**Name : Rajkumar A Shamal**

**Roll no. : 45**

**Div : B**

**Subject : java**

**Unit – 1**

**Day-01**

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

class HelloWorld

{

public static void main (String args[])

{

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

} }

**OUTPUT :**

Hello World

2.Write comments (Author, date, objective of the program, main function) in the above program. class SumCalculator

{

public static void main(String[] args)

{

int number1, number2, sum; number1 = 5; number2 = 10;

sum = number1 + number2;

System.out.println("The sum of " + number1 + " and " + number2 + " is: " + sum);

} }

OUTPUT :

The sum of 5 and 10 is: 15

3. 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.

class Array

{

public static void main(String[] args)

{

int[][] array1 = new int[2][3]; int counter = 1; for (int i = 0; i < array1.length; i++)

{

for (int j = 0; j < array1[i].length; j++)

{

array1[i][j] = counter++;

}

}

int[][] array2 = new int[2][]; array2[0] = new int[3]; array2[1] = new int[2]; counter = 1; for (int i = 0; i < array2.length; i++)

{

for (int j = 0; j < array2[i].length; j++)

{

array2[i][j] = counter++;

}

}

for (int i = 0; i < array1.length; i++)

{

for (int j = 0; j < array1[i].length; j++)

{

System.out.print(array1[i][j] + " ");

}

System.out.println();

}

System.out.println(); for (int i = 0; i < array2.length; i++)

{

for (int j = 0; j < array2[i].length; j++)

{

System.out.print(array2[i][j] + " ");

}

System.out.println();

}

}

}

OUTPUT :

1 2 3

4 5 6

1 2 3

4 5

1. Make a small report of 3 pages on difference between C & Java.

|  |  |
| --- | --- |
| C | JAVA |
| 1. C was developed by Dennis M.  Ritchie between 1969 and 1973. | 1. Java was developed by James  Gosling in 1995. |
| 2. C is a Procedural Programming  Language. | 2. Java is Object-Oriented language. |
| 3. C is more procedure-oriented. | 3. Java is more data-oriented. |
| 4.C is a middle-level language because binding of the gaps takes place between machine level language and high-level languages. | 4. Java is a high-level language because translation of code takes place into machine language using compiler or interpreter. |
| 5.C is a compiled language that is it converts the code into machine language so that it could be understood by the machine or system. | 5. Java is an Interpreted language that is in Java, the code is first transformed into bytecode and that bytecode is then executed by the JVM (JavaVirtual Machine). |
| 6.C generally breaks down into functions. | 6. Java breaks down into Objects. |
| 7. C programming language can be used for system programming as well as Application programming. | 7. This is not the case in Java. |
| 8. Memory allocation can be done by malloc in C | 8. Memory allocation can be done by a new keyword in Java. |
| 9. C is a low-level language. It has difficult interpretation for the user but it has a closer significance to the machine-level code. | 9. Java is a high-level language because translation of code takes place into machine language using compiler or interpreter. |

|  |  |
| --- | --- |
| 10. free is used for freeing the memory in C. | 10. A compiler will free up the memory internally by calling the garbage collector. |
| 11. Java supports the concept of threading. | 11. Java supports the concept of threading. |
| 12. Java does not supports pointers. | 12. Java does not supports pointers. |
| 13. It is portable. | 13. It is portable. |
| 14. It only supports a call by value. | 14. It only supports a call by value. |
| 15. Java is a platform independent. | 15. Java is a platform independent. |
| 16. It internally manages the memory. | 16. It internally manages the memory. |
| 17. Java is robust. | 17. Java is robust. |
| 18. Java follows a bottom-up approach. | 18. Java follows a bottom-up approach. |
| 19. Java supports method overloading which helps in code readability. | 19. Java supports method overloading which helps in code readability. |
| 20. Java does not support  Preprocessors. | 20. Java does not support  Preprocessors. |

1. Write a program to print your name and address.

class NameAndAddress

{

public static void main(String[] args)

{

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

System.out.println("Address: 123 Main Street, makarba, Ahemdabad");

}

}

OUTPUT :

Name: Liya

Address: 123 Main Street, makarba, Ahemdabad

# Day-02

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

public class pro1

{

public static void main(String[] args)

{

int num1 = 10; int num2 = 5; int sum = num1 + num2;

System.out.println("The sum is: " + sum);

}

}

output:

The sum is: 15

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

import java.util.Scanner; public class pro2

{

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();

int sum = num1 + num2;

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

scanner.close();

}

}

output:

Enter the first number: 20

Enter the second number: 30

The sum of 20 and 30 is 50

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

public class pro3

{

public static void main(String[] args)

{

int sum = 0; for (int i = 3; i <= 10; i++)

{

sum += i;

}

System.out.println("The sum of numbers between 3 to 10 is: " + sum);

}

}

output:

The sum of numbers between 3 to 10 is: 52

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

public class pro4

{

public static void main(String[] args)

{

int num1 = 10; int num2 = 25;

int maximum = Math.max(num1, num2);

System.out.println("The maximum number is: " + maximum);

}

}

output:

The maximum number is: 25

/\*5. 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);

}

}

\*/

public class pro5

{

public static void main(String[] args)

{

outer:

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

{

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

{

if (j == 3) break outer;

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

}

}

}

}

output: value of j = 0 value of j = 1 value of j = 2 /\*

6.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.Scanner; class Student

{

private String name; private int age; public void getDetails()

{

Scanner scanner = new Scanner(System.in); System.out.print("Enter name of the student: "); name = scanner.nextLine(); System.out.print("Enter age of the student: "); age = scanner.nextInt();

}

public void displayDetails()

{

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

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

}

}

class pro6

{

public static void main(String[] args)

{

Student student1 = new Student();

Student student2 = new Student();

Student student3 = new Student();

student1.getDetails(); student1.displayDetails(); student2.getDetails(); student2.displayDetails();

student3.getDetails(); student3.displayDetails();

}

}

output:

Enter name of the student: urmi

Enter age of the student: 21

Name: urmi

Age: 21

Enter name of the student: suhani

Enter age of the student: 21

Name: suhani

Age: 21

Enter name of the student: tejasvi

Enter age of the student: 21

Name: tejasvi

Age: 21

# Day-03

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.

|  |
| --- |
| class Student { |
| private String name; |
| private int age; |
|  |
| // Default constructor |
| public Student() { |
| this.name = ""; |
| this.age = 0; |
| } |
|  |
| // Parameterized constructor |
| public Student(String name, int age) { |
| this.name = name; |
| this.age = age; |
| } |
|  |
| // Copy constructor |
| public Student(Student otherStudent) { |
| this.name = otherStudent.name; |
| this.age = otherStudent.age; |
| } |
|  |
| // Getters and setters |
| public String getName() { |
| return name; |
| } |
|  |
| public int getAge() { |
| return age; |
| } |
|  |
| public void setName(String name) { |
| this.name = name; |
| } |
|  |
| public void setAge(int age) { |
| this.age = age; |
| } |
| } |
|  |
| class Stud{ |
| public static void main(String[] args) { |
| // Default constructor |
| Student student1 = new Student(); |
| student1.setName("riya"); |
| student1.setAge(20); |
| System.out.println("Student 1 - Name: " + student1.getName() + ", Age: " + |
| student1.getAge()); |
|  |
| // Parameterized constructor |
| Student student2 = new Student("ronak", 22); |
| System.out.println("Student 2 - Name: " + student2.getName() + ", Age: " + |
| student2.getAge()); |
|  |
| // Copy constructor |
| Student student3 = new Student(student1); |
| student3.setName("diya"); |
| System.out.println("Student 3 - Name: " + student3.getName() + ", Age: " + |
| student3.getAge()); |
| } |
| } |
|  |
| /\* |
| Student 1 - Name: riya, Age: 20 |
| Student 2 - Name: ronak, Age: 22 |
| Student 3 - Name: diya, Age: 20 |
| \*/ |

1. Modify the above program to jump from one constructor to other constructor.

|  |
| --- |
| class Program { |
| private String name; private int age; |
|  |
| public Program() |
| { |
| this("mina shah"); |
| } |
| public Program(String name) |
| { |
| this(name, 18); |
| } |
|  |
| public Program(String name, int age) |
| {  this.name = name; |
| this.age = age; |
| } |
|  |
| public static void main(String[] args) |
| { |
| Program program = new Program();  System.out.println("Name: " + program.getName()); |
| System.out.println("Age: " + program.getAge()); |
| } |
|  |
| public String getName() |
| { |
| return name; |
| } |
| public int getAge() |
| { |
| return age; |
| } |
| } |
| /\* |
| Name: mina shah |
| Age: 18 |
| \*/ |

1. Add initializer and static block in the above program.

|  |
| --- |
| class MyClass { |
| int x; |
|  |
| static |
| {  System.out.println("This is a static block."); |
| int x = 5; |
| } |
|  |
| { |
| System.out.println("This is an instance initializer."); |
| } |
| public MyClass() |
| { |
| System.out.println("This is the constructor."); |
| } |
|  |
| public static void main(String[] args)  { |
| MyClass myObj = new MyClass(); |
| } |
| } |
|  |
| /\* |
| This is a static block.  This is an instance initializer. |
| This is the constructor. |
| \*/ |

# Day-04

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.

|  |
| --- |
| class Student |
| { |
| private String name; |
| private int age; |
| public Student(String name, int age) |
| { |
| this.name = name; |
| this.age = age; |
| } |
|  |
| public int getAge()  { |
| return age; |
| } |
|  |
| public String getName() |
| { |
| return name;  } |
|  |
| public void compareAges(Student student1, Student student2) |
| { |
| if (student1.getAge() > student2.getAge()) { |
| System.out.println(student1.getName() + " has the highest age."); |
| } else if (student2.getAge() > student1.getAge()) { |
| System.out.println(student2.getName() + " has the highest age.");  } else { |
| System.out.println("Both students have the same age."); |
| } |
| } |
|  |
| public static void main(String[] args) |
| {  Student s1 = new Student("urmi", 21); |
| Student s2 = new Student("suhani", 20); |
| Student s3 = new Student("tejasvi", 19); |
|  |
| s1.compareAges(s2,s3); |
| } |
| } |
| /\* |
| output: |
| suhani has the highest age. |
|  |
| \*/ |

1. 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 |
| { |
| private String name; |
| private int age; |
|  |
| public Student(String name, int age)  { |
| this.name = name; |
| this.age = age; |
| } |
|  |
| public static Student getOldestStudent(Student[] students) |
| {  if (students.length == 0) |
| { |
| return null; |
| } |
|  |
| Student oldestStudent = students[0]; |
| for (int i = 1; i < students.length; i++) |
| { |
| if (students[i].getAge() > oldestStudent.getAge()) |
| { |
| oldestStudent = students[i]; |
| } |
| } |
| return oldestStudent; |
| } |
|  |
| public String getName() |
| { |
| return name; |
| } |
| public int getAge() |
| { |
| return age; |
| } |
| } |
| class pro2 |
| { |
| public static void main(String[] args) |
| { |
| Student[] students = new Student[3]; |
| students[0] = new Student("urmi", 21); students[1] = new Student("suhani", 18); |
| students[2] = new Student("tejasvi", 19); |
|  |
| Student oldestStudent = Student.getOldestStudent(students); |
|  |
| if (oldestStudent != null) |
| { |
| System.out.println("The student with the highest age is: " + oldestStudent.getName()); |
| } |
| else |
| { |
| System.out.println("No students found."); |
| } |
| }  } |
|  |
| /\* |
| output: |
|  |
| The student with the highest age is: urmi |
| \*/ |

1. Create a Calculator class to add two int and to add two floats.

Use method overloading.

class Calculator

|  |
| --- |
| { |
|  |
| public int add(int num1, int num2) |
| {  return num1 + num2; |
| } |
|  |
| public float add(float num1, float num2) |
| { |
| return num1 + num2; |
| } |
| public static void main(String[] args) |
| { |
| Calculator calculator = new Calculator(); |
|  |
| int result1 = calculator.add(5, 10); |
| System.out.println("Addition of 5 and 10 (int): " + result1); |
| float result2 = calculator.add(5.5f, 10.5f); |
| System.out.println("Addition of 5.5 and 10.5 (float): " + result2); |
| } |
|  |
| } |
|  |
| /\* output: |
|  |
| Addition of 5 and 10 (int): 15 |
| Addition of 5.5 and 10.5 (float): 16.0 |
|  |
| \*/ |

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 Student

|  |
| --- |
| { |
| private static int count = 0; |
| private String name; |
| public Student(String name) |
| { |
| this.name = name; |
| count++; |
| } |
|  |
| public static void displayCount()  { |
| System.out.println("Number of objects created: " + count); |
| } |
|  |
| public static void main(String[] args) { |
| Student s1 = new Student("ayushi"); |
| Student s2 = new Student("suhani"); |
| Student s3 = new Student("tejasvi"); |
| Student.displayCount(); |
| } |
| } |
|  |
| /\* |
| output: |
| Number of objects created: 3 |
|  |
| \*/ |

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 OuterClass |
| { |
| int a; |
|  |
| public OuterClass(int a) |
| {  this.a = a; |
| } |
|  |
| public void printValues() |
| { |
| System.out.println("Value of a: " + a); |
| InnerClass inner = new InnerClass(5); inner.printValues(); |
| } |
|  |
| class InnerClass |
| { |
| int b; |
| public InnerClass(int b) |
| { |
| this.b = b; |
| } |
|  |
| public void printValues() |
| {  System.out.println("Value of b: " + b); |
| } |
| } |
|  |
| public static void main(String[] args) |
| { |
| OuterClass outer = new OuterClass(10); outer.printValues(); |
| } |
| } |
|  |
| /\* |
| output: |
| Value of a: 10 |
| Value of b: 5  \*/ |

**Unit – 2**

**Day-05**

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 |
| { |
| protected int id; |
| protected String name; |
| protected 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 |
| {  private int numberOfHours; |
| private 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; |
| } |
|  |
| @Override |
| public String getDetails() |
| { |
| return super.getDetails() + ", NumberOfHours: " + numberOfHours + ",  RatePerHour: " + ratePerHour; |
| } |
| } |
|  |
| class FullTimeEmp extends Emp |
| { |
| private double basicPay; private 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; |
| } |
|  |
| @Override |
| public String getDetails() |
| { |
| return super.getDetails() + ", BasicPay: " + basicPay + ", DA: " + DA;  } |
| } |
|  |
| class pro1 |
| { |
| public static void main(String[] args) |
| { |
| PartTimeEmp partTimeEmp = new PartTimeEmp(1, "mahi", 25, 20, 15.0);  System.out.println("Part Time Employee Details:"); |
| partTimeEmp.showDetails(); |
|  |
|  |
| FullTimeEmp fullTimeEmp = new FullTimeEmp(2, "rohan", 30, 5000.0, |
| 1000.0); |
| System.out.println("\nFull Time Employee Details:"); fullTimeEmp.showDetails(); |
| } |
| } |
|  |
| /\* |
| output: |
| Part Time Employee Details:  ID: 1, Name: mahi, Age: 25, NumberOfHours: 20, RatePerHour: 15.0 |
|  |
| Full Time Employee Details: |
| ID: 2, Name: rohan, Age: 30, BasicPay: 5000.0, DA: 1000.0 |
|  |
| \*/ |

**Day-06**

/\*

|  |
| --- |
| 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 |
| { |
| public void getDetails() |
| { |
| System.out.println("Employee details"); |
| } |
| public void showDetails() |
| { |
| System.out.println("Showing employee details"); |
| } |
| } |
|  |
| class Manager extends Emp  { |
| @Override |
| public void getDetails() |
| { |
| System.out.println("Manager details"); |
| } |
| @Override |
| public void showDetails() |
| { |
| System.out.println("Showing manager details"); |
| } |
| } |
| class Supervisor extends Emp |
| { |
| @Override |
| public void getDetails() |
| { |
| System.out.println("Supervisor details"); |
| } |
| @Override |
| public void showDetails() |
| { |
| System.out.println("Showing supervisor details"); |
| } |
| } |
| class pro1 |
| { |
| public static void main(String[] args) |
| { |
| Emp emp; |
| emp = new Manager(); |
| emp.getDetails(); |
| emp.showDetails(); |
|  |
| emp = new Supervisor(); |
| emp.getDetails(); emp.showDetails(); |
| } |
| } |
|  |
| /\* |
| output: |
| Manager details |
| Showing manager details  Supervisor details |
| Showing supervisor details |
| \*/ |

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 |
| { public String name; |
| public int id; |
|  |
| public Emp(String name, int id) |
| { |
| this.name = name; |
| this.id = id; |
| } |
|  |
| public abstract void calculateSalary();  } |
|  |
| class Employee extends Emp |
| { |
| public double monthlySalary; |
|  |
| public Employee(String name, int id, double monthlySalary)  { |
| super(name, id); |
| this.monthlySalary = monthlySalary; |
| } |
|  |
| public void calculateSalary() |
| { |
| double annualSalary = monthlySalary \* 12;  System.out.println("Annual salary of employee " + name + " with id " + |
| id + " is: " + annualSalary); |
| } |
| } |
|  |
| class Main |
| { public static void main(String[] args) |
| { |
| Employee emp = new Employee("urmi", 123, 5000.0); |
| emp.calculateSalary(); |
| } |
| } |
| /\* |
|  |
| output: |
| Annual salary of employee urmi with id 123 is: 60000.0 |
|  |
| \*/ |

**Unit -2 & 6**

**Day-07**

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

/\*Essential Assignment:

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

Test working of both the methods.

\*/

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;

}

@Override

public String toString()

{

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

}

@Override 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 && name.equals(student.name) && age == student.age;

}

public static void main(String[] args)

{

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

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

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

System.out.println(student1);

System.out.println(student2);

System.out.println(student1.equals(student2));

// Outputs: false

System.out.println(student1.equals(student3));

// Outputs: true

}

}

2. 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 obj = new Test();

System.out.println("a = " + obj.a);

System.out.println("b = " + obj.b); System.out.println("c = " + obj.c);

}

}

OUTPUT :

a = 10 b = 20

c = 30

**Unit – 3**

# Day-08

1. Write a program which will handle the

ArrayIndexOutOfBoundsException and

DivideByZeroException with multiple catch and finally block.

import java.util.Scanner;

public class ExceptionHandlingExample

{

public static void main(String[] args)

{

Scanner scanner = new Scanner(System.in); int[] numbers = {1, 2, 3, 4, 5};

try

{

System.out.print("Enter the index of the number you want to access: "); int index = scanner.nextInt(); int number = numbers[index];

System.out.print("Enter a number to divide the array number with: "); int divisor = scanner.nextInt(); int result = number / divisor;

System.out.println("Result: " + result);

}

catch (ArrayIndexOutOfBoundsException e)

{

System.out.println("Invalid index! Array index should be

between 0 and " + (numbers.length - 1));

}

catch (ArithmeticException e)

{

System.out.println("Cannot divide by zero!");

}

catch (Exception e)

{

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

} finally

{

System.out.println("Finally block executed."); scanner.close();

}

}

}

/\*

OUTPUT :

Enter the index of the number you want to access: 12

Invalid index! Array index should be between 0 and 4

Finally block executed.

\*/

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

class AgeException extends Exception

{

public AgeException(String message)

{

super(message);

}

}

class Student

{

private int id; private String name; private int age;

public Student(int id, String name, int age) throws AgeException

{

this.id = id;

this.name = name; if (age < 20)

{

throw new AgeException("Age should be 20 years or above");

}

this.age = age;

}

public int getId()

{

return id;

}

public String getName()

{

return name;

}

public int getAge()

{

return age;

}

}

class pro2

{

public static void main(String[] args)

{

try

{

Student student1 = new Student(1, "John", 18); System.out.println("Student Created: "+student1.getName());

}

catch (AgeException e)

{

System.out.println("Error: "+e.getMessage());

}

try

{

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

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

}

catch (AgeException e)

{

System.out.println("Error: "+e.getMessage());

}

}

}

OUTPUT :

Error: Age should be 20 years or above

Student Created: Alice

# Day-09

/\*

1.With the help of Student(id,name,age) class do the followings:

**a**.Create list of 5 students. Iterate from first to last student and

display all the information about each student.

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

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

\*/

import java.util.ArrayList; import java.util.Stack; import java.util.Vector;

class pro1

{

public static void main(String[] args)

{

ArrayList<Student> studentList = new ArrayList<>(); studentList.add(new Student(1, "urmi", 20)); studentList.add(new Student(2, "suhani", 22)); studentList.add(new Student(3, "tejasvi", 19)); studentList.add(new Student(4, "parth", 21)); studentList.add(new Student(5, "khushali", 23));

System.out.println("List of students:"); for (Student student : studentList)

{

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

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

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

System.out.println();

}

Stack<Student> studentStack = new Stack<>(); studentStack.push(new Student(1, "urmi", 20)); studentStack.push(new Student(2, "suhani", 22)); studentStack.push(new Student(3, "tejasvi", 19)); studentStack.push(new Student(4, "parth", 21)); studentStack.push(new Student(5, "khushali", 23)); System.out.println("Stack of students:"); while (!studentStack.isEmpty())

{

Student student = studentStack.pop();

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

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

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

System.out.println();

}

Vector<Student> studentVector = new Vector<>(); studentVector.add(new Student(1, "tejasvi", 20)); studentVector.add(new Student(2, "suhani", 22)); studentVector.add(new Student(3, "urmi", 19)); studentVector.add(new Student(4, "krupa", 21)); studentVector.add(new Student(5, "greshi", 23));

System.out.println("Vector of students:"); for (Student student : studentVector)

{

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

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

System.out.println();

}

}

}

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 int getId()

{

return id;

}

public void setId(int id)

{

this.id = id;

}

public String getName()

{

return name;

}

public void setName(String name)

{

this.name = name;

}

public int getAge()

{

return age;

}

public void setAge(int age)

{

this.age = age;

}

}

OUTPUT:

List of students:

ID: 1

Name: urmi

Age: 20

ID: 2

Name: suhani

Age: 22

ID: 3

Name: tejasvi

Age: 19

ID: 4

Name: parth

Age: 21

5

khushali

23

Stack of students:

ID: 5

Name: khushali

Age: 23

ID: 4

Name: parth

Age: 21

ID: 3

Name: tejasvi

Age: 19

ID: 2

Name: suhani

Age: 22

1 urmi

20

Vector of students:

ID: 1

Name: urmi

Age: 20

ID: 2

Name: suhani

Age: 22

ID: 3

Name: tejasvi

Age: 19

ID: 4

Name: parth

Age: 21

5

khushali

23

**Unit – 4**

**Day-10**

**Difficulty Solving for Day1 to Day 9.**

# Day-11

/\*Essential Assignment:

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

\*/

class NumberThread extends Thread

{

private final int start; private final int end;

public NumberThread(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 class p1

{

public static void main(String[] args)

{

NumberThread thread1 = new NumberThread(1, 2500);

NumberThread thread2 = new NumberThread(2501, 5000);

thread1.start(); thread2.start();

}

}

**OUTPUT :**

1

2

3

4

5

6

7

8

9

10

11

12

13

….

2501

2502

2503

2504

2505

2506

....

2499

2500

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

class MyThread extends Thread { private int start; private int end;

public MyThread(int start, int end) { this.start = start; this.end = end;

}

@Override public void run() {

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

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

}

}

}

class MultiThrdEx {

public static void main(String[] args) {

// Create two threads

MyThread thrd1 = new MyThread(1, 2500);

MyThread thrd2 = new MyThread(2501, 5000);

// Start both threads thrd1.start(); thrd2.start();

}

}

OUTPUT:

Thread-1: 2501

Thread-1: 2502

Thread-1: 2503

Thread-1: 2504

Thread-1: 2505

Thread-1: 2506

Thread-1: 2507

Thread-1: 2508

…………

Thread-0: 100

Thread-0: 101

Thread-0: 102

# 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 pro1

{

private static int count = 0; private static final Object lock = new Object();

public static void main(String[] args) {

Thread t1 = new Thread(new CounterThread("Thread 1"));

Thread t2 = new Thread(new CounterThread("Thread 2"));

t1.start();

t2.start();

}

static class CounterThread implements Runnable { private final String threadName;

public CounterThread(String threadName) { this.threadName = threadName;

}

@Override public void run() { for (int i = 0; i < 5000; i++) { synchronized (lock) {

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

}

}

}

}

}

\*/

class pro1

{

private static int count = 0;

public static void main(String[] args)

{

Thread thread1 = new Thread(new CountIncrementTask(), "Thread 1");

Thread thread2 = new Thread(new CountIncrementTask(), "Thread 2");

thread1.start(); thread2.start();

try

{

thread1.join();

thread2.join();

}

catch (InterruptedException e)

{

e.printStackTrace();

}

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

}

static class CountIncrementTask implements Runnable

{

@Override public void run()

{

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

{

synchronized (pro1.class)

{

System.out.println(Thread.currentThread().getName() + " - Count: " + count);

count++;

}

}

}

}

}

//2.Write a program to create a new folder, rename and delete it. import java.io.File; class pro2

{

public static void main(String[] args)

{

String folderPath = "C:/ExampleFolder";

File folder = new File(folderPath); boolean isFolderCreated = folder.mkdir(); if (isFolderCreated)

{

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

}

else

{

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

}

String newFolderPath = "C:/RenamedFolder"; File newFolder = new File(newFolderPath); boolean isFolderRenamed = folder.renameTo(newFolder); if (isFolderRenamed)

{

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

}

else

{

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

}

boolean isFolderDeleted = newFolder.delete(); if (isFolderDeleted)

{

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

}

else

{

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

}

}

}

**OUTPUT :**

Folder created successfully.

Folder renamed successfully.

Folder deleted successfully.

# 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

\*/ import java.io.\*;

class pro1

{

public static void main(String[] args)

{

String inputFile = "input.txt"; String outputFile = "output.txt";

copyFileCharByChar(inputFile, outputFile); copyFileWordByWord(inputFile, outputFile); copyFileLineByLine(inputFile, outputFile);

}

public static void copyFileCharByChar(String inputFile, String

outputFile)

{

try (Reader reader = new FileReader(inputFile);

Writer writer = new FileWriter(outputFile))

{

int character; while ((character = reader.read()) != -1)

{

writer.write(character);

}

}

catch (IOException e)

{

e.printStackTrace();

}

}

public static void copyFileWordByWord(String inputFile, String

outputFile)

{

try (BufferedReader reader = new BufferedReader(new

FileReader(inputFile));

BufferedWriter writer = new BufferedWriter(new FileWriter(outputFile))) { String line; while ((line = reader.readLine()) != null)

{

String[] words = line.split("\\s+");

for (String word : words)

{

writer.write(word); writer.write(" ");

}

writer.newLine();

}

}

catch (IOException e)

{

e.printStackTrace();

}

}

public static void copyFileLineByLine(String inputFile, String

outputFile)

{

try (BufferedReader reader = new BufferedReader(new

FileReader(inputFile));

BufferedWriter writer = new BufferedWriter(new FileWriter(outputFile))) { String line; while ((line = reader.readLine()) != null)

{

writer.write(line); writer.newLine();

}

}

catch (IOException e)

{

e.printStackTrace();

}

}

}

// 2.Write the program to marge two files in to third file using any of //the above approaches. import java.io.\*;

class MergeFiles

{

public static void main(String[] args)

{

String filePath1 = "file1.txt"; // Path of the first file

String filePath2 = "file2.txt"; // Path of the second file String mergedFilePath = "merged\_file.txt";

try

{

File file1 = new File(filePath1);

File file2 = new File(filePath2);

File mergedFile = new File(mergedFilePath);

BufferedReader reader1 = new BufferedReader(new

FileReader(file1));

BufferedReader reader2 = new BufferedReader(new

FileReader(file2));

BufferedWriter writer = new BufferedWriter(new

FileWriter(mergedFile));

String line; while ((line = reader1.readLine()) != null)

{

writer.write(line); writer.newLine();

}

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

{

writer.write(line); writer.newLine();

}

reader1.close(); reader2.close(); writer.close();

System.out.println("Merge completed successfully.");

}

catch (IOException e)

{

e.printStackTrace();

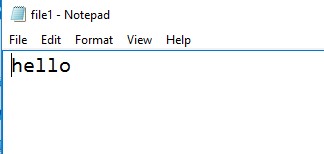
}

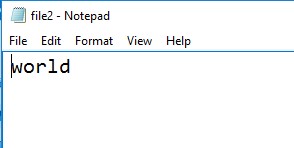
}

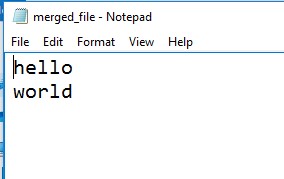
}

**OUTPUT :**

Merge completed successfully.







# 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.\*;

class Student implements Serializable

{

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 int getId()

{

return id;

}

public String getName()

{

return name;

}

public int getAge()

{

return age;

}

}

class pro1

{

public static void main(String[] args)

{

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

students.add(new Student(1, "Urmi Modh", 20)); students.add(new Student(2, "Suhani Babriya", 21)); students.add(new Student(3, "Tejasvi Dudhat", 19)); students.add(new Student(4, "Hani Soni", 22)); students.add(new Student(5, "Parth Trivedi", 20));

try

{

FileOutputStream fileOut = new

FileOutputStream("students.ser");

ObjectOutputStream objectOut = new

ObjectOutputStream(fileOut); for (Student student : students)

{

objectOut.writeObject(student);

}

objectOut.close(); fileOut.close();

System.out.println("Student objects have been written to students.ser file.");

}

catch (IOException e)

{

e.printStackTrace();

}

Scanner scanner = new Scanner(System.in);

System.out.println("Enter the ID of the student you want to retrieve: "); int studentId = scanner.nextInt();

try

{

FileInputStream fileIn = new FileInputStream("students.ser"); ObjectInputStream objectIn = new ObjectInputStream(fileIn); boolean found = false; while (true)

{

try

{

Student student = (Student) objectIn.readObject(); if (student.getId() == studentId)

{

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

System.out.println("Student Name: " +

student.getName());

System.out.println("Student Age: " + student.getAge()); found = true;

break;

}

}

catch (EOFException e)

{

break;

}

}

objectIn.close(); fileIn.close();

if (!found)

{

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

}

}

catch (IOException | ClassNotFoundException e)

{

e.printStackTrace();

}

}

}

OUTPUT :

Student objects have been written to students.ser file.

Enter the ID of the student you want to retrieve:

4

Student ID: 4

Student Name: Hani Soni

Student Age: 22

**Unit – 5**

**Day-15**

/\*

1. Create a JFrame with a button named “Click Me”.

Register JFrame with MouseListener, MouseMotion Listener, KeyListener, WindowsListener.

Print appropriate message which will show the execution of particular event of above listener.

While clicking on button, with help of ActionListener,

\*/

import javax.swing.\*; import java.awt.\*; import java.awt.event.\*;

class MainFrame extends JFrame implements MouseListener, MouseMotionListener, KeyListener, WindowListener

{

private JButton button; public MainFrame()

{

setTitle("Event Listener Example"); setSize(400, 300); setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

button = new JButton("Click Me"); button.addMouseListener(this); button.addMouseMotionListener(this); button.addKeyListener(this); addWindowListener(this); setLayout(new FlowLayout()); add(button);

}

public void printMessage(String message)

{

System.out.println(message);

}

public void mouseClicked(MouseEvent e)

{

printMessage("Mouse Clicked");

}

public void mouseEntered(MouseEvent e)

{

printMessage("Mouse Entered");

}

public void mouseExited(MouseEvent e)

{

printMessage("Mouse Exited");

}

public void mousePressed(MouseEvent e)

{

printMessage("Mouse Pressed");

}

public void mouseReleased(MouseEvent e)

{

printMessage("Mouse Released");

}

public void mouseDragged(MouseEvent e)

{

printMessage("Mouse Dragged");

}

public void mouseMoved(MouseEvent e)

{

printMessage("Mouse Moved");

}

public void keyTyped(KeyEvent e)

{

printMessage("Key Typed");

}

public void keyPressed(KeyEvent e)

{

printMessage("Key Pressed");

}

public void keyReleased(KeyEvent e)

{

printMessage("Key Released");

}

public void windowOpened(WindowEvent e)

{

printMessage("Window Opened");

}

public void windowClosing(WindowEvent e)

{

printMessage("Window Closing");

}

public void windowClosed(WindowEvent e)

{

printMessage("Window Closed");

}

public void windowIconified(WindowEvent e)

{

printMessage("Window Iconified");

}

public void windowDeiconified(WindowEvent e)

{

printMessage("Window Deiconified");

}

public void windowActivated(WindowEvent e)

{

printMessage("Window Activated");

}

public void windowDeactivated(WindowEvent e)

{

printMessage("Window Deactivated");

}

public static void main(String[] args)

{

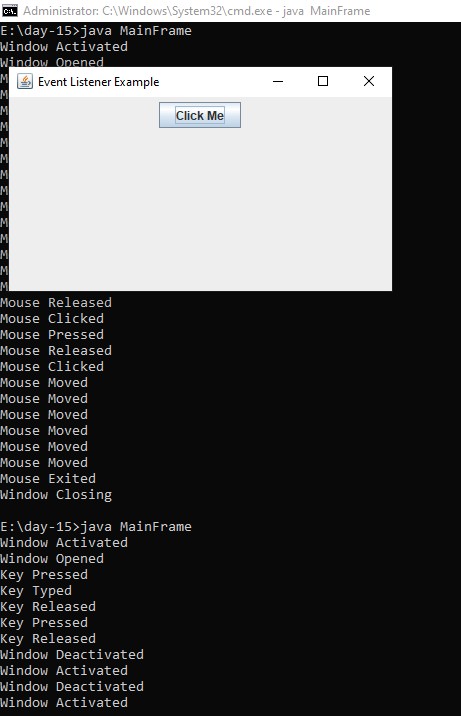
MainFrame frame = new MainFrame();

frame.setVisible(true);

}

}

OUTPUT :



# Day-16

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

import javax.swing.\*; import java.awt.\*;

class pro1

{

public static void main(String[] args)

{

JFrame frame = new JFrame("Layout Example"); frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); frame.setSize(400, 300);

JPanel flowPanel = new JPanel(new FlowLayout()); flowPanel.add(new JButton("Button 1")); flowPanel.add(new JButton("Button 2")); flowPanel.add(new JButton("Button 3")); frame.add(flowPanel, BorderLayout.NORTH);

JPanel borderPanel = new JPanel(new BorderLayout()); borderPanel.add(new JButton("Button 4"), BorderLayout.WEST);

borderPanel.add(new JButton("Button 5"),

BorderLayout.CENTER); borderPanel.add(new JButton("Button 6"), BorderLayout.EAST); frame.add(borderPanel, BorderLayout.CENTER);

JPanel gridPanel = new JPanel(new GridLayout(2, 2)); gridPanel.add(new JButton("Button 7")); gridPanel.add(new JButton("Button 8")); gridPanel.add(new JButton("Button 9")); gridPanel.add(new JButton("Button 10")); frame.add(gridPanel, BorderLayout.SOUTH);

JPanel gridBagPanel = new JPanel(new GridBagLayout()); GridBagConstraints gbc = new GridBagConstraints(); gbc.gridx = 0; gbc.gridy = 0; gridBagPanel.add(new JButton("Button 11"), gbc); gbc.gridx = 1; gbc.gridy = 0; gridBagPanel.add(new JButton("Button 12"), gbc); gbc.gridx = 0;

gbc.gridy = 1; gbc.gridwidth = 2; gridBagPanel.add(new JButton("Button 13"), gbc); frame.add(gridBagPanel, BorderLayout.WEST);

JPanel cardPanel = new JPanel(new CardLayout()); cardPanel.add(new JLabel("Card 1"), "card1"); cardPanel.add(new JLabel("Card 2"), "card2"); cardPanel.add(new JLabel("Card 3"), "card3");

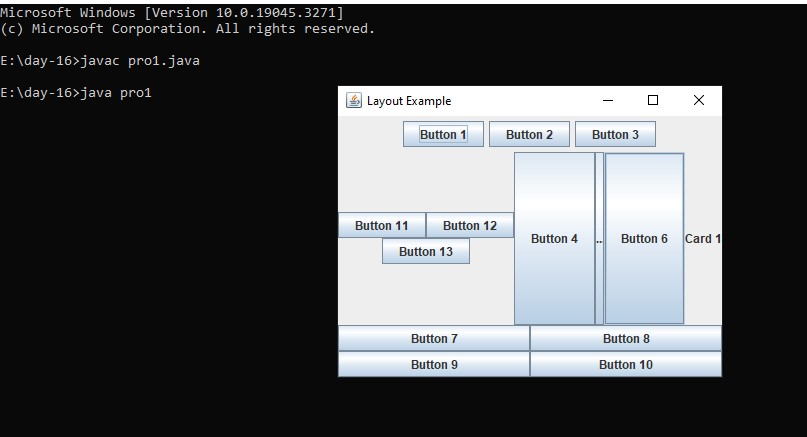
frame.add(cardPanel, BorderLayout.EAST);

frame.setVisible(true);

}

}

OUTPUT :



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

import javax.swing.\*; import java.awt.event.ActionEvent; import java.awt.event.ActionListener;

class pro2

{

public static void main(String[] args)

{

JFrame frame = new JFrame("Java Text Field and Text Area

Example"); frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); frame.setLayout(null);

JTextField textField = new JTextField(); textField.setBounds(50, 50, 200, 30); frame.add(textField);

JTextArea textArea = new JTextArea(); textArea.setBounds(50, 100, 200, 150); frame.add(textArea);

JButton button = new JButton("Submit"); button.setBounds(50, 270, 80, 30); frame.add(button);

button.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

String inputText = textField.getText(); textArea.append(inputText + "\n");

}

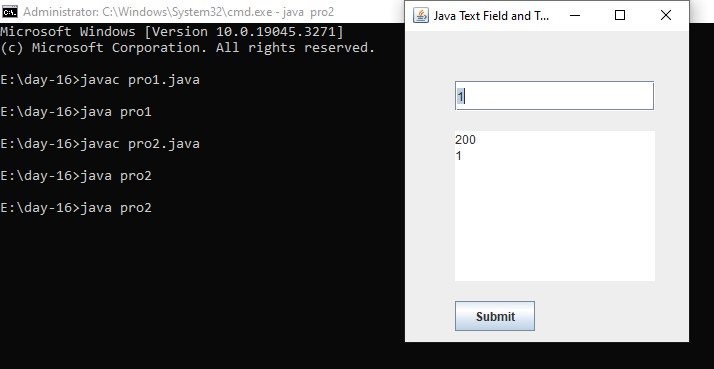
});

frame.setSize(300, 350); frame.setVisible(true);

}

}

OUTPUT :



# 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.Pattern;

class StudentInfoApp extends JFrame implements ActionListener {

private JTextField idField; private JTextField nameField; private JTextField ageField; private JTextField emailField; private JTextField semField;

public StudentInfoApp() { setTitle("Student Information"); setSize(300, 250); setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); setLocationRelativeTo(null);

JPanel panel = new JPanel(new GridLayout(5, 2));

JLabel idLabel = new JLabel("ID:"); idField = new JTextField();

JLabel nameLabel = new JLabel("Name:"); nameField = new JTextField(); JLabel ageLabel = new JLabel("Age:"); ageField = new JTextField(); JLabel emailLabel = new JLabel("Email:"); emailField = new JTextField();

JLabel semLabel = new JLabel("Semester:");

semField = new JTextField();

panel.add(idLabel); panel.add(idField); panel.add(nameLabel); panel.add(nameField); panel.add(ageLabel); panel.add(ageField); panel.add(emailLabel); panel.add(emailField); panel.add(semLabel); panel.add(semField);

JButton submitButton = new JButton("Submit"); JButton clearButton = new JButton("Clear");

submitButton.addActionListener(this); clearButton.addActionListener(this);

JPanel buttonPanel = new JPanel(); buttonPanel.add(submitButton); buttonPanel.add(clearButton);

getContentPane().add(panel, BorderLayout.CENTER); getContentPane().add(buttonPanel, BorderLayout.SOUTH);

}

@Override public void actionPerformed(ActionEvent e)

{

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

{

String id = idField.getText();

String name = nameField.getText();

String age = ageField.getText();

String email = emailField.getText(); String sem = semField.getText();

if (isValidEmail(email))

{

String message = "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 isValidEmail(String email)

{

String emailRegex = "^[A-Za-z0-9+\_.-]+@[A-Za-z0-9.-]+$"; Pattern pattern = Pattern.compile(emailRegex); return pattern.matcher(email).matches();

}

private void clearFields()

{

idField.setText(""); nameField.setText(""); ageField.setText(""); emailField.setText(""); semField.setText("");

}

public static void main(String[] args)

{

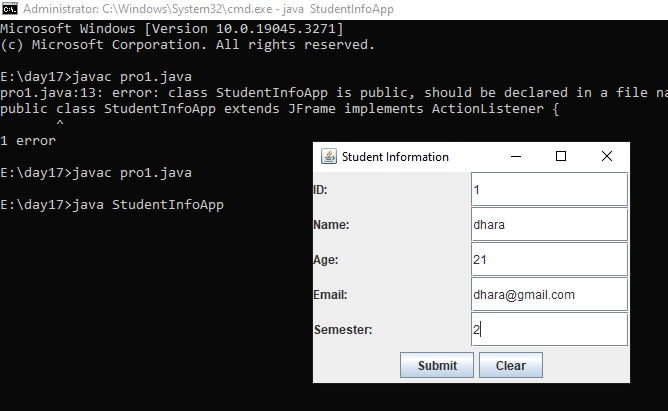
StudentInfoApp app = new StudentInfoApp();

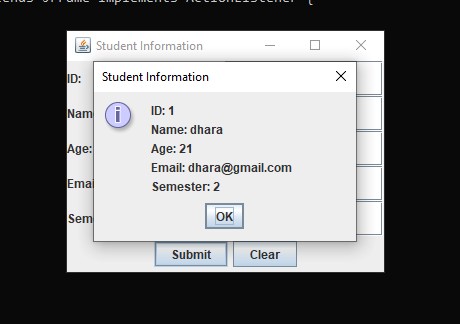
app.setVisible(true);

}

}

OUTPUT :





/\*

2.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.\*;

public class StudentInfoMenu extends JFrame implements

ActionListener {

private JTextField idField, nameField, ageField, emailField,

semField;

public StudentInfoMenu() { setTitle("Student Information Form"); setSize(400, 300); setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

// Create labels

JLabel idLabel = new JLabel("ID:");

JLabel nameLabel = new JLabel("Name:");

JLabel ageLabel = new JLabel("Age:");

JLabel emailLabel = new JLabel("Email:");

JLabel semLabel = new JLabel("Semester:");

// Create text fields idField = new JTextField(10); nameField = new JTextField(10); ageField = new JTextField(10); emailField = new JTextField(10); semField = new JTextField(10);

// Create buttons

JButton submitButton = new JButton("Submit");

JButton clearButton = new JButton("Clear");

// Add action listeners to buttons submitButton.addActionListener(this); clearButton.addActionListener(this);

// Create panel for form elements JPanel panel = new JPanel(); panel.setLayout(new GridLayout(6, 2)); panel.add(idLabel); panel.add(idField); panel.add(nameLabel); panel.add(nameField); panel.add(ageLabel); panel.add(ageField); panel.add(emailLabel); panel.add(emailField); panel.add(semLabel); panel.add(semField); panel.add(submitButton); panel.add(clearButton);

// Add panel to frame

getContentPane().add(panel);

// Create menu bar

JMenuBar menuBar = new JMenuBar();

// Operations menu

JMenu operationsMenu = new JMenu("Operations");

JMenuItem newMenuItem = new JMenuItem("New");

JMenuItem closeMenuItem = new JMenuItem("Close"); JMenuItem exitMenuItem = new JMenuItem("Exit"); newMenuItem.addActionListener(this); closeMenuItem.addActionListener(this); exitMenuItem.addActionListener(this); operationsMenu.add(newMenuItem); operationsMenu.add(closeMenuItem); operationsMenu.add(exitMenuItem); menuBar.add(operationsMenu);

// Help menu

JMenu helpMenu = new JMenu("Help");

JMenuItem aboutMenuItem = new JMenuItem("About

Application"); aboutMenuItem.addActionListener(this); helpMenu.add(aboutMenuItem); menuBar.add(helpMenu); setJMenuBar(menuBar);

setVisible(true);

}

// ActionListener implementation public void actionPerformed(ActionEvent e) { if (e.getActionCommand().equals("Submit")) {

// Validate email

if (!isValidEmail(emailField.getText())) {

JOptionPane.showMessageDialog(this, "Please enter a valid email address", "Invalid Email", JOptionPane.ERROR\_MESSAGE);

return;

}

// Display student information

String message = "ID: " + idField.getText() + "\n" +

"Name: " + nameField.getText() + "\n" +

"Age: " + ageField.getText() + "\n" +

"Email: " + emailField.getText() + "\n" +

"Semester: " + semField.getText(); JOptionPane.showMessageDialog(this, message, "Student

Information", JOptionPane.INFORMATION\_MESSAGE);

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

// Clear all fields idField.setText(""); nameField.setText(""); ageField.setText(""); emailField.setText(""); semField.setText("");

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

// Clear all fields idField.setText(""); nameField.setText(""); ageField.setText(""); emailField.setText(""); semField.setText("");

} else if (e.getActionCommand().equals("Close")) { dispose(); // Close the window

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

System.exit(0); // Exit the application

} else if (e.getActionCommand().equals("About Application")) { JOptionPane.showMessageDialog(this, "This is a simple application for storing student information.", "About", JOptionPane.INFORMATION\_MESSAGE);

}

}

// Method to validate email address private boolean isValidEmail(String email) {

// Simple email validation using regular expression

String emailRegex =

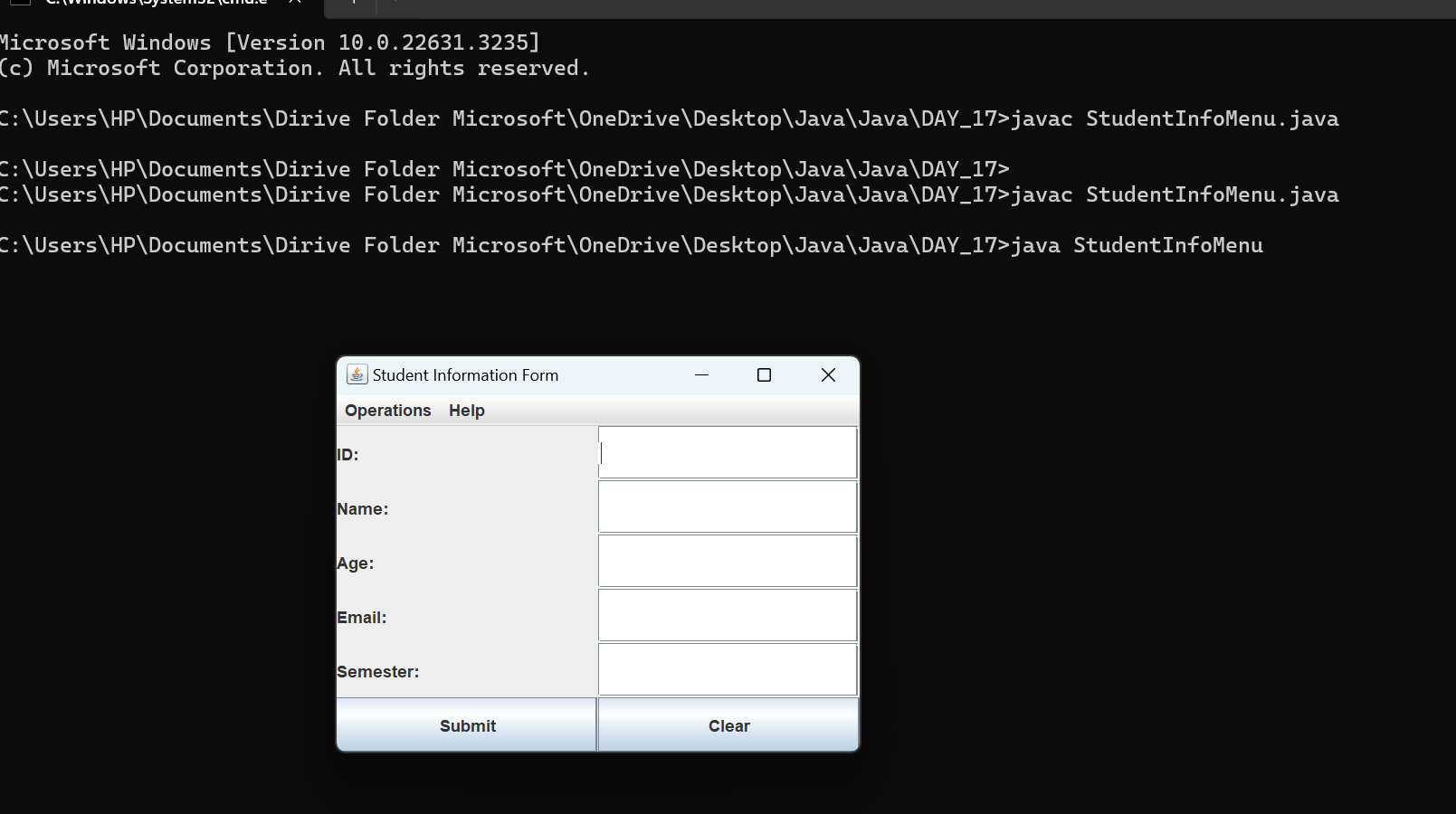
"^[a-zA-Z0-9\_+&\*-]+(?:\\.[a-zA-Z0-9\_+&\*-]+)\*@(?:[a-zA-Z0-9-]+\\.)+[ a-zA-Z]{2,7}$"; return email.matches(emailRegex);

}

public static void main(String[] args) { new StudentInfoMenu();

}

}



# Day – 18

**Completing pending work of Day1 to Day 17**