\_VALUE++; // ❌ This will cause an error because final variables cannot be changed.

}

}

// ✅ Main Class

public class Demo\_Class {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

Demo1 obj1 = new Demo1();

// ⭐ 1. Leap Year Check

int year;

while (true) {

try {

System.out.println("enter the year");

year = Integer.parseInt(sc.nextLine().trim());

break;

} catch (NumberFormatException e) {

System.out.println("enter valid number");

}

}

obj1.checkLeapYear(year);

// ⭐ 2. Standard Deviation Calculation

int nofElemesnts;

while (true) {

try {

System.out.print("\nEnter the number of elements for STDDEV: ");

nofElemesnts = Integer.parseInt(sc.nextLine().trim());

break;

} catch (NumberFormatException e) {

System.out.println("enter valid number");

}

}

double[] numbers = new double[nofElemesnts];

System.out.println("Enter " + nofElemesnts + " numbers: ");

for (int i = 0; i < nofElemesnts; i++) {

while (true) {

try {

System.out.print("\nEnter the element ");

numbers[i] = Integer.parseInt(sc.nextLine().trim());

break;

} catch (NumberFormatException e) {

System.out.println("enter valid number");

}

}

}

double stddev = obj1.calculateSTDDEV(numbers);

System.out.println("Standard Deviation: " + stddev);

// ⭐ 3. Display User Details

// Consume newline

while (true) {

try {

System.out.print("\nEnter the number of users ");

nofElemesnts = Integer.parseInt(sc.nextLine().trim());

break;

} catch (NumberFormatException e) {

System.out.println("enter valid number");

}

}

ArrayList<Demo\_Class.user> list=new ArrayList<Demo\_Class.user>();

for(int i=0;i<nofElemesnts;i++){

String name;

while (true) {

System.out.print("\nEnter your Name: ");

name = sc.nextLine().trim();

try {

if (name.matches("[A-Za-z]+( [A-Za-z]+)\*")) {

// Exit loop if valid

break;

} else {

System.out.println("Error: Invalid name");

}

} catch (NumberFormatException e) {

System.out.println("Error: Invalid input! Enter a valid name ");

}

}

int age;

while (true) {

try {

System.out.print("Enter your Age: ");

age = Integer.parseInt(sc.nextLine().trim());

break;

} catch (NumberFormatException e) {

System.out.println("enter valid number");

}

}

// Consume newline

String address;

while (true) {

System.out.print("Enter your Address: ");

address = sc.nextLine().trim();

try {

if (address.matches("[A-Za-z0-9 ,.-]{5,100}")) {

// Exit loop if valid

break;

} else {

System.out.println("Error: Invalid number");

}

} catch (NumberFormatException e) {

System.out.println("Error: Invalid input! Enter a valid one ");

}

}

String phone;

while (true) {

System.out.print("Enter your Phone Number: ");

phone = sc.nextLine().trim();

try {

if (phone.matches("\\d{10,15}")) {

// Exit loop if valid

break;

} else {

System.out.println("Error: Invalid number");

}

} catch (NumberFormatException e) {

System.out.println("Error: Invalid input! Enter a valid one ");

}

}

String dept;

while (true) {

System.out.print("Enter your Department: ");

dept = sc.nextLine().trim();

try {

if (dept.matches("[A-Za-z]+( [A-Za-z]+)\*")) {

// Exit loop if valid

break;

} else {

System.out.println("Error: Invalid department");

}

} catch (NumberFormatException e) {

System.out.println("Error: Invalid input! Enter a valid one ");

}

}

Demo\_Class.user u=new Demo\_Class.user();

u.name=name;

u.age=age;

u.address=address;

u.phone=phone;

u.dept=dept;

list.add(u);

}

obj1.displayUserDetails(list);

// ⭐ 4. Switch Case Example

;

int choice;

while (true) {

try {

System.out.print("\nEnter your choice (1-4) for switch case example: ");

choice= Integer.parseInt(sc.nextLine().trim());

break;

} catch (NumberFormatException e) {

System.out.println("enter valid choice number");

}

}

Demo2.switchExample(choice);

// ⭐ 5. Static Implementation Example

Demo2.staticImplementation();

sc.close();

}

static class user

{

String name;

int age;

String address;

String phone;

String dept;

}

}

13)AES Algorithm

package programs;

import javax.crypto.Cipher;

import javax.crypto.SecretKeyFactory;

import javax.crypto.spec.IvParameterSpec;

import javax.crypto.spec.PBEKeySpec;

import javax.crypto.spec.SecretKeySpec;

import java.security.SecureRandom;

import java.security.spec.KeySpec;

import java.util.Arrays;

import java.util.Base64;

import java.util.Scanner;

public class AES\_Algorithm {

private static final String FACTORY\_INSTANCE = "PBKDF2WithHmacSHA256";

private static final String CIPHER\_INSTANCE = "AES/CBC/PKCS5PADDING";

private static final String SECRET\_KEY\_TYPE = "AES";

private static final byte[] IV\_CODE = new byte[16];

private static final String SECRET\_KEY = "yourSecretKey";

private static final int ITERATION\_COUNT = 65536;

private static final int KEY\_LENGTH = 256;

public static String encrypt(String secretKey, String salt, String value) throws Exception {

Cipher cipher = initCipher(secretKey, salt, Cipher.ENCRYPT\_MODE);

byte[] cipherText = cipher.doFinal(value.getBytes());

byte[] cipherWithIv = addIVToCipher(cipherText);

return Base64.getEncoder().encodeToString(cipherWithIv);

}

public static String decrypt(String secretKey, String salt, String encrypted) throws Exception {

Cipher cipher = initCipher(secretKey, salt, Cipher.DECRYPT\_MODE);

byte[] original = cipher.doFinal(Base64.getDecoder().decode(encrypted));

// unpad

byte[] originalWithoutIv = Arrays.copyOfRange(original, IV\_CODE.length, original.length);

return new String(originalWithoutIv);

}

private static Cipher initCipher(String secretKey, String salt, int mode) throws Exception {

SecretKeyFactory factory = SecretKeyFactory.getInstance(FACTORY\_INSTANCE);

KeySpec spec = new PBEKeySpec(secretKey.toCharArray(), salt.getBytes(), ITERATION\_COUNT, KEY\_LENGTH);

SecretKeySpec sKeySpec = new SecretKeySpec(factory.generateSecret(spec).getEncoded(), SECRET\_KEY\_TYPE);

Cipher cipher = Cipher.getInstance(CIPHER\_INSTANCE);

// Generating random IV

SecureRandom random = new SecureRandom();

random.nextBytes(IV\_CODE);

cipher.init(mode, sKeySpec, new IvParameterSpec(IV\_CODE));

return cipher;

}

private static byte[] addIVToCipher(byte[] cipherText) {

byte[] cipherWithIv = new byte[IV\_CODE.length + cipherText.length];

System.arraycopy(IV\_CODE, 0, cipherWithIv, 0, IV\_CODE.length);

System.arraycopy(cipherText, 0, cipherWithIv, IV\_CODE.length, cipherText.length);

return cipherWithIv;

}

public static void main(String[] args) throws Exception {

String fSalt = "anySaltYouCanUseOfOn";

Scanner scanner=new Scanner(System.in);

String plainText = "M0993000353";

while (true) {

System.out.print("\nEnter your Name: ");

plainText= scanner.nextLine().trim();

try {

if (plainText.matches("[A-Za-z]+( [A-Za-z]+)\*")) {

// Exit loop if valid

break;

} else {

System.out.println("Error: Invalid plain text");

}

} catch (NumberFormatException e) {

System.out.println("Error: Invalid input! Enter a valid text ");

}

}

String cipherText = encrypt(SECRET\_KEY, fSalt, plainText);

System.out.println("Cipher: " + cipherText);

String decryptedText = decrypt(SECRET\_KEY, fSalt, cipherText);

System.out.println("Decrypted: " + decryptedText);

}

}

Steps:

1)plaintext to binary(based on the ascii value)

2)xor with the secret key(cipher text)

3)add iv to the cipher text(iv is the random key or number)

4)convert them to base 64(normal alphabets)

14)Method overloading

package programs;

import java.util.Scanner;

public class Math\_Function\_Overloading {

public static void main(String args[])

{

Scanner sc=new Scanner(System.in);

double number1,number2;

while(true)

{

try {

System.out.println("enter the number1");

number1=Double.parseDouble(sc.nextLine().trim());

break;

}

catch (NumberFormatException e)

{

System.out.println("enter valid number");

}

}

while(true)

{

try {

System.out.println("enter the number2");

number2=Double.parseDouble(sc.nextLine().trim());

break;

}

catch (NumberFormatException e)

{

System.out.println("enter valid number");

}

}

calculate(number1);

calculate(number1,number2);

calculate((int) number1);

calculate((int) number1,(int) number2);

}

static void calculate(double number1)

{

System.out.println("Single double Type variable");

System.out.println("Absolute : "+Math.abs(number1));

System.out.println("square root : "+Math.sqrt(number1));

System.out.println("Cube root : "+Math.cbrt(number1));

System.out.println("Log : "+Math.log10(number1));

System.out.println("floor : "+Math.floor(number1));

System.out.println("ceil : "+Math.ceil(number1));

System.out.println("round-off : "+Math.round(number1));

System.out.println("round to int : "+Math.rint(number1));

}

static void calculate(double number1,double number2)

{

System.out.println(" 2 double type variable");

System.out.println("Maximum : "+Math.max(number1,number2));

System.out.println("Minimum : "+Math.min(number1,number2));

System.out.println("Power : "+Math.pow(number1,number2));

System.out.println("Hypotenuse : "+Math.hypot(number1,number2));

}

static void calculate(int number1,int number2) {

System.out.println("Double int variable");

System.out.println("Addition : " + Math.addExact(number1, number2));

System.out.println("Subtraction: " + Math.subtractExact(number1, number2));

System.out.println("Multiplication : " + Math.multiplyExact(number1, number2));

System.out.println("floor division : "+Math.floorDiv(number1,number2));

System.out.println("floor modulus : "+Math.floorMod(number1,number2));

}

static void calculate(int number1)

{

System.out.println("Single int variable");

System.out.println("Increment : "+Math.incrementExact(number1));

System.out.println("Decrement : "+Math.decrementExact(number1));

System.out.println("Cube root : "+Math.cbrt(number1));

}

}

15)Student Details Creation(Class Student - void userDetails(); void college details(); and double percentageCalculation();) Object Cloing Implemntation

16)Type casting

package programs;

class Animal {

void makeSound() {

System.out.println("Animal makes a sound");

}

}

class Dog extends Animal {

void bark() {

System.out.println("Dog barks");

}

}

public class Typecasting {

public static void main(String[] args) {

Animal a = new Dog(); // Upcasting (automatic)

a.makeSound(); // Allowed

// a.bark(); // Not allowed (because reference is of type Animal)

((Dog) a).bark();//downcasting

}

}

#### **Explanation:**

* Dog object is stored in an Animal reference.
* Only methods available in Animal can be accessed.

⚠ **If downcasting is done on an object that is not an instance of the child class, it will throw a ClassCastException**.

17)Wrapper class

They are slow and memory consuming as compared to primitive types.

They have null values and the primitive type cannot be null.

Autoboxing (Primitive → Wrapper)

int num = 50;

Integer obj = num;

Unboxing (Wrapper → Primitive)-implicitly

Integer obj = Integer.valueOf(25); // Wrapper Object

int num = obj; // Unboxing

Non-Autounboxing (Manual Unboxing)- optimised

Integer obj = Integer.valueOf(40);

int num = obj.intValue();

19) Date and Time

The reference to "1970" when discussing UTC time is because this date, specifically January 1st, 1970 at 00:00:00 UTC, is considered the "epoch" in Unix time, meaning it's the starting point from which all time is measured in Unix systems, and is used to calculate the number of seconds that have passed since then; essentially, it's the date chosen by Unix developers as a convenient reference point when creating the operating system, making it the standard for many computer systems today.

package programs;

import java.time.\*;

import java.time.format.DateTimeFormatter;

import java.util.Scanner;

public class Date\_and\_Time {

public static void main(String[] args) {

Scanner scanner=new Scanner(System.in);

// Get current local date and time

LocalDateTime now = LocalDateTime.now();

LocalDate currentDate = now.toLocalDate();

// Calculate before and after dates

LocalDate yesterday = currentDate.minusDays(1);

LocalDate tomorrow = currentDate.plusDays(1);

// Print current, previous, and next day

System.out.println("Current Date: " + currentDate);

System.out.println("Yesterday's Date: " + yesterday);

System.out.println("Tomorrow's Date: " + tomorrow);

// Date Formats

DateTimeFormatter format1 = DateTimeFormatter.ofPattern("yyyy-MM-dd");

DateTimeFormatter format2 = DateTimeFormatter.ofPattern("dd/MM/yyyy");

DateTimeFormatter format3 = DateTimeFormatter.ofPattern("MM/dd/yyyy");

DateTimeFormatter format4 = DateTimeFormatter.ofPattern("yyyy-MM-dd-HH-mm-ss-ns");

DateTimeFormatter format5 = DateTimeFormatter.ofPattern("MMMM dd, yyyy");

System.out.println("\nFormatted Dates:");

System.out.println("Format (yyyy-MM-dd) : " + currentDate.format(format1));

System.out.println("Format (dd/MM/yyyy) : " + currentDate.format(format2));

System.out.println("Format (MM/dd/yyyy) : " + currentDate.format(format3));

System.out.println("Format (yyyy-MM-dd-HH-mm-ss-ns): " + now.format(format4));

System.out.println("Format (Month dd, yyyy) : " + currentDate.format(format5));

// Convert Long Value to Date Format

long timestamp = 0;

while (true) {

System.out.println("Enter a long value:");

String input = scanner.nextLine().trim();

try {

timestamp = Long.parseLong(input);

break; // Exit the loop if the value is a valid long

} catch (NumberFormatException e) {

System.out.println("Error: Invalid input! Enter a valid long value.");

}

}

// Example long timestamp (milliseconds)

Instant instant = Instant.ofEpochMilli(timestamp);

ZonedDateTime utcTime = instant.atZone(ZoneId.of("UTC"));

ZonedDateTime localTime = instant.atZone(ZoneId.systemDefault());

System.out.println("\nConvert Long Value to Date:");

System.out.println("Local Date: " + localTime.format(format1));

System.out.println("UTC Date : " + utcTime.format(format1));

System.out.println("UTC Full Format: " + utcTime);

}

}

Notes:

1)Inheritance -IS-A relationship

2)Composition

Used to implement multiple inheritance

Has-A relationship

Dependent objects

A Car has an Engine

3)Aggregation

Has-A relationship

Independent objects

A University has a Department

4)Association

Uses-A Relationship

Generalization

A Student uses a Library

20.String operations

package programs;

import java.io.IOException;

import java.util.ArrayList;

import java.util.Scanner;

public class String\_Operations {

public static void main(String args[])

{

Scanner scanner=new Scanner(System.in);

String word1,word2;

while (true) {

System.out.print("Enter the string1: ");

word1 = scanner.nextLine().trim();

if (word1.matches("\\S+")) {

break;

} else {

System.out.println("Error: Invalid string");

}

}

while (true) {

System.out.print("Enter the string2: ");

word2 = scanner.nextLine().trim();

// Allow any characters (alphabets, numbers, special characters, spaces)

if (word2.matches("\\S+")) {

break;

} else {

System.out.println("Error: Invalid string");

}

}

applyStringMethods(word1);

compareStrings(word1,word2);

vowelsCount(word1);

findUppercase(word1);

findLowercase(word1);

extractNumbers(word1);

alternatePattern(word1);

char c;

while (true) {

System.out.print("Enter the character for frequency: ");

String input = scanner.nextLine();

// Check if the input is a single character

if (input.length() == 1 && input.matches("[A-Za-z]")) {

c = input.charAt(0);

// Check if the character is present in str1

if (word1.contains(String.valueOf(c))) {

break;

} else {

System.out.println("Error: Character not found in the given string. Try again.");

}

} else {

System.out.println("Error: Invalid input. Enter a single alphabet character.");

}

}

findFrequency(c,word1);

findBlankspaces(word1);

}

public static void applyStringMethods(String inputString) {

System.out.println("\nApplying Java Inbuilt String Methods:");

// Substring (First 5 chars if available)

System.out.println("substring(0, 5): " + (inputString.length() >= 5 ? inputString.substring(0, 5) : inputString));

// Index of a specific character

System.out.println("indexOf('a'): " + inputString.indexOf('a'));

// Replace character

System.out.println("replace('a', '\*'): " + inputString.replace('a', '\*'));

// Convert to Uppercase

System.out.println("toUpperCase(): " + inputString.toUpperCase());

// Convert to Lowercase

System.out.println("toLowerCase(): " + inputString.toLowerCase());

// Trim leading and trailing spaces

System.out.println("trim(): " + inputString.trim());

// Check if string starts with a specific prefix

System.out.println("startsWith(\"Java\"): " + inputString.startsWith("Java"));

// Check if string ends with a specific suffix

System.out.println("endsWith(\"Fun\"): " + inputString.endsWith("Fun"));

// Split the string by space

String[] words = inputString.split(" ");

System.out.println("split(\" \") (words count): " + words.length);

// Character at a specific position

if (inputString.length() > 2) {

System.out.println("charAt(2): " + inputString.charAt(2));

}

// Length of the string

System.out.println("length(): " + inputString.length());

// Replace all vowels with '\*'

System.out.println("replaceAll(\"[AEIOUaeiou]\", \"\*\"): " + inputString.replaceAll("[AEIOUaeiou]", "\*"));

// Convert string to char array

char[] charArray = inputString.toCharArray();

System.out.println("toCharArray() (First 5 chars): " + (charArray.length >= 5 ? new String(charArray, 0, 5) : new String(charArray)));

// Reverse string using StringBuilder

System.out.println("Reverse String: " + new StringBuilder(inputString).reverse().toString());

}

public static void compareStrings(String word1,String word2) {

System.out.println("\nString Comparisons:");

System.out.println("Using equals(): " + word1.equals(word2));

System.out.println("Using equalsIgnoreCase(): " + word1.equalsIgnoreCase(word2));

}

public static void vowelsCount(String word1)

{

int count=0;

ArrayList<Character> list=new ArrayList<>();

for(int i=0;i<word1.length();i++)

{

if(word1.charAt(i)=='a'||word1.charAt(i)=='e'||word1.charAt(i)=='i'||word1.charAt(i)=='o'||word1.charAt(i)=='u')

{

count++;

list.add(word1.charAt(i));

}

}

System.out.println("vowels count: "+count);

System.out.println("vowels are: "+list);

}

public static void findUppercase(String word1)

{

ArrayList<Character> list=new ArrayList<>();

for(int i=0;i<word1.length();i++)

{

if(Character.isUpperCase(word1.charAt(i)))

{

list.add(word1.charAt(i));

}

}

System.out.println("Upper case letters are: "+list);

}

public static void findLowercase(String word1)

{

ArrayList<Character> list=new ArrayList<>();

for(int i=0;i<word1.length();i++)

{

if(Character.isLowerCase(word1.charAt(i)))

{

list.add(word1.charAt(i));

}

}

System.out.println("Lower case letters are: "+list);

}

public static void findFrequency(Character character,String word1)

{

int count=0;

for(int i=0;i<word1.length();i++)

{

if(word1.charAt(i)==character)

{

count++;

}

}

System.out.println("Frequency of "+character+" is: "+count);

}

public static void findBlankspaces(String word1)

{

int count=0;

for(int i=0;i<word1.length();i++)

{

if(word1.charAt(i)==' ')

{

count++;

}

}

System.out.println("Blank spaces count: "+count);

}

public static void extractNumbers(String word1)

{

ArrayList<Character> list=new ArrayList<>();

for(int i=0;i<word1.length();i++)

{

if(Character.isDigit(word1.charAt(i)))

{

list.add(word1.charAt(i));

}

}

System.out.println("Numbers are: "+list);

}

public static void alternatePattern(String word1)

{

String result="";

for(int i=0;i<word1.length();i++)

{

if(i%2!=0)

{

if( Character.isAlphabetic(word1.charAt(i))) {

if (Character.isUpperCase(word1.charAt(i)))

result += word1.charAt(i) - 65+1;

else result += word1.charAt(i) - 97+1;

}

else

result+=word1.charAt(i);

}

else

result+=String.valueOf(word1.charAt(i));

}

System.out.println("Alternate pattern :"+result);

}

}

21.Employee Pay Slip using Inheritance

package programs;

import java.io.IOException;

import java.util.ArrayList;

import java.util.Scanner;

abstract class Employee

{

String employeeID;

String employeeName;

String Designation;

double basicPay;

abstract public void getDetail();

abstract public void print();

abstract public void calculateSalary();

}

class Salary extends Employee

{

Scanner scanner=new Scanner(System.in);

public void getDetail()

{

while (true) {

System.out.print("Enter the employee id ");

employeeID = scanner.nextLine();

if (employeeID.matches("^[A-Za-z0-9]+$")) {

break;

} else {

System.out.println("Error: Invalid string");

}

}

while (true) {

System.out.print("Enter the employee name: ");

employeeName = scanner.nextLine();

if (employeeName.matches("[A-Za-z]+( [A-Za-z]+)\*")) {

break;

} else {

System.out.println("Error: Invalid name");

}

}

while (true) {

System.out.print("Enter the Designation: ");

Designation = scanner.nextLine();

if (Designation.matches("[A-Za-z]+( [A-Za-z]+)\*")) {

break;

} else {

System.out.println("Error: Invalid designation");

}

}

while(true)

{

try {

System.out.println("enter the basic pay");

basicPay= Double.parseDouble(scanner.nextLine());

break;

}

catch (NumberFormatException e)

{

System.out.println("enter valid number");

}

}

}

double houserentAllowance;

double conveyenceAlowance;

double educationAllowance;

double grossPay;

double netPay;

double providendFund;

double tax;

public void calculateSalary()

{

houserentAllowance=0.01\*basicPay;

conveyenceAlowance=0.02\*basicPay;

educationAllowance=0.01\*basicPay;

providendFund=0.1\*basicPay;

tax=0.02\*basicPay;

grossPay=houserentAllowance+conveyenceAlowance+educationAllowance+basicPay;

netPay=grossPay-providendFund-tax;

}

public void print()

{

System.out.printf("%-15s%-25s%-20s%-15.2f%-30.2f%-30.2f%-30.2f%-30.2f\n",

employeeID, employeeName, Designation, basicPay,

houserentAllowance,conveyenceAlowance, educationAllowance , grossPay);

// Print net pay

System.out.printf("\n%-15s%-25s%-20s%-15s%-30s%-30s%-30s%-30s\n",

"", "", "", "", "", "", "Net Pay", netPay);

}

}

public class Employee\_Payslip {

public static void main(String args[])

{

Scanner scanner=new Scanner(System.in);

ArrayList<Employee> list=new ArrayList<>();

int nofEmployees;

while(true)

{

try {

System.out.println("enter the no of employees");

nofEmployees= Integer.parseInt(scanner.nextLine());

if(nofEmployees>=1&&nofEmployees<=6)

break;

else

System.out.println("enter the value less than 6");

}

catch (NumberFormatException e)

{

System.out.println("enter valid number");

}

}

for(int i=1;i<=nofEmployees;i++)

{

System.out.println("enter details of employee "+i);

Employee e=new Salary();

e.getDetail();

list.add(e);

}

System.out.printf("%-15s%-25s%-20s%-15s%-30s%-30s%-30s%-30s\n",

"Employee ID", "Employee Name", "Designation", "Basic Pay",

"House Rent Allowance", "Conveyance Allowance",

"Education Allowance", "Gross Pay");

for(Employee e:list)

{

e.calculateSalary();

e.print();

}

}

}

22.Personal loan calculation

Loan Calculation(combined)

package programs;

import java.util.ArrayList;

import java.util.Scanner;

class Customer

{

int customerID;

String customerName;

int bankID;

int loanID;

double principal;

double annualRate;

int tenureInYears;

double emi;

double finalAmount;

double interest;

Scanner scanner=new Scanner(System.in);

public void getDetail(ArrayList<Bank> banksList)

{

while (true) {

try {

System.out.println("enter the customer id: ");

customerID = Integer.parseInt(scanner.nextLine());

break;

} catch (NumberFormatException e) {

System.out.println("Invalid Format!");

}

}

while (true) {

System.out.print("Enter the customer name: ");

customerName = scanner.nextLine();

try {

if (customerName.matches("[A-Za-z]+( [A-Za-z]+)\*")) {

break;

} else {

System.out.println("Error: Invalid name");

}

}

catch (Exception e)

{

System.out.println(e);

}

}

while (true) {

try {

System.out.println("1-SBI 2-Axis Bank 3-HDFC Bank");

System.out.println("enter the bank name by bank numer: ");

bankID = Integer.parseInt(scanner.nextLine());

if (bankID >= 1 && bankID <=3)

{

annualRate=banksList.get(bankID-1).rateInYears;

break;

}

else

System.out.println("enter valid number");

} catch (NumberFormatException e) {

System.out.println("Invalid Format!");

}

}

while (true) {

try {

System.out.println("enter the principal amount: ");

principal = Double.parseDouble(scanner.nextLine());

break;

} catch (NumberFormatException e) {

System.out.println("Invalid Format!");

}

}

while (true) {

try {

System.out.println("enter the time period: ");

tenureInYears = Integer.parseInt(scanner.nextLine());

break;

} catch (NumberFormatException e) {

System.out.println("Invalid Format!");

}

}

}

public void calculateEMI()

{

}

@Override

public String toString() {

return "Customer ID: " + customerID + ", Name: " + customerName +

", Bank ID: " + bankID + ", Loan ID: " + loanID +

", Principal: " + principal + ", Rate: " + annualRate +

"%, Tenure: " + tenureInYears + " years";

}

}

class Personalloan extends Customer

{

double monthlyRate;

int tenureInMonths;

public void calculateEMI()

{

monthlyRate = annualRate / 12 / 100;

// Convert tenure from years to months

tenureInMonths = tenureInYears \* 12;

// EMI Formula: (P \* r \* (1 + r)^n) / ((1 + r)^n - 1)

emi = (principal \* monthlyRate \* Math.pow(1 + monthlyRate, tenureInMonths)) /

(Math.pow(1 + monthlyRate, tenureInMonths) - 1);

finalAmount=emi\*tenureInMonths;

interest=finalAmount-principal;

}

}

class Educationloan extends Customer

{

double monthlyRate;

int tenureInMonths;

public void calculateEMI()

{

monthlyRate = annualRate / 12 / 100; // Convert annual interest rate to monthly rate

int tenureInMonths = tenureInYears \* 12; // Convert tenure to months

// EMI Formula: EMI = [P \* R \* (1 + R)^n] / [(1 + R)^n - 1]

emi = (principal \* monthlyRate \* Math.pow(1 + monthlyRate, tenureInMonths)) /

(Math.pow(1 + monthlyRate, tenureInMonths) - 1);

finalAmount=emi\*tenureInMonths;

interest=finalAmount-principal;

}

}

class Bank

{

int bankID;

String bankName;

double rateInYears;

ArrayList<Customer> customersList=new ArrayList<>();

int loanID;

Bank(int bankID,String bankName,double rateInYears)

{

this.bankID=bankID;

this.bankName=bankName;

this.rateInYears=rateInYears;

}

}

public class Loan\_Calculation {

public static void main(String args[])

{

Scanner scanner=new Scanner(System.in);

Customer customer;

ArrayList<Bank> banksList=new ArrayList<>();

banksList.add(new Bank(1,"SBI",10));

banksList.add(new Bank(2,"Axis Bank",11));

banksList.add(new Bank(3,"HDFC Bank",12));

int loanchoice,nofCustomers;

while (true) {

try {

System.out.println("enter the total number of customers");

nofCustomers = Integer.parseInt(scanner.nextLine());

if (nofCustomers >= 1 && nofCustomers <= 6)

break;

else

System.out.println("enter number less than 6");

} catch (NumberFormatException e) {

System.out.println("Invalid Format!");

}

}

for(int i=0;i<nofCustomers;i++)

{

while (true) {

try {

System.out.println("1-Personal Loan 2-Education Loan 3-Exit");

System.out.println("enter the loan type by loan number");

loanchoice = Integer.parseInt(scanner.nextLine());

if (loanchoice >= 1 && loanchoice <= 3)

break;

else

System.out.println("enter valid number");

} catch (NumberFormatException e) {

System.out.println("Invalid Format!");

}

}

if (loanchoice==1)

{

customer= new Personalloan();

customer.loanID=loanchoice;

customer.getDetail(banksList);

customer.calculateEMI();

banksList.get(customer.bankID-1).customersList.add(customer);

//System.out.println(banksList.get(0).customersList.get(0));

}

else if(loanchoice==2)

{

customer=new Educationloan();

customer.loanID=loanchoice;

customer.getDetail(banksList);

customer.calculateEMI();

banksList.get(customer.bankID-1).customersList.add(customer);

}

else

break;

}

for (Bank b : banksList) {

if (!b.customersList.isEmpty()) { // Only print banks with customers

System.out.println("\nBank Name: " + b.bankName + " | Interest Rate: " + b.rateInYears + "%");

System.out.println("------------------------------------------------------------");

System.out.printf("%-10s %-15s %-10s %-12s %-8s %-10s %-10s %-10s\n",

"Cust ID", "Name", "Loan ID", "Principal", "Tenure", "EMI", "Final Amt","Interest");

for (Customer c : b.customersList) {

System.out.printf("%-10d %-15s %-10d %-12.2f %-8d %-10.2f %-10.2f %-10.2f\n",

c.customerID, c.customerName, c.loanID, c.principal,

c.tenureInYears, c.emi, c.finalAmount,c.interest);

}

System.out.println("------------------------------------------------------------\n");

}

}

}

}

Personal loan calculation

package programs;

import java.util.ArrayList;

import java.util.Scanner;

class Customer

{

int customerID;

String customerName;

int bankID;

int loanID;

double principal;

double annualRate;

int tenureInYears;

double emi;

double finalAmount;

double interest;

Scanner scanner=new Scanner(System.in);

public void getDetail(ArrayList<Bank> banksList)

{

while (true) {

try {

System.out.println("enter the customer id: ");

customerID = Integer.parseInt(scanner.nextLine());

break;

} catch (NumberFormatException e) {

System.out.println("Invalid Format!");

}

}

while (true) {

System.out.print("Enter the customer name: ");

customerName = scanner.nextLine();

try {

if (customerName.matches("[A-Za-z]+( [A-Za-z]+)\*")) {

break;

} else {

System.out.println("Error: Invalid name");

}

}

catch (Exception e)

{

System.out.println(e);

}

}

while (true) {

try {

System.out.println("1-SBI 2-Axis Bank 3-HDFC Bank");

System.out.println("enter the bank name by bank numer: ");

bankID = Integer.parseInt(scanner.nextLine());

if (bankID >= 1 && bankID <=3)

{

annualRate=banksList.get(bankID-1).rateInYears;

break;

}

else

System.out.println("enter valid number");

} catch (NumberFormatException e) {

System.out.println("Invalid Format!");

}

}

while (true) {

try {

System.out.println("enter the principal amount: ");

principal = Double.parseDouble(scanner.nextLine());

break;

} catch (NumberFormatException e) {

System.out.println("Invalid Format!");

}

}

while (true) {

try {

System.out.println("enter the time period(years): ");

tenureInYears = Integer.parseInt(scanner.nextLine());

break;

} catch (NumberFormatException e) {

System.out.println("Invalid Format!");

}

}

}

public void calculateEMI()

{

}

@Override

public String toString() {

return "Customer ID: " + customerID + ", Name: " + customerName +

", Bank ID: " + bankID + ", Loan ID: " + loanID +

", Principal: " + principal + ", Rate: " + annualRate +

"%, Tenure: " + tenureInYears + " years";

}

}

class Personalloan extends Customer

{

double monthlyRate;

int tenureInMonths;

public void calculateEMI()

{

monthlyRate = annualRate / 12 / 100;

// Convert tenure from years to months

tenureInMonths = tenureInYears \* 12;

// EMI Formula: (P \* r \* (1 + r)^n) / ((1 + r)^n - 1)

emi = (principal \* monthlyRate \* Math.pow(1 + monthlyRate, tenureInMonths)) /

(Math.pow(1 + monthlyRate, tenureInMonths) - 1);

finalAmount=emi\*tenureInMonths;

interest=finalAmount-principal;

}

}

class Bank

{

int bankID;

String bankName;

double rateInYears;

ArrayList<Customer> customersList=new ArrayList<>();

int loanID;

Bank(int bankID,String bankName,double rateInYears)

{

this.bankID=bankID;

this.bankName=bankName;

this.rateInYears=rateInYears;

}

}

public class Loan\_Calculation {

public static void main(String args[])

{

Scanner scanner=new Scanner(System.in);

Customer customer;

ArrayList<Bank> banksList=new ArrayList<>();

banksList.add(new Bank(1,"SBI",10));

banksList.add(new Bank(2,"Axis Bank",11));

banksList.add(new Bank(3,"HDFC Bank",12));

int loanchoice,nofCustomers;

while (true) {

try {

System.out.println("enter the total number of customers");

nofCustomers = Integer.parseInt(scanner.nextLine());

if (nofCustomers >= 1 && nofCustomers <= 6)

break;

else

System.out.println("enter number less than 6");

} catch (NumberFormatException e) {

System.out.println("Invalid Format!");

}

}

for(int i=1;i<=nofCustomers;i++)

{

System.out.println("enter the details of the customer-"+i);

customer= new Personalloan();

customer.loanID=i;

customer.getDetail(banksList);

customer.calculateEMI();

banksList.get(customer.bankID-1).customersList.add(customer);

//System.out.println(banksList.get(0).customersList.get(0));

}

for (Bank b : banksList) {

if (!b.customersList.isEmpty()) { // Only print banks with customers

System.out.println("\nBank Name: " + b.bankName + " | Interest Rate: " + b.rateInYears + "%");

System.out.println("------------------------------------------------------------");

System.out.printf("%-10s %-15s %-10s %-12s %-8s %-10s %-10s %-10s\n",

"Cust ID", "Name", "Loan ID", "Principal", "Tenure", "EMI", "Final Amt","Interest");

for (Customer c : b.customersList) {

System.out.printf("%-10d %-15s %-10d %-12.2f %-8d %-10.2f %-10.2f %-10.2f\n",

c.customerID, c.customerName, c.loanID, c.principal,

c.tenureInYears, c.emi, c.finalAmount,c.interest);

}

System.out.println("------------------------------------------------------------\n");

}

}

}

}

Educational Loan

package programs;

import java.util.ArrayList;

import java.util.Scanner;

class Customers

{

int customerID;

String customerName;

int bankID;

int loanID;

double principal;

double annualRate;

int tenureInYears;

double emi;

double finalAmount;

double interest;

int courseDuration;

boolean isSubsidued;

int subsidued;

Scanner scanner=new Scanner(System.in);

public void getDetail(ArrayList<Banks> banksList)

{

while (true) {

try {

System.out.println("enter the customer id: ");

customerID = Integer.parseInt(scanner.nextLine());

break;

} catch (NumberFormatException e) {

System.out.println("Invalid Format!");

}

}

while (true) {

System.out.print("Enter the customer name: ");

customerName = scanner.nextLine();

try {

if (customerName.matches("[A-Za-z]+( [A-Za-z]+)\*")) {

break;

} else {

System.out.println("Error: Invalid name");

}

}

catch (Exception e)

{

System.out.println(e);

}

}

while (true) {

try {

System.out.println("1-SBI 2-Axis Bank 3-HDFC Bank");

System.out.println("enter the bank name by bank numer: ");

bankID = Integer.parseInt(scanner.nextLine());

if (bankID >= 1 && bankID <=3)

{

annualRate=banksList.get(bankID-1).rateInYears;

break;

}

else

System.out.println("enter valid number");

} catch (NumberFormatException e) {

System.out.println("Invalid Format!");

}

}

while (true) {

try {

System.out.println("enter the principal amount: ");

principal = Double.parseDouble(scanner.nextLine());

break;

} catch (NumberFormatException e) {

System.out.println("Invalid Format!");

}

}

while (true) {

try {

System.out.println("enter the time period(years): ");

tenureInYears = Integer.parseInt(scanner.nextLine());

break;

} catch (NumberFormatException e) {

System.out.println("Invalid Format!");

}

}

while (true) {

try {

System.out.println("enter the duration of the course: ");

courseDuration = Integer.parseInt(scanner.nextLine());

break;

} catch (NumberFormatException e) {

System.out.println("Invalid Format!");

}

}

while (true) {

try {

System.out.println("Are you subsidued by government 1-yes 2-no ");

subsidued= Integer.parseInt(scanner.nextLine());

if(subsidued==1)

{

isSubsidued=true;

break;

}

else if(subsidued==2)

{

isSubsidued=false;

break;

}

else

System.out.println("enter valid inputs either 1 or 2");

} catch (NumberFormatException e) {

System.out.println("Invalid Format!");

}

}

}

public void calculateEMI()

{

}

@Override

public String toString() {

return "Customer ID: " + customerID + ", Name: " + customerName +

", Bank ID: " + bankID + ", Loan ID: " + loanID +

", Principal: " + principal + ", Rate: " + annualRate +

"%, Tenure: " + tenureInYears + " years";

}

}

class Educational\_Loan extends Customers

{

double monthlyRate;

int tenureInMonths;

public void calculateEMI()

{

if (isSubsidued) {

annualRate -= 2; // Example: Reduce interest rate by 2% for subsidized loans

}

if(!isSubsidued)

{

// Calculate extra interest accumulated during grace period (Simple Interest)

double interestDuringGrace = (principal \* annualRate \* courseDuration) / (12 \* 100);

// New principal after grace period

double newPrincipal = principal + interestDuringGrace;

}

double monthlyRate = annualRate / 12 / 100;

int totalMonths = (tenureInYears \* 12) + courseDuration; // Grace period is included

emi = (principal \* monthlyRate \* Math.pow(1 + monthlyRate, totalMonths)) /

(Math.pow(1 + monthlyRate, totalMonths) - 1);

finalAmount = emi \* totalMonths;

interest = finalAmount - principal;

}

}

class Banks

{

int bankID;

String bankName;

double rateInYears;

ArrayList<Customers> customersList=new ArrayList<>();

int loanID;

Banks(int bankID,String bankName,double rateInYears)

{

this.bankID=bankID;

this.bankName=bankName;

this.rateInYears=rateInYears;

}

}

class Educational\_Loan\_Calculation {

public static void main(String args[])

{

Scanner scanner=new Scanner(System.in);

Customers customer;

ArrayList<Banks> banksList=new ArrayList<>();

banksList.add(new Banks(1,"SBI",8));

banksList.add(new Banks(2,"Axis Bank",9));

banksList.add(new Banks(3,"HDFC Bank",10));

int loanchoice,nofCustomers;

while (true) {

try {

System.out.println("enter the total number of customers");

nofCustomers = Integer.parseInt(scanner.nextLine());

if (nofCustomers >= 1 && nofCustomers <= 6)

break;

else

System.out.println("enter number less than 6");

} catch (NumberFormatException e) {

System.out.println("Invalid Format!");

}

}

for(int i=1;i<=nofCustomers;i++)

{

System.out.println("enter the details of the customer-"+i);

customer= new Educational\_Loan();

customer.loanID=i;

customer.getDetail(banksList);

customer.calculateEMI();

banksList.get(customer.bankID-1).customersList.add(customer);

//System.out.println(banksList.get(0).customersList.get(0));

}

for (Banks b : banksList) {

if (!b.customersList.isEmpty()) { // Only print banks with customers

System.out.println("\nBank Name: " + b.bankName + " | Interest Rate: " + b.rateInYears + "%");

System.out.println("------------------------------------------------------------");

System.out.printf("%-10s %-15s %-10s %-12s %-8s %-10s %-10s %-10s\n",

"Cust ID", "Name", "Loan ID", "Principal", "Tenure", "EMI", "Final Amt","Interest");

for (Customers c : b.customersList) {

System.out.printf("%-10d %-15s %-10d %-12.2f %-8d %-10.2f %-10.2f %-10.2f\n",

c.customerID, c.customerName, c.loanID, c.principal,

c.tenureInYears, c.emi, c.finalAmount,c.interest);

}

System.out.println("------------------------------------------------------------\n");

}

}

}

}

24.Invoice Bill Generation Using Interface Implementation

**Interface notes**

interface Invoice {

Scanner scanner = new Scanner(System.in);

// Default method to get common customer details

default void getCustomerDetails() {

System.out.print("Enter Customer ID: ");

int customerID = scanner.nextInt();

scanner.nextLine(); // Consume newline

System.out.print("Enter Customer Name: ");

String customerName = scanner.nextLine();

System.out.print("Enter Phone Number: ");

String phoneNumber = scanner.nextLine();

// Store these details in the implementing class

setCustomerDetails(customerID, customerName, phoneNumber);

}

// Abstract method to set customer details in implementing classes

void setCustomerDetails(int id, String name, String phone);

void getDetails(); // Method to input specific details

void calculateTotal(); // Method to calculate total

void displayInvoice(); // Method to display invoice

}

// Product Invoice Class

class ProductInvoice implements Invoice {

int customerID;

String customerName, phoneNumber;

String productName;

int quantity;

double pricePerUnit;

double totalAmount;

@Override

public void setCustomerDetails(int id, String name, String phone) {

this.customerID = id;

this.customerName = name;

this.phoneNumber = phone;

}

@Override

public void getDetails() {

getCustomerDetails(); // Call the common method to get customer details

System.out.print("Enter Product Name: ");

productName = scanner.nextLine();

System.out.print("Enter Quantity: ");

quantity = scanner.nextInt();

System.out.print("Enter Price per Unit: ");

pricePerUnit = scanner.nextDouble();

}

getCustomerDetails() works because default methods in interfaces can be called from instances of implementing classes, just like normal methods!

If it is put as static then,

Invoice.getCustomerDetails();

If it is made as public , then it cannot be defined in the interface class, it can only be defined in the subclass that implements it, as a mandatory one.

**String Buffer and String Builder**

String Buffer -Used in threads for synchronization and it is slow due to synchronization

String Builder-It is Asynchronous and fast, but in other ways both are the same.

Arrays.sort()

In Java, the Arrays.sort() method **uses different sorting algorithms** based on the type of data being sorted:

1. **For Primitive Types (int[], double[], char[], etc.)** → Uses **Dual-Pivot QuickSort**
2. **For Non-Primitive Types (Integer[], String[], CustomObjects[], etc.)** → Uses **TimSort**

Dual pivot quick sort

* Unlike the standard **QuickSort** (which uses **one pivot**), **Dual-Pivot QuickSort** selects **two pivots**.
* The array is **partitioned into three parts**:
* Elements **smaller than the first pivot**
* Elements **between the two pivots**
* Elements **greater than the second pivot**
* **Recursively sort** each partition.

Tim Sort

* **Divides the array** into small sorted segments (**runs**).
* **Sorts small runs using Insertion Sort** (efficient for small data).
* **Merges sorted runs using Merge Sort**.

**Streams**

A Stream in Java is a sequence of elements that supports functional-style operations on collections like Lists, Sets, and Arrays. It allows processing data in a declarative and parallel way.

## 🔹 When to Use Streams?

✅ When processing large data collections (filtering, mapping, etc.).  
✅ When method chaining makes code more readable.  
✅ When you need parallel processing for performance boost.

Memory MAnagement

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

// Create a stream from the list

Stream<String> nameStream = names.stream();

**Here,**  the namestream is stores the reference of the names list in stack.

"On the fly" means processing data immediately, without storing it.

**Lazy evaluation** (only processes when needed). And discards immediately if it is not stored in a list.

* **Notes**
* Only the data is operated from the source(list,set,etc)
* The new streams and the operations are created in the heap on **ForkJoinPool**
* Only **references to the source collection** are maintained in the stack
* The memory allocation in heap is done only when the stream is collected or stored in a new list or set.
* The unused streams are garbage collected onsumed (forEach()) ,hence they rely on garbage collection.
* Usually streams are sequential , but it can be made parallel, leading to multithreading.

Methods

long evenCount = numbers.stream() .filter(n -> n % 2 == 0) .count();

int sum = numbers.stream() .reduce(0, (a, b) -> a + b);

boolean hasEven = numbers.stream().anyMatch(n -> n % 2 == 0);

boolean allEven = numbers.stream().allMatch(n -> n % 2 == 0);

Sample Program

package programs;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.List;

import java.util.stream.Collector;

import java.util.stream.Collectors;

import java.util.stream.Stream;

public class Stream\_Implementation {

public static void main(String args[])

{

List<Integer> numberList= Arrays.asList(11,41,61,15,13);

//Stream Creation

//filter()

numberList.stream().filter(n->n%2==0).forEach(n->System.out.print(n+" "));

System.out.println();

//Map()

numberList.stream().map(n->n\*n).forEach(n->System.out.print(n+" "));

System.out.println();

//sorted()

numberList.stream().sorted().forEach(n->System.out.print(n+" "));

System.out.println();

// Sort names in reverse order

numberList.stream().sorted((a, b) -> b.compareTo(a)).forEach(n->System.out.print(n+" "));

System.out.println();

//parallel stream

numberList.parallelStream().forEach(n -> System.out.print(Thread.currentThread().getName() + " - " + n+" "));

System.out.println();

//Sequential Stream by default

numberList.stream().forEach(n -> System.out.print(Thread.currentThread().getName() + " - " + n+" "));

System.out.println();

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");// this is not stored just printed and removed from memory

// Collecting results (stored in memory)

List<String> upperCaseNames = names.stream()

.map(String::toUpperCase)

.collect(Collectors.toList());

}

}

Files methods

| exists() | Checks if file/directory exists |
| --- | --- |
| createNewFile() | Creates a new file |
| mkdir() | Creates a single directory |
| mkdirs() | Creates nested directories |
| isFile() | Checks if it’s a file |
| isDirectory() | Checks if it’s a directory |
| list() | Lists file names in a directory |
| listFiles() | Lists File objects in a directory |
| getName() | Gets file name |
| getAbsolutePath() | Gets full file path |
| length() | Gets file size in bytes |
| canRead() | Checks if readable |
| canWrite() | Checks if writable |
| lastModified() | Gets last modified timestamp |
| renameTo() | Renames or moves a file |
| delete() | Deletes a file/directory |

Threading

Locks

Race condition

class SharedResource {

int count = 0;

void increment() {

count++; // Not thread-safe

}

}

public class RaceConditionExample {

public static void main(String[] args) {

SharedResource resource = new SharedResource();

Thread t1 = new Thread(() -> {

for (int i = 0; i < 1000; i++) resource.increment();

});

Thread t2 = new Thread(() -> {

for (int i = 0; i < 1000; i++) resource.increment();

});

t1.start();

t2.start();

try {

t1.join();

t2.join();

} catch (InterruptedException e) {

e.printStackTrace();

}

System.out.println("Final Count: " + resource.count); // Expected: 2000, but may be less due to race condition

}

}

Methods of lock

1)Synchronisation

Simple use cases, small programs

class SharedResource {

int count = 0;

synchronized void increment() { // Lock acquired on this object

count++;

}

}

2)(Explicit Locking)

Complex applications, Deadlock handling

import java.util.concurrent.locks.ReentrantLock;

class SharedResource {

private int count = 0;

private final ReentrantLock lock = new ReentrantLock(); // Explicit lock

void increment() {

lock.lock(); // Acquire lock

try {

count++;

} finally {

lock.unlock(); // Always release lock

}

}

int getCount() {

return count;

}

}

3) ReadWriteLock (Optimized Locking for Reads and Writes)

Read-heavy applications (e.g., Caches)

import java.util.concurrent.locks.ReentrantReadWriteLock;

class SharedResource {

private int count = 0;

private final ReentrantReadWriteLock lock = new ReentrantReadWriteLock();

void increment() {

lock.writeLock().lock(); // Exclusive write lock

try {

count++;

} finally {

lock.writeLock().unlock();

}

}

int getCount() {

lock.readLock().lock(); // Shared read lock

try {

return count;

} finally {

lock.readLock().unlock();

}

}

}

4)StampedLock (Optimized Read-Write Locking)

High-performance read-heavy scenarios

import java.util.concurrent.locks.StampedLock;

class SharedResource {

private int count = 0;

private final StampedLock lock = new StampedLock();

void increment() {

long stamp = lock.writeLock(); // Exclusive write lock

try {

count++;

} finally {

lock.unlockWrite(stamp);

}

}

int getCount() {

long stamp = lock.tryOptimisticRead(); // Non-blocking optimistic read

int currentCount = count;

if (!lock.validate(stamp)) { // If data changed, get a proper read lock

stamp = lock.readLock();

try {

currentCount = count;

} finally {

lock.unlockRead(stamp);

}

}

return currentCount;

}

}

Atomic variables

The count variable is declared as

import java.util.concurrent.atomic.AtomicInteger;

class AtomicCounter {

private AtomicInteger count = new AtomicInteger(0);

void increment() {

count.incrementAndGet(); // Thread-safe atomic operation

}

int getCount() {

return count.get();

}

}

public class AtomicExample {

public static void main(String[] args) {

AtomicCounter counter = new AtomicCounter();

Thread t1 = new Thread(() -> {

for (int i = 0; i < 1000; i++) counter.increment();

});

Thread t2 = new Thread(() -> {

for (int i = 0; i < 1000; i++) counter.increment();

});

t1.start();

t2.start();

try {

t1.join();

t2.join();

} catch (InterruptedException e) {

e.printStackTrace();

}

System.out.println("Final Count: " + counter.getCount()); // Always 2000

}

}

Thread-safe Collections

* Most of the Collections classes objects (like ArrayList, LinkedList, HashMap etc) are non-synchronized in nature i.e. multiple threads can perform on a object at a time simultaneously. Therefore objects are not thread-safe.
* Very few Classes objects (like Vector, Stack, HashTable) are synchronized in nature i.e. at a time only one thread can perform on an Object. But here the problem is performance is low because at a time single thread execute an object and rest thread has to wait.

Notes

Class-level synchronization— static synchronized-locks entire class

Object-level– without static

Join()--> waits for the thread completion and then only go to the next thread for execution. Hence, ensures synchronization.

B+ tree→searching(mysql,postgres)

B-tree→ retrieval of records