**Name:-** Ketan Rathod

**Roll No.:-** D-43  
**Subject:-** Java

Day-1

**2.Write the “Hello World” - first Java Program. Compile and execute it.**

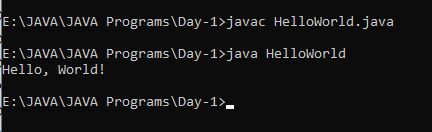
class HelloWorld {

public static void main(String[] args) {

System.out.println("Hello, World!");

}

}



**3. Write comments (Author, date, objective of the program, main function) in the above program.**

class p2 {

public static void main(String[] args) {

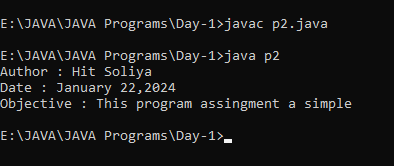
System.out.println("Author : Hit Soliya");

System.out.println("Date : January 22,2024");

System.out.println("Objective : This program assingment a simple");

}

}



**4. Create following arrays and assign incremental values starting from 1 to each element.**

**1. Array with 2 rows and 3 columns.**

**2. Array with 2 rows. First row has 3 cells, and second row has 2 cells.**

**Print the content of both the arrays.**

class IncrementalArrays {

public static void main(String[] args) {

int[][] array1 = new int[2][3];

fillArray(array1);

int[][] array2 = new int[2][];

array2[0] = new int[3];

array2[1] = new int[2];

fillArray(array2);

System.out.println("Array 1:");

printArray(array1);

System.out.println("\nArray 2:");

printArray(array2);

}

private static void fillArray(int[][] array) {

int value = 1;

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

for (int j = 0; j < array[i].length; j++) {

array[i][j] = value++;

}

}

}

private static void printArray(int[][] array) {

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

for (int j = 0; j < array[i].length; j++) {

System.out.print(array[i][j] + " ");

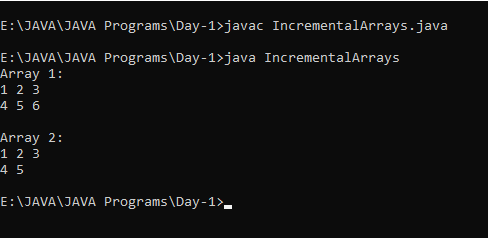
}

System.out.println();

}

}

}



**6. Write a program to print your name and address.**

class NameAndAddress {

public static void main(String[] args) {

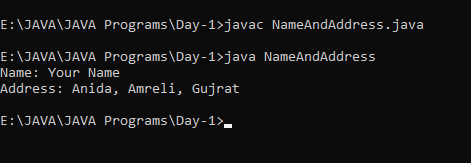
// Print name and address

System.out.println("Name: Your Name");

System.out.println("Address: Anida, Amreli, Gujrat");

}

}



Day-2

1. **Write a program to perform addition of two numbers and display its answer. (No need to take user input)**

class AdditionProgram {

public static void main(String[] args) {

int num1 = 5;

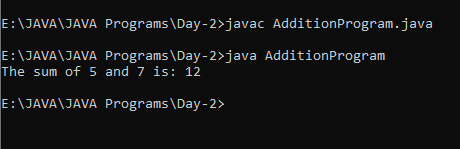
int num2 = 7;

int sum = num1 + num2;

System.out.println("The sum of " + num1 + " and " + num2 + " is: " + sum);

}

}



1. **Write a program to perform addition of two given numbers and display its answer. (With user input)**

import java.util.Scanner;

class AdditionWithInput {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the first number: ");

int num1 = scanner.nextInt();

System.out.print("Enter the second number: ");

int num2 = scanner.nextInt();

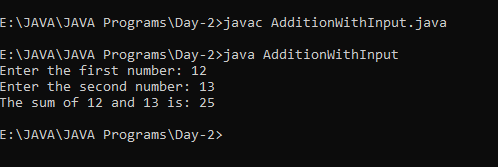
scanner.close();

int sum = num1 + num2;

System.out.println("The sum of " + num1 + " and " + num2 + " is: " + sum);

}

}



1. **Write a program to perform addition of numbers between 3 to 10 using for loop.**

class AdditionWithForLoop {

public static void main(String[] args) {

int startNumber = 3;

int endNumber = 10;

int sum = 0;

for (int i = startNumber; i <= endNumber; i++) {

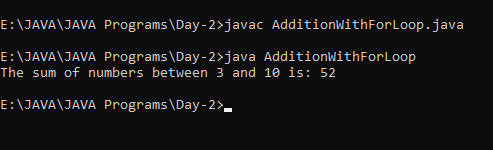
sum += i;

}

System.out.println("The sum of numbers between " + startNumber + " and " + endNumber + " is: " + sum);

}

}



1. **Write a program to find the maximum from two numbers. (Without user input)**

class MaximumOfTwoNumbers {

public static void main(String[] args) {

int num1 = 15;

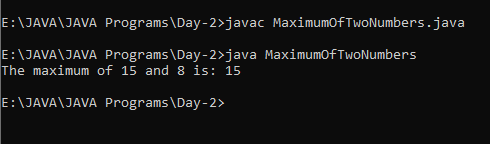
int num2 = 8;

int maximum = (num1 > num2) ? num1 : num2;

System.out.println("The maximum of " + num1 + " and " + num2 + " is: " + maximum);

}

}



1. **Check the working of the following code and give comments with respect to use of “outer” :**

**outer:**

**for (int i = 0; i < 10; i++) {**

**for (int j = 0; j < 10; j++) {**

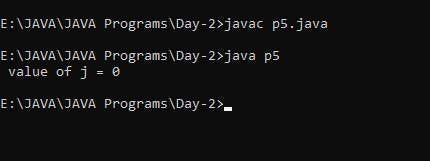
**if (j == 1)**

**break outer;**

**System.out.println(" value of j = " + j);**

**}**

**} // end of outer loop**

****

1. **Write a program to create three objects of Student class(name, age). Write getDetails method which takes user inputs (name and age). Write other method named displayDetails to display the name and age of the user. Create three objects and test working of above two methods.**

import java.util.\*;

class getSdetail {

String name;

int age;

void getDetails() {

Scanner sc = new Scanner(System.in);

System.out.print("Enter student name: ");

name = sc.nextLine();

System.out.print("Enter student age: ");

age = sc.nextInt();

}

void displayDetails() {

System.out.println("Student Name: " + name);

System.out.println("Student Age: " + age);

}

}

class Sgd {

public static void main(String[] args) {

getSdetail student1 = new getSdetail();

getSdetail student2 = new getSdetail();

getSdetail student3 = new getSdetail();

System.out.println("Enter details for Student 1:");

student1.getDetails();

System.out.println("\nEnter details for Student 2:");

student2.getDetails();

System.out.println("\nEnter details for Student 3:");

student3.getDetails();

System.out.println("\nDetails for Student 1:");

student1.displayDetails();

System.out.println("\nDetails for Student 2:");

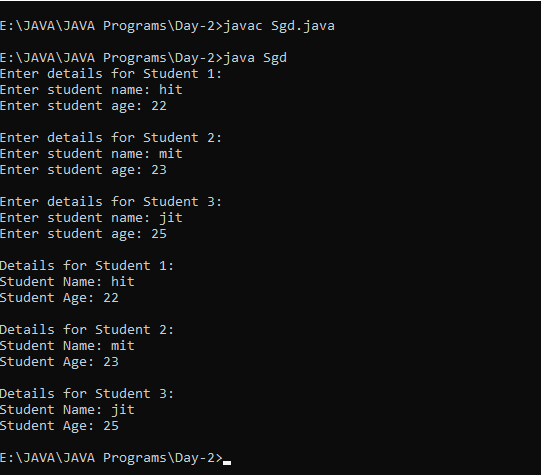
student2.displayDetails();

System.out.println("\nDetails for Student 3:");

student3.displayDetails();

}

}



Day-3

1. **­­­­­­­­­­­ Modify the Student class to add default, parameterized & copy constructors. Create three objects of the Student class to test the above three constructors. Print the content of the all the three objects.**
2. **Modify the above program to jump from one constructor to other constructor.**
3. **Add initializer and static block in the above program.**

class Student2 {

String name;

int age;

public Student2() {

name = "Unknown";

age = 0;

}

public Student2(String name, int age) {

this.name = name;

this.age = age;

}

public Student2(Student2 otherStudent) {

this(otherStudent.name, otherStudent.age); // Jump to parameterized constructor

}

{

System.out.println("Initializer Block Called");

}

static {

System.out.println("Static Block Called");

}

void displayDetails() {

System.out.println("Student Name: " + name);

System.out.println("Student Age: " + age);

}

}

class Stestp1 {

public static void main(String[] args) {

Student2 s1 = new Student2(); // Default constructor

Student2 s2 = new Student2("Alice", 20); // Parameterized constructor

Student2 s3 = new Student2(s2); // Copy constructor

System.out.println("Details for Student 1:");

s1.displayDetails();

System.out.println("\nDetails for Student 2:");

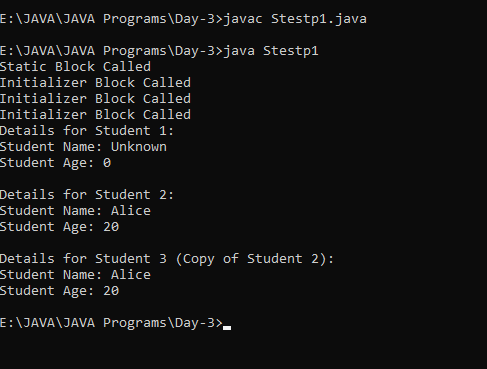
s2.displayDetails();

System.out.println("\nDetails for Student 3 (Copy of Student 2):");

s3.displayDetails();

}

}

****

Day-4

1. **Modify the Student class to add a method which will compare two other Student’s age and print the name of student with highest age.**
2. **Modify the above method so that it returns the Student’s object having the highest age. Print the name of the student and test your program. Test your program.**

class Student {

String name;

int age;

public Student() {

this("Unknown", 0);

}

public Student(String name, int age) {

this.name = name;

this.age = age;

}

public Student(Student otherStudent) {

this(otherStudent.name, otherStudent.age);

}

void displayDetails() {

System.out.println("Student Name: " + name);

System.out.println("Student Age: " + age);

System.out.println();

}

static void compareAges(Student student1, Student student2) {

if (student1.age > student2.age) {

System.out.println(student1.name + " has the highest age.");

} else if (student1.age < student2.age) {

System.out.println(student2.name + " has the highest age.");

} else {

System.out.println("Both students have the same age.");

}

}

}

class Max {

public static void main(String[] args) {

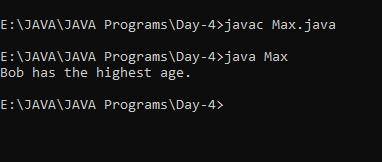
Student student1 = new Student("Alice", 22);

Student student2 = new Student("Bob", 25);

Student.compareAges(student1, student2);

}

}



1. **Create a Calculator class to add two int and to add two floats. Use method overloading.**

class Calculator {

int add(int num1, int num2) {

return num1 + num2;

}

float add(float num1, float num2) {

return num1 + num2;

}

}

class CalculatorTest {

public static void main(String[] args) {

Calculator calculator = new Calculator();

int sumIntegers = calculator.add(5, 7);

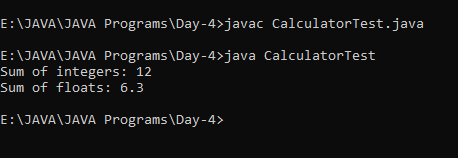
System.out.println("Sum of integers: " + sumIntegers);

float sumFloats = calculator.add(3.5f, 2.8f);

System.out.println("Sum of floats: " + sumFloats);

}

}



1. **In the Student class add static variable to store the number of the objects created. Display the value of the count using the static method. Test your program.**

class Std {

String name;

int age;

static int studentCount = 0; // Static variable to store the number of objects created

public Std() {

this("Unknown", 0);

studentCount++; // Increment count when a new object is created

}

public Std(String name, int age) {

this.name = name;

this.age = age;

studentCount++; // Increment count when a new object is created

}

public Std(Std otherStudent) {

this(otherStudent.name, otherStudent.age);

}

void displayDetails() {

System.out.println("Student Name: " + name);

System.out.println("Student Age: " + age);

System.out.println();

}

static void displayStudentCount() {

System.out.println("Number of Students: " + studentCount);

}

}

public class studentcount {

public static void main(String[] args) {

Std student1 = new Std();

Std student2 = new Std("Alice", 22);

Std student3 = new Std(student2);

System.out.println("Details for Student 1:");

student1.displayDetails();

System.out.println("Details for Student 2:");

student2.displayDetails();

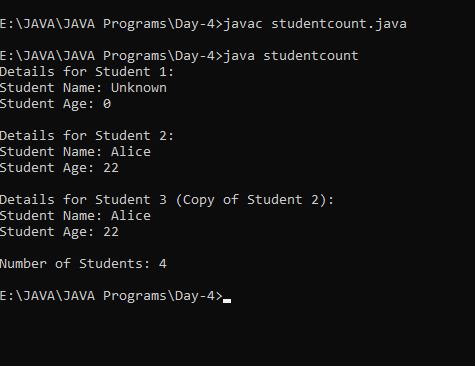
System.out.println("Details for Student 3 (Copy of Student 2):");

student3.displayDetails();

Std.displayStudentCount();

}

}



1. **Create an Outer class with instance variable named “a”. Inside this Outer class, create an Inner class. This inner class must have instance variable named “b”. Assign some initial values to this “a” and “b”. Create object of Outer class and print values of “a” and “b”.**

class Outer {

int a;

Outer(int a) {

this.a = a;

}

class Inner {

int b;

Inner(int b) {

this.b = b;

}

}

public static void main(String[] args) {

Outer outerObj = new Outer(10);

Outer.Inner innerObj = outerObj.new Inner(20);

System.out.println("Value of 'a': " + outerObj.a);

System.out.println("Value of 'b': " + innerObj.b);

}

}

Day-5

1. **Create following classes:**
   1. **Emp(id,name,age)**
   2. **PartTimeEmp(numberOfHours,ratePerHour) inherits Emp**
   3. **FullTimeEmp(basicPay,DA) inherits Emp**

**In all the three classes, add default and parameterized constructors. Create one object of both the child classes and print the content of both the objects.**

**In the all the three classes, add getDetails and showDetails methods. Create one object of both the child classes and print content of both the objects.**

class Emp {

int id;

String name;

int age;

public Emp() {

this(0, "Unknown", 0);

}

public Emp(int id, String name, int age) {

this.id = id;

this.name = name;

this.age = age;

}

public void getDetails() {

}

public void showDetails() {

System.out.println("Employee ID: " + id);

System.out.println("Employee Name: " + name);

System.out.println("Employee Age: " + age);

}

}

class PartTimeEmp extends Emp {

int numberOfHours;

double ratePerHour;

public PartTimeEmp() {

super();

this.numberOfHours = 0;

this.ratePerHour = 0.0;

}

public PartTimeEmp(int id, String name, int age, int numberOfHours, double ratePerHour) {

super(id, name, age);

this.numberOfHours = numberOfHours;

this.ratePerHour = ratePerHour;

}

public void showDetails() {

super.showDetails();

System.out.println("Number of Hours: " + numberOfHours);

System.out.println("Rate per Hour: " + ratePerHour);

}

}

class FullTimeEmp extends Emp {

double basicPay;

double DA;

public FullTimeEmp() {

super(); // Call the default constructor of the base class

this.basicPay = 0.0;

this.DA = 0.0;

}

public FullTimeEmp(int id, String name, int age, double basicPay, double DA) {

super(id, name, age); // Call the parameterized constructor of the base class

this.basicPay = basicPay;

this.DA = DA;

}

public void showDetails() {

super.showDetails();

System.out.println("Basic Pay: " + basicPay);

System.out.println("Dearness Allowance (DA): " + DA);

}

}

class EmpTest {

public static void main(String[] args) {

PartTimeEmp partTimeEmp = new PartTimeEmp(101, "John", 25, 20, 15.0);

FullTimeEmp fullTimeEmp = new FullTimeEmp(102, "Alice", 30, 50000.0, 2000.0);

System.out.println("Details of PartTimeEmp:");

partTimeEmp.showDetails();

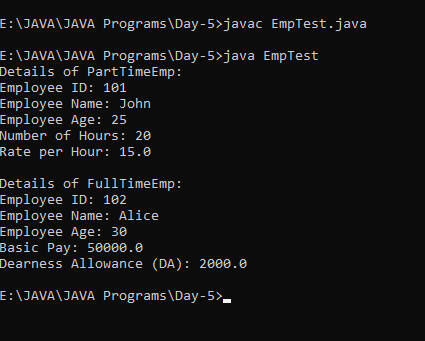
System.out.println();

System.out.println("Details of FullTimeEmp:");

fullTimeEmp.showDetails();

}

}



Day-6

1. **In the day-5 program, create object / reference of Emp class which will point to any of the child class. Call the getDetails and showDetails methods and check working of polymorphism.**

class Emp {

int id;

String name;

int age;

public Emp() {

this.id = 0;

this.name = "";

this.age = 0;

}

public Emp(int id, String name, int age) {

this.id = id;

this.name = name;

this.age = age;

}

public String getDetails() {

return "ID: " + id + ", Name: " + name + ", Age: " + age;

}

public void showDetails() {

System.out.println(getDetails());

}

}

class PartTimeEmp extends Emp {

int numberOfHours;

double ratePerHour;

public PartTimeEmp() {

super();

this.numberOfHours = 0;

this.ratePerHour = 0.0;

}

public PartTimeEmp(int id, String name, int age, int numberOfHours, double ratePerHour) {

super(id, name, age);

this.numberOfHours = numberOfHours;

this.ratePerHour = ratePerHour;

}

public String getDetails() {

return super.getDetails() + ", Number of Hours: " + numberOfHours + ", Rate per Hour: " + ratePerHour;

}

}

class FullTimeEmp extends Emp {

double basicPay;

double DA;

public FullTimeEmp() {

super();

this.basicPay = 0.0;

this.DA = 0.0;

}

public FullTimeEmp(int id, String name, int age, double basicPay, double DA) {

super(id, name, age);

this.basicPay = basicPay;

this.DA = DA;

}

public String getDetails() {

return super.getDetails() + ", Basic Pay: " + basicPay + ", DA: " + DA;

}

}

public class Main {

public static void main(String[] args) {

PartTimeEmp partTimeEmp = new PartTimeEmp(101, "John", 25, 20, 15.5);

FullTimeEmp fullTimeEmp = new FullTimeEmp(102, "Alice", 30, 50000, 10000);

System.out.println("Details of Part Time Employee:");

System.out.println(partTimeEmp.getDetails());

System.out.println("\nDetails of Full Time Employee:");

System.out.println(fullTimeEmp.getDetails());

// Polymorphism example

Emp emp1 = new PartTimeEmp(103, "Bob", 22, 15, 12.5);

Emp emp2 = new FullTimeEmp(104, "Eva", 28, 60000, 12000);

System.out.println("\nPolymorphism Example:");

System.out.println("Details of Employee 1:");

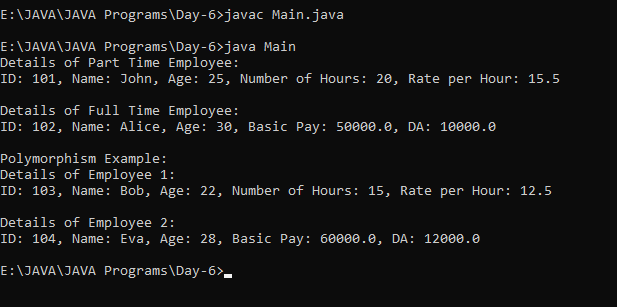
emp1.showDetails();

System.out.println("\nDetails of Employee 2:");

emp2.showDetails();

}

}



1. **In the day-5 program, make the Emp class as an abstract class. Add calculateSalary as abstract function. Try to create object of this class.**

abstract class Emp {

int id;

String name;

int age;

public Emp() {

this.id = 0;

this.name = "";

this.age = 0;

}

public Emp(int id, String name, int age) {

this.id = id;

this.name = name;

this.age = age;

}

public abstract double calculateSalary();

public String getDetails() {

return "ID: " + id + ", Name: " + name + ", Age: " + age;

}

public void showDetails() {

System.out.println(getDetails());

}

}

class PartTimeEmp extends Emp {

int numberOfHours;

double ratePerHour;

public PartTimeEmp() {

super();

this.numberOfHours = 0;

this.ratePerHour = 0.0;

}

public PartTimeEmp(int id, String name, int age, int numberOfHours, double ratePerHour) {

super(id, name, age);

this.numberOfHours = numberOfHours;

this.ratePerHour = ratePerHour;

}

public double calculateSalary() {

return numberOfHours \* ratePerHour;

}

public String getDetails() {

return super.getDetails() + ", Number of Hours: " + numberOfHours + ", Rate per Hour: " + ratePerHour;

}

}

class FullTimeEmp extends Emp {

double basicPay;

double DA;

public FullTimeEmp() {

super();

this.basicPay = 0.0;

this.DA = 0.0;

}

public FullTimeEmp(int id, String name, int age, double basicPay, double DA) {

super(id, name, age);

this.basicPay = basicPay;

this.DA = DA;

}

public double calculateSalary() {

return basicPay + DA;

}

public String getDetails() {

return super.getDetails() + ", Basic Pay: " + basicPay + ", DA: " + DA;

}

}

public class Abs {

public static void main(String[] args) {

PartTimeEmp partTimeEmp = new PartTimeEmp(101, "John", 25, 20, 15.5);

FullTimeEmp fullTimeEmp = new FullTimeEmp(102, "Alice", 30, 50000, 10000);

System.out.println("Details of Part Time Employee:");

System.out.println(partTimeEmp.getDetails());

System.out.println("Salary: " + partTimeEmp.calculateSalary());

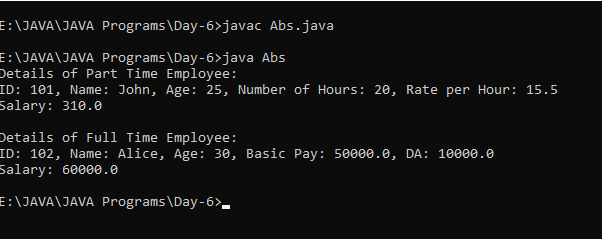
System.out.println("\nDetails of Full Time Employee:");

System.out.println(fullTimeEmp.getDetails());

System.out.println("Salary: " + fullTimeEmp.calculateSalary());

}

}



Day-7

1. **For the Student class(id,name,age) override toString and equals methods of object class. Test working of both the methods.**

import java.util.Objects;

class Student {

private int id;

private String name;

private int age;

public Student(int id, String name, int age) {

this.id = id;

this.name = name;

this.age = age;

}

public String toString() {

return "Student{" +

"id=" + id +

", name='" + name + '\'' +

", age=" + age +

'}';

}

public boolean equals(Object obj) {

if (this == obj) {

return true;

}

if (obj == null || getClass() != obj.getClass()) {

return false;

}

Student student = (Student) obj;

return id == student.id &&

age == student.age &&

Objects.equals(name, student.name);

}

public static void main(String[] args) {

Student student1 = new Student(1, "John", 20);

Student student2 = new Student(2, "Jane", 22);

Student student3 = new Student(1, "John", 20);

System.out.println("Student 1: " + student1);

System.out.println("Student 2: " + student2);

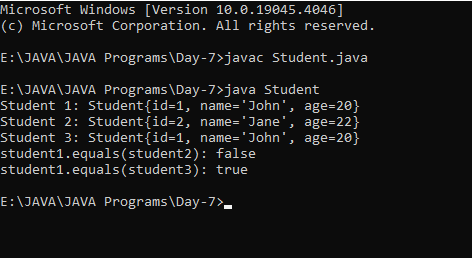
System.out.println("Student 3: " + student3);

System.out.println("student1.equals(student2): " + student1.equals(student2));

System.out.println("student1.equals(student3): " + student1.equals(student3));

}

}



1. **Create Interface Ione with class constant a having initial value 10.**

**Create Interface Itwo with class constant b having initial value 20.**

**Create Interface Ithree with class constant c having initial value 30.**

**Interface Ithree must inherit Ione & Itwo (multiple inheritance).**

**Crete class Test which implements Ithree. Test your program by creating an object of Test class and printing contents of a, b and c class constants.**

interface Ione{

int a=10;

}

interface Itwo{

int b=20;

}

interface Ithree extends Ione,Itwo {

int c=30;

}

class Test implements Ithree{

public static void main(String[] args){

Test t1=new Test();

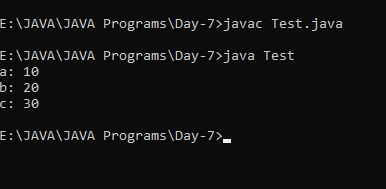
System.out.println("a: " + t1.a);

System.out.println("b: " + t1.b);

System.out.println("c: " + t1.c);

}

}

****

Day-8

**1 .rite a program which will handle the ArrayIndexOutOfBoundsException and**

**DivideByZeroException with multiple catch and finally block.**

class ArrayIndexOutOfBoundsException {

public static void main(String[] args) {

try {

int array[] = new int[10];

array[10] = 30 / 0;

array[20] = 40;

} catch (ArithmeticException e) {

System.out.println("An ArithmeticException occurred: " + e.getMessage());

} catch (ArrayIndexOutOfBoundsException e) {

System.out.println("An ArrayIndexOutOfBoundsException occurred: " + e.getMessage());

} finally {

System.out.println("This code will run no matter what.");

}

}

}

1. **For the Student class (id,name,age), write a program to raise an user define exception for the age less than 20 years.**

class InvalidAgeException extends Exception {

public InvalidAgeException(String message) {

super(message);

}

}

class Student {

private int id;

private String name;

private int age;

public Student(int id, String name, int age) throws InvalidAgeException {

this.id = id;

this.name = name;

if (age < 20) {

throw new InvalidAgeException("Age should be at least 20 years.");

} else {

this.age = age;

}

}

public static void main(String[] args) {

try {

Student student = new Student(1, "John Doe", 17);

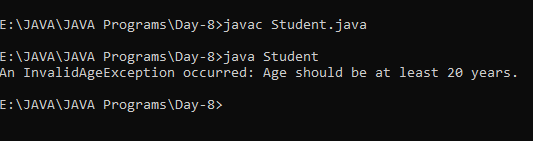
} catch (InvalidAgeException e) {

System.out.println("An InvalidAgeException occurred: " + e.getMessage());

}

}

}



Day-9

1. **With the help of Student(id,name,age) class do the followings:**
   1. **Create list of 5 students. Iterate from first to last student and display all the information about each student.**

import java.util.ArrayList;

import java.util.List;

class Student {

int id;

String name;

int age;

public Student(int id, String name, int age) {

this.id = id;

this.name = name;

this.age = age;

}

public String toString() {

return "Student[id=" + id + ", name=" + name + ", age=" + age + "]";

}

}

class Main {

public static void main(String[] args) {

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

students.add(new Student(1, "Hit Soliya", 21));

students.add(new Student(2, "Mohit Libachiya", 20));

students.add(new Student(3, "Wagha Kiran", 23));

students.add(new Student(4, "Gupta Vivek", 20));

students.add(new Student(5, "Patel Bhutik", 22));

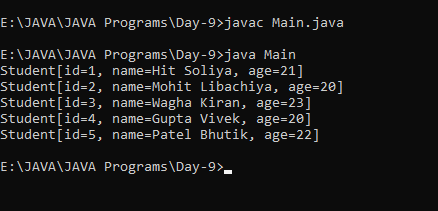
for (Student student : students) {

System.out.println(student);

}

}

}

****

* 1. **Create stack of 5 students. Iterate from first to last student and display all the information about each student.**

import java.util.Stack;

class Student {

int id;

String name;

int age;

public Student(int id, String name, int age) {

this.id = id;

this.name = name;

this.age = age;

}

public String toString() {

return "Student[id=" + id + ", name=" + name + ", age=" + age + "]";

}

}

class B {

public static void main(String[] args) {

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

students.push(new Student(1, "Hit Soliya", 21));

students.push(new Student(2, "Parmar Maulik", 20));

students.push(new Student(3, "Wagh Kiran", 23));

students.push(new Student(4, "Gupta Vivek", 20));

students.push(new Student(5, "Samir Prajapati", 22));

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

for (Student student : students) {

System.out.println(student);

}

System.out.println("Popped: " + students.pop());

System.out.println("Stack after popping: ");

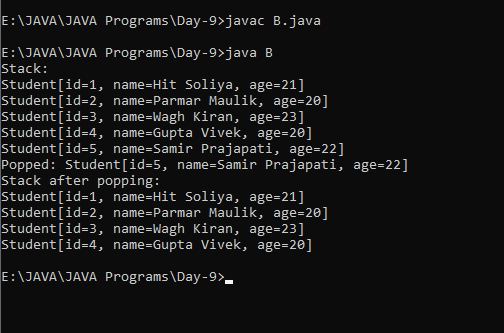
for (Student student : students) {

System.out.println(student);

}

}

}



* 1. **Create vector of 5 students. Iterate from first to last student and display all the information about each student.**

import java.util.Vector;

class Student {

int id;

String name;

int age;

public Student(int id, String name, int age) {

this.id = id;

this.name = name;

this.age = age;

}

public String toString() {

return "Student[id =" + id + ", name = " + name + ", age = " + age + "]";

}

}

class studentInfo {

public static void main(String[] args) {

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

students.add(new Student(1, "Hit Soliya", 21));

students.add(new Student(2, "Gupta Vivek", 20));

students.add(new Student(3, "Wagh Kiran", 23));

students.add(new Student(4, "Ketan Rathod", 21));

students.add(new Student(5, "Patel Bhutik", 20));

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

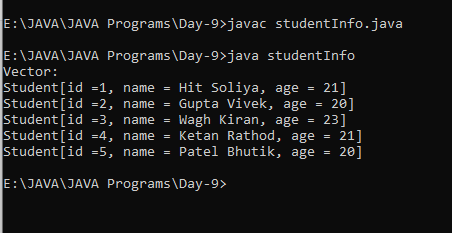
for (Student student : students) {

System.out.println(student);

}

}

}



Day-11

1. **Write a multi-threaded program in which two threads prints 1 to 5000 numbers by extending Thread class.**

class NumberPrinter extends Thread {

private final int start;

private final int end;

public NumberPrinter(int start, int end) {

this.start = start;

this.end = end;

}

public void run() {

for (int i = start; i <= end; i++) {

System.out.println(i);

}

}

public static void main(String[] args) {

int mid = 5000 / 2;

NumberPrinter thread1 = new NumberPrinter(1, mid + 1);

NumberPrinter thread2 = new NumberPrinter(mid + 2, 5000);

thread1.start();

thread2.start();

try {

thread1.join();

thread2.join();

} catch (InterruptedException e) {

e.printStackTrace();

}

System.out.println("All numbers printed!");

}

}



**2.Write a multi-threaded program in which two threads prints 1 to 5000 numbers by implementing Runnable interface.**

class NumPrint implements Runnable {

private final int start;

private final int end;

public NumPrint(int start, int end) {

this.start = start;

this.end = end;

}

public void run() {

for (int i = start; i <= end; i++) {

System.out.println(i);

}

}

public static void main(String[] args) {

int mid = 5000 / 2;

NumPrint printer1 = new NumPrint(1, mid);

NumPrint printer2 = new NumberPrinter(mid + 1, 5000);

Thread thread1 = new Thread(printer1);

Thread thread2 = new Thread(printer2);

thread1.start();

thread2.start();

try {

thread1.join();

thread2.join();

} catch (InterruptedException e) {

e.printStackTrace();

}

System.out.println("All numbers printed!");

}

}



Day-12

1. **Write a multi-threaded program in which two threads access the common variable count which is initialize with 0. Both the read this count and print on console along with thread name. After printing, it will increment the content of this count variable. Both thread will do this work for 5000 times. Use the synchronization for count increment logic.**

class CountPrinter implements Runnable {

private static int count;

public void run() {

for (int i = 0; i < 5000; i++) {

synchronized (CountPrinter.class) { // Use class as lock directly int currentCount = count;

System.out.println(Thread.currentThread().getName() + ": " + currentCount);

count++;

}

}

}

public static void main(String[] args) {

CountPrinter printer = new CountPrinter();

Thread thread1 = new Thread(printer);

Thread thread2 = new Thread(printer);

thread1.start();

thread2.start();

try {

thread1.join();

thread2.join();

} catch (InterruptedException e) {

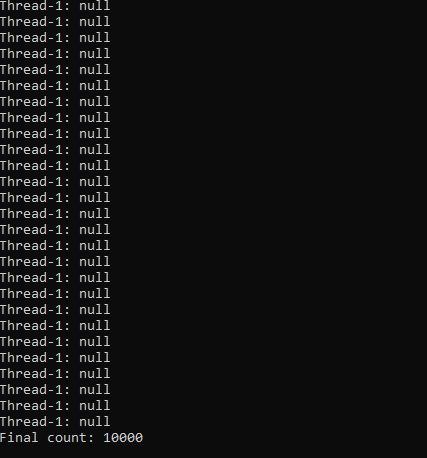
e.printStackTrace();

}

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

}

}



1. **Write a program to create a new folder, rename and delete it.**

import java.io.File;

class FolderOperations {

public static void main(String[] args) {

String folderName = "NewFolder";

File folder = new File(folderName);

// Create the folder

boolean folderCreated = folder.mkdir();

if (folderCreated) {

System.out.println("Folder created successfully.");

} else {

System.out.println("Failed to create folder.");

return;

}

// Rename the folder

String newFolderName = "RenamedFolder";

File newFolder = new File(newFolderName);

boolean renamed = folder.renameTo(newFolder);

if (renamed) {

System.out.println("Folder renamed successfully.");

} else {

System.out.println("Failed to rename folder.");

return;

}

// Delete the folder

boolean deleted = newFolder.delete();

if (deleted) {

System.out.println("Folder deleted successfully.");

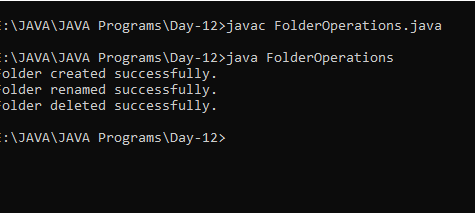
} else {

System.out.println("Failed to delete folder.");

}

}

}



Day-13

**1. Write a program to copy one file to other file using:**

**1) Character by character**

**2) Word by word**

**3) Line by line**

1. **Character by character**

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

public class CopyFileCharacterByCharacter {

public static void main(String[] args) {

try (FileReader reader = new FileReader("source.txt");

FileWriter writer = new FileWriter("destination\_character.txt")) {

int character;

while ((character = reader.read()) != -1) {

writer.write(character);

}

System.out.println("File copied character by character successfully.");

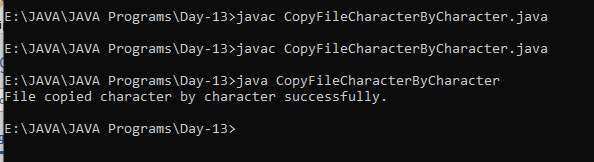
} catch (IOException e) {

e.printStackTrace();

}

}

}



1. **Word by word**

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

import java.util.Scanner;

public class CopyFileWordByWord {

public static void main(String[] args) {

try (Scanner scanner = new Scanner(new FileReader("source.txt"));

FileWriter writer = new FileWriter("destination\_word.txt")) {

while (scanner.hasNext()) {

String word = scanner.next();

writer.write(word + " ");

}

System.out.println("File copied word by word successfully.");

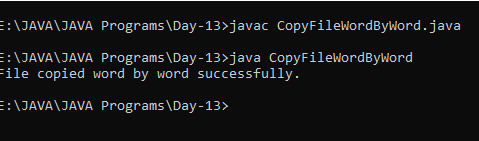
} catch (IOException e) {

e.printStackTrace();

}

}

}



**3) Line by line**

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

public class CopyFileLineByLine {

public static void main(String[] args) {

try (BufferedReader reader = new BufferedReader(new FileReader("source.txt"));

FileWriter writer = new FileWriter("destination\_line.txt")) {

String line;

while ((line = reader.readLine()) != null) {

writer.write(line + "\n");

}

System.out.println("File copied line by line successfully.");

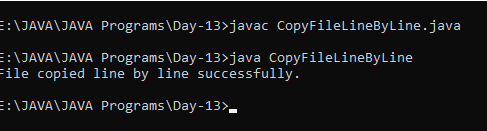
} catch (IOException e) {

e.printStackTrace();

}

}

}



**2. Write the program to marge two files in to third file using any of the above approaches.**

1. **Character by character**

import java.io.\*;

public class MergeFilesCharacterByCharacter {

public static void main(String[] args) {

try (FileReader reader1 = new FileReader("file1.txt");

FileReader reader2 = new FileReader("file2.txt");

FileWriter writer = new FileWriter("merged\_character.txt")) {

int character;

while ((character = reader1.read()) != -1) {

writer.write(character);

}

while ((character = reader2.read()) != -1) {

writer.write(character);

}

System.out.println("Files merged character by character successfully.");

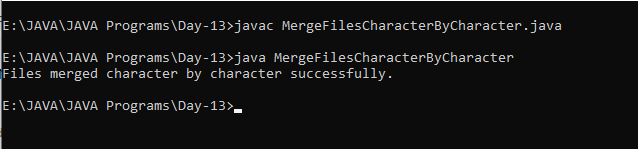
} catch (IOException e) {

e.printStackTrace();

}

}

}



1. **Word by word**

import java.io.\*;

import java.util.Scanner;

public class MergeFilesWordByWord {

public static void main(String[] args) {

try (Scanner scanner1 = new Scanner(new FileReader("file1.txt"));

Scanner scanner2 = new Scanner(new FileReader("file2.txt"));

FileWriter writer = new FileWriter("merged\_word.txt")) {

while (scanner1.hasNext()) {

String word = scanner1.next();

writer.write(word + " ");

}

while (scanner2.hasNext()) {

String word = scanner2.next();

writer.write(word + " ");

}

System.out.println("Files merged word by word successfully.");

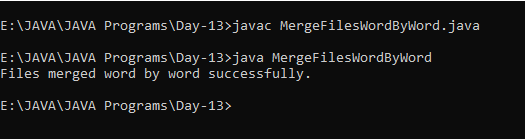
} catch (IOException e) {

e.printStackTrace();

}

}

}



1. **Line by line**

import java.io.\*;

public class MergeFilesLineByLine {

public static void main(String[] args) {

try (BufferedReader reader1 = new BufferedReader(new FileReader("file1.txt"));

BufferedReader reader2 = new BufferedReader(new FileReader("file2.txt"));

FileWriter writer = new FileWriter("merged\_line.txt")) {

String line;

while ((line = reader1.readLine()) != null) {

writer.write(line + "\n");

}

while ((line = reader2.readLine()) != null) {

writer.write(line + "\n");

}

System.out.println("Files merged line by line successfully.");

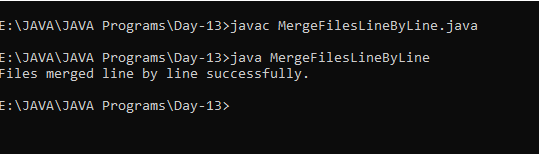
} catch (IOException e) {

e.printStackTrace();

}

}

}



Day-14

1. **Create Student class(Id,name,age). Put appropriate constructor to initialize all the data member of the**

**class. Write 5 objects to the file. Id of the student must be 1,2,3,4 and 5. User will provide the Id of**

**the student, and your program will display the information of that student.**

import java.io.\*;

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

class Student implements Serializable {

private static final long serialVersionUID = 1L;

private int id;

private String name;

private int age;

// Constructor

public Student(int id, String name, int age) {

this.id = id;

this.name = name;

this.age = age;

}

// Getters

public int getId() {

return id;

}

public String getName() {

return name;

}

public int getAge() {

return age;

}

}

public class Main {

public static void main(String[] args) {

// Create a list of students

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

students.add(new Student(1, "Alice", 20));

students.add(new Student(2, "Bob", 21));

students.add(new Student(3, "Charlie", 22));

students.add(new Student(4, "David", 23));

students.add(new Student(5, "Eve", 24));

// Write objects to file

try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("students.dat"))) {

for (Student student : students) {

oos.writeObject(student);

}

System.out.println("Objects written to file successfully.");

} catch (IOException e) {

e.printStackTrace();

}

// Retrieve information of a specific student by ID

try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream("students.dat"));

Scanner scanner = new Scanner(System.in)) {

System.out.print("Enter the ID of the student: ");

int id = scanner.nextInt();

boolean found = false;

while (true) {

try {

Student student = (Student) ois.readObject();

if (student.getId() == id) {

System.out.println("Student found:");

System.out.println("ID: " + student.getId());

System.out.println("Name: " + student.getName());

System.out.println("Age: " + student.getAge());

found = true;

break;

}

} catch (EOFException e) {

break;

}

}

if (!found) {

System.out.println("Student with ID " + id + " not found.");

}

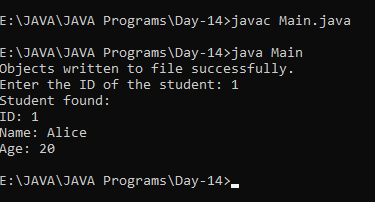
} catch (IOException | ClassNotFoundException e) {

e.printStackTrace();

}

}

}



­­­­

Day-15

1. **Create Student class(Id,name,age). Put appropriate constructor to initialize all the data member of the**

**class. Write 5 objects to the file. Id of the student must be 1,2,3,4 and 5. User will provide the Id of**

**the student, and your program will display the information of that student.**

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

public class Main {

public static void main(String[] args) {

JFrame frame = new JFrame("Event Handling Example");

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setSize(300, 200);

frame.setLayout(new FlowLayout());

JButton button = new JButton("Click Me");

// Registering listeners

frame.addMouseListener(new MouseListener() {

@Override

public void mouseClicked(MouseEvent e) {

System.out.println("Mouse Clicked on JFrame");

}

@Override

public void mousePressed(MouseEvent e) {}

@Override

public void mouseReleased(MouseEvent e) {}

@Override

public void mouseEntered(MouseEvent e) {}

@Override

public void mouseExited(MouseEvent e) {}

});

frame.addMouseMotionListener(new MouseMotionListener() {

@Override

public void mouseDragged(MouseEvent e) {}

@Override

public void mouseMoved(MouseEvent e) {

System.out.println("Mouse Moved on JFrame");

}

});

frame.addKeyListener(new KeyListener() {

@Override

public void keyTyped(KeyEvent e) {}

@Override

public void keyPressed(KeyEvent e) {

System.out.println("Key Pressed on JFrame: " + e.getKeyChar());

}

@Override

public void keyReleased(KeyEvent e) {}

});

frame.addWindowListener(new WindowListener() {

@Override

public void windowOpened(WindowEvent e) {}

@Override

public void windowClosing(WindowEvent e) {

System.out.println("Window Closing");

}

@Override

public void windowClosed(WindowEvent e) {}

@Override

public void windowIconified(WindowEvent e) {}

@Override

public void windowDeiconified(WindowEvent e) {}

@Override

public void windowActivated(WindowEvent e) {}

@Override

public void windowDeactivated(WindowEvent e) {}

});

button.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

System.out.println("Button Clicked");

}

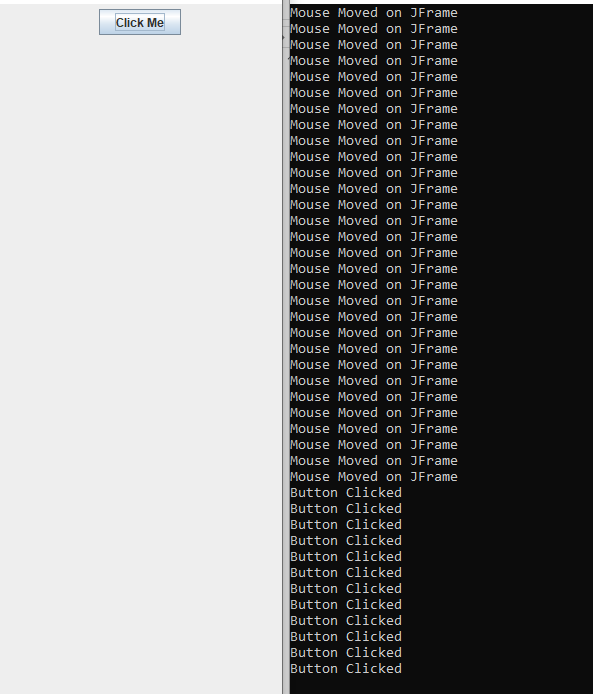
});

frame.add(button);

frame.setVisible(true);

}

}



Day-16

1. **Demonstrate the use of FlowLayout, BorderLayout, GridLayout, GridBagLayout & CardLayout.**

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class LayoutManagerDemo {

public static void main(String[] args) {

JFrame frame = new JFrame("Layout Manager Demo");

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setSize(400, 300);

// FlowLayout

JPanel flowLayoutPanel = new JPanel(new FlowLayout());

flowLayoutPanel.add(new JButton("Button 1"));

flowLayoutPanel.add(new JButton("Button 2"));

flowLayoutPanel.add(new JButton("Button 3"));

frame.add(flowLayoutPanel, BorderLayout.NORTH);

// BorderLayout

JPanel borderLayoutPanel = new JPanel(new BorderLayout());

borderLayoutPanel.add(new JButton("Button 4 (North)"), BorderLayout.NORTH);

borderLayoutPanel.add(new JButton("Button 5 (Center)"), BorderLayout.CENTER);

borderLayoutPanel.add(new JButton("Button 6 (South)"), BorderLayout.SOUTH);

frame.add(borderLayoutPanel, BorderLayout.CENTER);

// GridLayout

JPanel gridLayoutPanel = new JPanel(new GridLayout(2, 3));

gridLayoutPanel.add(new JButton("Button 7"));

gridLayoutPanel.add(new JButton("Button 8"));

gridLayoutPanel.add(new JButton("Button 9"));

gridLayoutPanel.add(new JButton("Button 10"));

gridLayoutPanel.add(new JButton("Button 11"));

gridLayoutPanel.add(new JButton("Button 12"));

frame.add(gridLayoutPanel, BorderLayout.SOUTH);

// GridBagLayout

JPanel gridBagLayoutPanel = new JPanel(new GridBagLayout());

GridBagConstraints gbc = new GridBagConstraints();

gbc.gridx = 0;

gbc.gridy = 0;

gbc.insets = new Insets(5, 5, 5, 5);

gridBagLayoutPanel.add(new JButton("Button 13"), gbc);

gbc.gridx = 1;

gridBagLayoutPanel.add(new JButton("Button 14"), gbc);

gbc.gridy = 1;

gbc.gridx = 0;

gridBagLayoutPanel.add(new JButton("Button 15"), gbc);

gbc.gridx = 1;

gridBagLayoutPanel.add(new JButton("Button 16"), gbc);

frame.add(gridBagLayoutPanel, BorderLayout.EAST);

// CardLayout

JPanel cardLayoutPanel = new JPanel(new CardLayout());

JButton card1 = new JButton("Card 1");

JButton card2 = new JButton("Card 2");

JButton card3 = new JButton("Card 3");

cardLayoutPanel.add(card1, "card1");

cardLayoutPanel.add(card2, "card2");

cardLayoutPanel.add(card3, "card3");

Timer timer = new Timer(1000, new ActionListener() {

int count = 1;

@Override

public void actionPerformed(ActionEvent e) {

CardLayout layout = (CardLayout) cardLayoutPanel.getLayout();

layout.show(cardLayoutPanel, "card" + (count % 3 + 1));

count++;

}

});

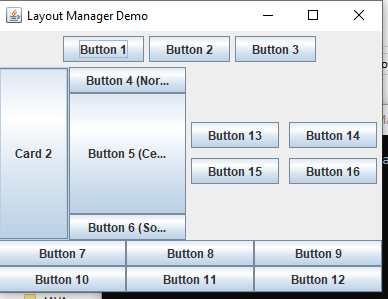
timer.start();

frame.add(cardLayoutPanel, BorderLayout.WEST);

frame.setVisible(true);

}

}



**2. Demonstrate working of JTextField, JTextArea with event handling.**

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class TextFieldAndTextAreaDemo extends JFrame implements ActionListener {

JTextField textField;

JTextArea textArea;

public TextFieldAndTextAreaDemo() {

setTitle("TextField and TextArea Demo");

setSize(400, 300);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

// Create a panel to hold components

JPanel panel = new JPanel();

panel.setLayout(new BorderLayout());

// Create a text field

textField = new JTextField(20);

textField.addActionListener(this); // Register ActionListener

panel.add(textField, BorderLayout.NORTH);

// Create a text area

textArea = new JTextArea();

JScrollPane scrollPane = new JScrollPane(textArea);

panel.add(scrollPane, BorderLayout.CENTER);

// Add panel to the frame

add(panel);

setVisible(true);

}

@Override

public void actionPerformed(ActionEvent e) {

// Get the text from the text field and append it to the text area

String text = textField.getText();

textArea.append(text + "\n");

// Clear the text field

textField.setText("");

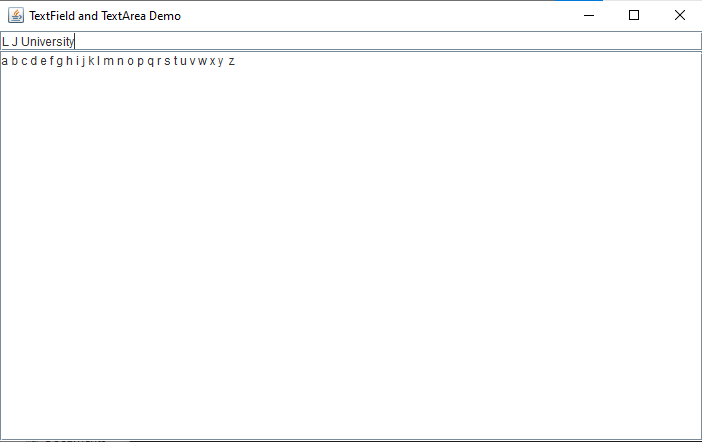
}

public static void main(String[] args) {

SwingUtilities.invokeLater(() -> new TextFieldAndTextAreaDemo());

}

}



Day-17

1. **Create GUI based application for storing student’s information like (id,name,age,email,sem). Do the**

**validation for valid email address. Form must have Submit and Clear button. On clicking on Submit**

**button, all the information filled must be displayed by appropriate dialog box.**

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

public class StudentInformationApp extends JFrame implements ActionListener {

JTextField idField, nameField, ageField, emailField, semField;

public StudentInformationApp() {

setTitle("Student Information App");

setSize(400, 300);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

JPanel panel = new JPanel();

panel.setLayout(new GridLayout(6, 2, 5, 5));

panel.add(new JLabel("ID:"));

idField = new JTextField();

panel.add(idField);

panel.add(new JLabel("Name:"));

nameField = new JTextField();

panel.add(nameField);

panel.add(new JLabel("Age:"));

ageField = new JTextField();

panel.add(ageField);

panel.add(new JLabel("Email:"));

emailField = new JTextField();

panel.add(emailField);

panel.add(new JLabel("Semester:"));

semField = new JTextField();

panel.add(semField);

JButton submitButton = new JButton("Submit");

submitButton.addActionListener(this);

panel.add(submitButton);

JButton clearButton = new JButton("Clear");

clearButton.addActionListener(this);

panel.add(clearButton);

add(panel);

setVisible(true);

}

@Override

public void actionPerformed(ActionEvent e) {

if (e.getActionCommand().equals("Submit")) {

if (validateEmail()) {

String id = idField.getText();

String name = nameField.getText();

String age = ageField.getText();

String email = emailField.getText();

String sem = semField.getText();

String message = "Student Information:\n"

+ "ID: " + id + "\n"

+ "Name: " + name + "\n"

+ "Age: " + age + "\n"

+ "Email: " + email + "\n"

+ "Semester: " + sem;

JOptionPane.showMessageDialog(this, message, "Student Information", JOptionPane.INFORMATION\_MESSAGE);

} else {

JOptionPane.showMessageDialog(this, "Invalid email address!", "Error", JOptionPane.ERROR\_MESSAGE);

}

} else if (e.getActionCommand().equals("Clear")) {

clearFields();

}

}

private boolean validateEmail() {

String email = emailField.getText();

String emailRegex = "^[a-zA-Z0-9\_+&\*-]+(?:\\.[a-zA-Z0-9\_+&\*-]+)\*@(?:[a-zA-Z0-9-]+\\.)+[a-zA-Z]{2,7}$";

Pattern pattern = Pattern.compile(emailRegex);

Matcher matcher = pattern.matcher(email);

return matcher.matches();

}

private void clearFields() {

idField.setText("");

nameField.setText("");

ageField.setText("");

emailField.setText("");

semField.setText("");

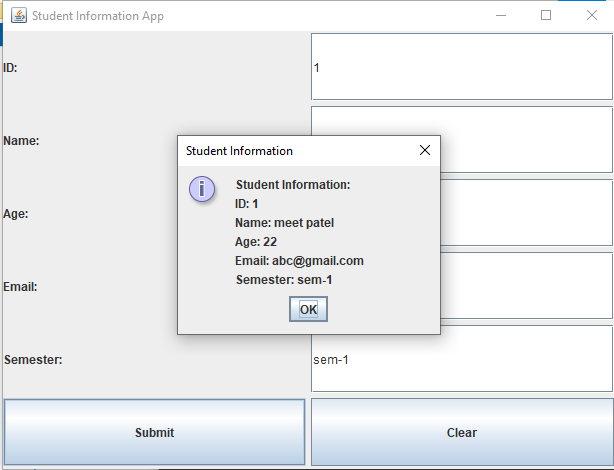
}

public static void main(String[] args) {

SwingUtilities.invokeLater(StudentInformationApp::new);

}

}



**2. Demonstrate working Convert the above program with following menus:**

**a. Operations : New, Close, Exit**

**b. Help: About Application**

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

public class StudentInformationAppWithMenus extends JFrame implements ActionListener {

JTextField idField, nameField, ageField, emailField, semField;

public StudentInformationAppWithMenus() {

setTitle("Student Information App");

setSize(400, 300);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

JMenuBar menuBar = new JMenuBar();

setJMenuBar(menuBar);

JMenu operationsMenu = new JMenu("Operations");

menuBar.add(operationsMenu);

JMenuItem newMenuItem = new JMenuItem("New");

newMenuItem.addActionListener(this);

operationsMenu.add(newMenuItem);

JMenuItem closeMenuItem = new JMenuItem("Close");

closeMenuItem.addActionListener(this);

operationsMenu.add(closeMenuItem);

JMenuItem exitMenuItem = new JMenuItem("Exit");

exitMenuItem.addActionListener(this);

operationsMenu.add(exitMenuItem);

JMenu helpMenu = new JMenu("Help");

menuBar.add(helpMenu);

JMenuItem aboutMenuItem = new JMenuItem("About Application");

aboutMenuItem.addActionListener(this);

helpMenu.add(aboutMenuItem);

JPanel panel = new JPanel();

panel.setLayout(new GridLayout(6, 2, 5, 5));

panel.add(new JLabel("ID:"));

idField = new JTextField();

panel.add(idField);

panel.add(new JLabel("Name:"));

nameField = new JTextField();

panel.add(nameField);

panel.add(new JLabel("Age:"));

ageField = new JTextField();

panel.add(ageField);

panel.add(new JLabel("Email:"));

emailField = new JTextField();

panel.add(emailField);

panel.add(new JLabel("Semester:"));

semField = new JTextField();

panel.add(semField);

add(panel);

setVisible(true);

}

@Override

public void actionPerformed(ActionEvent e) {

String command = e.getActionCommand();

switch (command) {

case "New":

clearFields();

break;

case "Close":

dispose();

break;

case "Exit":

System.exit(0);

break;

case "About Application":

JOptionPane.showMessageDialog(this, "Student Information App\nVersion 1.0\nDeveloped by Your Name",

"About", JOptionPane.INFORMATION\_MESSAGE);

break;

}

}

private void clearFields() {

idField.setText("");

nameField.setText("");

ageField.setText("");

emailField.setText("");

semField.setText("");

}

public static void main(String[] args) {

SwingUtilities.invokeLater(StudentInformationAppWithMenus::new);

}

}

