Assignment No. 1

Assignment Title: A java program to demonstrate the concept of operators.

Aim: Write a Java program that accepts four integers from the user and prints equal if all four are equal, and not equal otherwise.

Pre-Requisites: C/C++

Objective: The objective is to impart fundamentals of object-oriented programming in Java, including defining classes, invoking methods, using class libraries, etc.

Outcomes: After learning this concept students will be able to,

1. Analyze the necessity for Object Oriented Programming paradigm over structured programming and become familiar with the fundamental concepts in OOP like encapsulation, Inheritance and Polymorphism

2. Design and develop java programs, analyse, and interpret object oriented data and report results.

Theory:

OPERATORS IN JAVA

Operator in Java is a symbol that is used to perform operations. For example: +, -, *, / etc.

There are many types of operators in Java which are given below:

Unary Operator,
Arithmetic Operator,
Shift Operator,
Relational Operator,
Bitwise Operator,
Logical Operator,
Ternary Operator and
Assignment Operator.

Java Operator Precedence

Operator Type	Category	Precedence

Unary	postfix	*expr*++ *expr*--
	prefix	++*expr* --*expr* +*expr* -*expr* ~ !
Arithmetic	multiplicative	* / %
	additive	+ -
Shift	shift	<< >> >>>
Relational	comparison	< > <= >= instance of
	equality	== !=
Bitwise	bitwise AND	&
	bitwise exclusive OR	^
	bitwise inclusive OR	|
Logical	logical AND	&&
	logical OR	||
Ternary	ternary	? :
Assignment	assignment	= += -= *= /= %= &= ^= |= <<= >>= >>>=

Java Arithmetic Operators

Java arithmetic operators are used to perform addition, subtraction, multiplication, and division. They act as basic mathematical operations.

Java User Input

The Scanner class is used to get user input, and it is found in the java.util package.

To use the Scanner class, create an object of the class and use any of the available methods found in the Scanner class documentation. In our example, we will use the nextLine() method, which is used to read Strings

Example

```
import java.util.Scanner;  // Import the Scanner class
```

```java
class Main {
  public static void main(String[] args) {
    Scanner myObj = new Scanner(System.in);  // Create a Scanner object
    System.out.println("Enter username");
    String userName = myObj.nextLine(); // Read user input
    System.out.println("Username is: " + userName); // Output user input
  }
}
```

INPUT TYPES

In the example above, we used the nextLine() method, which is used to read Strings. To read other types, look at the table below:

Method	Description
nextBoolean()	Reads a boolean value from the user
nextByte()	Reads a byte value from the user
nextDouble()	Reads a double value from the user
nextFloat()	Reads a float value from the user
nextInt()	Reads a int value from the user

nextLine()	Reads a String value from the user
nextLong()	Reads a long value from the user
nextShort()	Reads a short value from the user

Java Arithmetic Operator Example

```
public class OperatorExample{
public static void main(String args[]){
int a=10;
int b=5;
System.out.println(a+b);//15
System.out.println(a-b);//5
System.out.println(a*b);//50
System.out.println(a/b);//2
System.out.println(a%b);//0
}}
```

Output:

```
15
5
50
2
0
```

Fig. Acceptance of four integers from user and checking there equality

Algorithm/Steps:

Step 1: Start

Step 2: Accept four integer numbers from user.

Step 3: Check the equality of numbers using assignment and AND operator.

Step 4: If all numbers are equal Print output ”All four integers are equal”

Step 5: Else Print output ”All four integers are not equal”

Step 6: Stop

Flowchart:

Program Code:

```
import java.util.Scanner;


public class Assignment1 {
    public static void main(String[] args)
        { Scanner scanner = new Scanner(System.in);
        System.out.println("Enter four Integers:");
        int firstNumber = scanner.nextInt();
        int secondNumber = scanner.nextInt();
        int thirdNumber = scanner.nextInt();
        int fourthNumber = scanner.nextInt();
        if ((firstNumber == secondNumber) && (secondNumber == thirdNumber) &&
(thirdNumber == fourthNumber)) {
```

```
            System.out.println("All four Integers are equal.");
        }
        else{
            System.out.print("All four Integers are not equal.");
        }
    }
}
```

Sample Output:

```
PS C:\SOURAV\CODE\Java\Java Assignments> cd "c:\SOURAV\CODE\
Enter four Integers:
1
1
1
1
All four Integers are equal.
PS C:\SOURAV\CODE\Java\Java Assignments>
```

Conclusion: Thus we implemented a Java program that accepts four integers from the user and prints equal if all four are equal, and not equal otherwise.

Frequently Asked Questions:

MIT SCHOOL OF ENGINEERING
Rajbaug, Loni-Kalbhor, Pune

MIT-ADT UNIVERSITY PUNE, INDIA

Name : Sourav Shailesh Toshniwal.
Class : SY - 1 (B)
Roll no : 2213047
Subject : Java Programming lab.

Assignment - 1

1. What is Class?

Ans - A class can be stated as the blueprint of the object. Class states what an object can do. We can say that class contains the defination of object.

Syntax : Access Modifier class ClassName {
data member;
method;
constructor; ← Class body.
nested class;
interface;
}

2. What is object - oriented Programming?

Ans - OOP is programming methedology that has certain principles & with the help of these principles, the software is developed.
- OOP principles is the most popular amongst others because it relates to real-life objects.
- It provides principles like inheritance, Abstraction, Encapsulation.
- Every operation that is going to be functional is considered in terms of classes and objects.

- Languages that supports OOPs are C++, Java, Javascript, C#, PHP, etc.

2) How to invoke method in Java?

Ans Methods in Java can be invoked or called by creating an instance of the class in which the method is defined and then using the instance to call the method. The basic syntax for calling method in Java is:

Objectname.methodName (arg1, arg2, ...)

4) which operators are supported by Java.

Ans 1) Arithmetic operators: Performs operation like addition, subtraction, multiplication, etc. (+, -, *, /)

2) Comparison operators: Compare two values and return a boolean result, such as equal to (==) or not equal to (!=), etc.

3) Logical operators: Perform logical operations, such as AND ('&&'), OR ('||') and NOT ('!').

4) Bitwise operators: Perform operations on individual bits within a number, such as bitwise AND ('&') bitwise OR ('|'), bitwise XOR ('^') etc.

5) Write syntax for Scanner in Java?

Ans Scanner class in Java is used to read input from various sources. To use 'scanner' class you first need to import the 'java.util.scanner' package.

```
import java.util.Scanner;
Scanner sc = new Scanner (System.in);
```

5) What is operator precedence?

Ans Operator precedence in java determines the order in which the operators given in a particular Expression are evaluated if there are multiple operators in a single expression. Operators with higher precedence are evaluated first.

Assignment No. 02

Assignment Title: Basic concepts of Java Programming with classes and objects.

Aim: Write a program to create a class Student2 along with two method getData

(),
printData () to get the value through argument and display the data in printData.
Create the two objects s1, s2 to declare and access the values from class STtest

Pre-Requisites: C/C++

Objective: To demonstrate concept of object oriented programming paradigm and Java including Simple Java Program, Java Tokens, Java Statements, Constants, variables, data types. Declaration of variables, Giving values to variables

Outcomes: Students are able to Understand and Apply Object oriented features and Java concepts

Theory:

Classes and Objects.

- Object – Objects have states and behaviors. Example: A dog has states - color, name, breed as well as behaviors – wagging the tail, barking, eating. An object is an instance of a class.
- Class – A class can be defined as a template/blueprint that describes the behavior/state that the object of its type support.
- Objects in Java
- Let us now look deep into what are objects. If we consider the real-world, we can find many objects around us, cars, dogs, humans, etc. All these objects have a state and a behavior.
- If we consider a dog, then its state is - name, breed, color, and the behavior is - barking, wagging the tail, running.
- If you compare the software object with a real-world object, they have very similar characteristics.
- Software objects also have a state and a behavior. A software object's state is stored in fields and behavior is shown via methods.
- So in software development, methods operate on the internal state of an object and the object-to-object communication is done via methods.

Classes in Java

A class is a blueprint from which individual objects are

created. Following is a sample of a class.

Example
```
public class Dog
  {String breed;
```

```
  int age;
  String color;

  void barking() {
  }

  void hungry() {
  }

  void sleeping() {
  }
}
```

A class can contain any of the following variable types.

- Local variables – Variables defined inside methods, constructors or blocks are called local variables. The variable will be declared and initialized within the method and the variable will be destroyed when the method has completed.
- Instance variables – Instance variables are variables within a class but outside any method. These variables are initialized when the class is instantiated. Instance variables can be accessed from inside any method, constructor or blocks of that particular class.
- Class variables – Class variables are variables declared within a class, outside any method, with the static keyword.

Creating an Object

As mentioned previously, a class provides the blueprints for objects. So basically, an object is created from a class. In Java, the new keyword is used to create new objects.

There are three steps when creating an object from a class –

- Declaration – A variable declaration with a variable name with an object type.
- Instantiation – The 'new' keyword is used to create the object.
- Initialization – The 'new' keyword is followed by a call to a constructor. This call initializes the new object.

Example

```
public class Puppy {
  public Puppy(String name) {
    // This constructor has one parameter, name.
    System.out.println("Passed Name is :" +
    name );
  }

  public static void main(String []args) {
    // Following statement would create an object
    myPuppy Puppy myPuppy = new Puppy( "tommy" );
```

```
  }
}
```

What is a method in Java?

A method is a block of code or collection of statements or a set of code grouped together to perform a certain task or operation. It is used to achieve the reusability of code. We write a method once and use it many times. We are not required to write code again and again. It also provides the easy modification and readability of code, just by adding or removing a chunk of code. The method is executed only when we call or invoke it.

```
//user defined method
public static void findEvenOdd(int num)
{
//method
body
if(num%2==0)
System.out.println(num+" is even");
else
System.out.println(num+" is odd");
}
```

Algorithm/Steps:

- Create a class Student2
- Define required variables
- Define method getdata()
- Define method printdata()
- Define class STest
- Create the objects
- Access the values from Class STest

Code:

```java
import java.util.Scanner;


class Student2 {
    private String name;
    private String rollno;
    private int marks;
    Scanner scanner = new Scanner(System.in);
    public void getData() {
        System.out.println("Enter the Name, Roll No. and Marks of student: ");
        String name = scanner.nextLine();
        String rollno = scanner.nextLine();
        int marks = scanner.nextInt();
        this.name = name;
        this.rollno = rollno;
        this.marks = marks;
    }


    public void printData()
        { System.out.println("Name: " +
        this.name);
        System.out.println("Roll number: " + this.rollno);
        System.out.println("Marks: " + this.marks);
    }
}
public class Assignment2 {
    public static void main(String[] args)
        {Student2 s1=new Student2();
        Student2 s2=new Student2();
        s1.getData();
```

```
        s2.getData();
        System.out.println("\nDetails of Student 1 are:");
        s1.printData();
        System.out.println("\nDetails of Student 2 are:");
        s2.printData();
    }
}
```

Output:

```
PS C:\SOURAV\CODE\Java\Java Assignments> cd "c:\SOURAV\CODE\Java\Java Assignments\
Enter the Name, Roll No. and Marks of student:
Sourav
2213047
20
Enter the Name, Roll No. and Marks of student:
Naman
2213027
20

Details of Student 1 are:
Name: Sourav
Roll number: 2213047
Marks: 20

Details of Student 2 are:
Name: Naman
Roll number: 2213027
Marks: 20
PS C:\SOURAV\CODE\Java\Java Assignments>
```

Conclusion:

Hence we studied the basic programming of java with classes and objects.

FAQs:

MIT SCHOOL OF ENGINEERING
Rajbaug, Loni-Kalbhor, Pune

MIT-ADT UNIVERSITY
PUNE, INDIA

Name: Sourav Shailesh Toshniwal.
Class: SY-1 (B)
Roll no: 2213047
Subject: JPL

Assignment - 2

1) Explain how to create a class?

Ans Syntax: access modifier class ClassName {
data members;
Constructors;
method;
nested class;
interface;
}

2) What is the role of public static void main () method?

Ans The keyword public static void main () is the mean by which you create a main method within the Java application. It's the core method of the program and calls all other. It can't return values and accept parameters for complex command line processing

3) Difference between structure & classes?

Ans - Structures are values type and classes are reference types.
- A variable of a structure type contains structure's data rather than containing a reference to the data as a class type does.
- Structure use stack allocation and classes use heap allocations.
- Structure syntax: struct mystruct {
};

- Class syntax: public class Myclass {
};

4) What is the meaning of instance variables and local variables?

Ans Instance variables: They are variables within a class but outside any method. These variables are initialized when the class is instantiated. Instance variables can be accessed from inside any method, constructor or blocks of that particular class.

Local variables: Variables defined inside methods constructors or blocks are called local variables. The variables will be declared and initialized within the method and the variable will be destroyed when the method has completed.

5) How to declare and define methods in Java?

Ans A method is a block of code or collection of statements or a set of code grouped together to perform a certain task or operations. The only required elements of a method declaration are the method's return type, name, a pair of parantheses () and a body.

Eg : void mymethod () {
}

6) Explain the process to create Object in Java?

Ans There are three steps when creating an object from a class.

i) Declaration: A variable declaration with a variable name with an object type.

ii) Instantiation: The 'new' keyword is used to create the object.

iii) Initialization: The 'new' keyword is followed by a call to a constructor. This call initializes the new object.

Assignment No. 03

Assignment Title: Implement a concept of inheritance in Java.

Aim: Write a java program to create class car, truck and motorcycle which extends the vehicle class (attribute registration number, color, type of vehicle) with their own attribute like make, CC and fuel type. Input data from the user and print all the details.

Pre-Requisites: C/C++

Objective: Demonstrate and apply the concept of inheritance.

Outcomes: Students are able to Understand and Apply inheritance concepts with java programming

Theory:

Inheritance in Java
Inheritance is an important pillar of OOP(Object-Oriented Programming). It is the mechanism in java by which one class is allowed to inherit the features(fields and methods) of another class. In Java, inheritance means creating new classes based on existing ones. A class that inherits from another class can reuse the methods and fields of that class. In addition, you can add new fields and methods to your current class as well.

Important terminologies used in Inheritance:

Class: Class is a set of objects which shares common characteristics/ behavior and common properties/ attributes. Class is not a real-world entity. It is just a template or blueprint or prototype from which objects are created.

Super Class/Parent Class: The class whose features are inherited is known as a superclass(or a base class or a parent class).

Sub Class/Child Class: The class that inherits the other class is known as a subclass(or a derived class, extended class, or child class). The subclass can add its own fields and methods in addition to the superclass fields and methods.

Reusability: Inheritance supports the concept of “reusability”, i.e. when we want to create a new class and there is already a class that includes some of the code that we want, we can derive our new class from the existing class. By doing this, we are reusing the fields and methods of the existing class.

Types of Inheritance:-

1) Single

2) Multilevel

3) Hierarchical

4) Multiple

5) Hybrid

Example:-

```
class Animal{
void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
void bark(){System.out.println("barking...");}
}
```

```
class TestInheritance{
public static void main(String
args[]){Dog d=new Dog();
d.bark();
d.eat();
}}
```

Algorithm/Steps:

Define superclass vehicle
Suppose that a program has to deal with motor vehicles, including cars, trucks, and motorcycles.
The program could use a class named Vehicle to represent all types of vehicles.
Since cars, trucks, and motorcycles are types of vehicles, they would be represented by subclasses of the Vehicle class, as shown in this class hierarchy diagram:

The Vehicle class would include instance variables such as registration Number and owner and instance methods such as transferOwnership(). These are variables and methods common to all vehicles.
The three subclasses of Vehicle—Car, Truck, and Motorcycle—could then be used to hold variables and methods specific to particular types of vehicles

Code:

```
import java.util.Scanner;



class Vehicle {

    String registrationNumber;

    String color;

    String typeOfVehicle;
```

```
    public Vehicle(String registrationNumber, String color, String typeOfVehicle)
{
        this.registrationNumber = registrationNumber;
        this.color = color;
        this.typeOfVehicle = typeOfVehicle;
    }

    void displayDetails() {
        System.out.println("Registration number: " + registrationNumber);
        System.out.println("Color: " + color);
        System.out.println("Type of vehicle: " + typeOfVehicle);
    }
}


class Car extends Vehicle
    {String make;
    int CC;
    String fuelType;

    public Car(String registrationNumber, String color, String make, int CC,
String fuelType) {
        super(registrationNumber, color, "Car");
        this.make = make;
        this.CC = CC;
        this.fuelType = fuelType;
    }

    void displayDetails()
        { super.displayDetails()
        ;
```

```
        System.out.println("Make: " + make);
        System.out.println("CC: " + CC);
        System.out.println("Fuel type: " + fuelType);
        System.out.println();
    }
}


class Truck extends Vehicle
    {String make;
    int CC;
    String fuelType;

    public Truck(String registrationNumber, String color, String make, int CC,
String fuelType) {
        super(registrationNumber, color, "Truck");
        this.make = make;
        this.CC = CC;
        this.fuelType = fuelType;
    }

    void displayDetails()
        { super.displayDetails();
        System.out.println("Make: " + make);
        System.out.println("CC: " + CC);
        System.out.println("Fuel type: " + fuelType);
        System.out.println();
    }
}
```

```java
class Motorcycle extends Vehicle
    {String make;
    int CC;
    String fuelType;

    public Motorcycle(String registrationNumber, String color, String make, int
CC, String fuelType) {
        super(registrationNumber, color, "Motorcycle");
        this.make = make;
        this.CC = CC;
        this.fuelType = fuelType;
    }

    void displayDetails()
        { super.displayDetails();
        System.out.println("Make: " + make);
        System.out.println("CC: " + CC);
        System.out.println("Fuel type: " + fuelType);
        System.out.println();
    }
}


public class Assignment3 {
    public static void main(String[] args)
        { Scanner scanner = new
        Scanner(System.in);
        String rn1, rn2, rn3, c1, c2, c3, m1, m2, m3, f1, f2, f3;
        int  cc1, cc2, cc3;
```

```
        System.out.println("Enter the Registration Number, Color, Make, CC and
fuel type of Car: ");
        rn1 = scanner.next();
        c1 = scanner.next();
        m1 = scanner.next();
        cc1 = scanner.nextInt();
        f1 = scanner.next();
        System.out.println("Enter the Registration Number, Color, Make, CC and
fuel type of Truck: ");
        rn2 = scanner.next();
        c2 = scanner.next();
        m2 = scanner.next();
        cc2 = scanner.nextInt();
        f2 = scanner.next();
        System.out.println("Enter the Registration Number, Color, Make, CC and
fuel type of Motorcycle: ");
        rn3 = scanner.next();
        c3 = scanner.next();
        m3 = scanner.next();
        cc3 = scanner.nextInt();
        f3 = scanner.next();


        Car car = new Car(rn1, c1, m1, cc1, f1);
        car.displayDetails();
        Truck truck = new Truck(rn2, c2, m2, cc2, f2);
        truck.displayDetails();
        Motorcycle motorcycle = new Motorcycle(rn3, c3, m3, cc3, f3);
        motorcycle.displayDetails();
```

```
    }
}
```

Output:

```
Enter the Registration Number, Color, Make, CC and fuel type of Car:
sgsgagas
Black
Tata
2000
Nexxon
Enter the Registration Number, Color, Make, CC and fuel type of Truck:
savgassdv
Red
Tata
5000
Bull
Enter the Registration Number, Color, Make, CC and fuel type of Motorcycle:
sdgfass
White
TVS
125
Jupiter
Registration number: sgsgagas
Color: Black
Type of vehicle: Car
Make: Tata
CC: 2000
Fuel type: Nexxon

Registration number: savgassdv
Color: Red
Type of vehicle: Truck
Make: Tata
CC: 5000
Fuel type: Bull

Registration number: sdgfass
Color: White
Type of vehicle: Motorcycle
Make: TVS
CC: 125
Fuel type: Jupiter

PS C:\SOURAV\CODE\Java\Java Assignments>
```

Conclusion: Hence we studied and implemented the concept of inheritance with java programming.

FAQs

Name : Sourav Shailesh Toshniwal.
Class : SY-1 (B)
Roll no. : 2213047
Subject: JPL

Assignment- 3

1) Explain Inheritance ?

Ans Inheritance is a mechanism in which one object aquires all the properties and behaviour of a parent object. The idea behind inheritance is that you can create new classes that are build upon existing class.

2) Advantages of Inheritance ?

Ans - Code Reusability : It allows sharing of the common code along other subclasses.
- Flexibility: As you will adjust only in one place, and the rest of the code will work smoothly.
- Overriding: You can override the methods of the base class.
- Data Hiding: The base class decides which data to be kept private, so it cannot be altered.

3) What is parent class and child class ?

Ans The class which inherits the properties of other is known as child class and the class whose properties are inherited is called as parent class.

4) Why does java not support multiple inheritance ?

Ans Java doesn't support multiple inheritances in classes because it

can lead to diamond problem and rather than providing some complex way to solve it, there are better ways through which we can achieve the same result as multiple inheritances.

5) Explain different types of inheritances?

Ans i) Single inheritance : Single child class inherits properties from a single parent class.

ii) Multilevel inheritance: It is a inheritance where a child class inherits from a class, which inherits properties from another parent class.

iii) Hierarchical inheritance: A parent class has many child classes inheriting its properties.

iv) Hybrid inheritance - It is combination of hierarchical and multiple inheritance.

Assignment No. 4

Assignment Title: A java program to demonstrate the concept of Class and Objects.

Aim: Write a java program to create a class Student with data name, city and age along with method addData and printData to input and display the data. Create the two objects s1, s2 to declare and access the values.

Pre-Requisites: C/C++

Objective: The objective is to impart fundamentals of object-oriented programming in Java, including defining classes, invoking methods, using class libraries, etc.

Outcomes: After learning these concept students will be able to,

1. Analyze the necessity for Object Oriented Programming paradigm over structured programming and become familiar with the fundamental concepts in OOP like encapsulation, Inheritance and Polymorphism

2. Design and develop java programs with methods and access the methods by using instances of class i.e. objects.

3. Understand the concept of Packages.

Theory:

CLASS:

Java is an object-oriented programming language. Everything in Java is associated with classes and objects, along with its attributes and methods. For example: in real life, a car is an object. The car has **attributes**, such as weight and colour, and **methods**, such as drive and brake.

A class is a group of objects which have common properties. It is a template or blueprint from which objects are created. It is a logical entity. It can't be physical.

OBJECTS IN JAVA

An entity that has state and behavior is known as an object e.g., chair, bike, marker, pen, table, car, etc. It can be physical or logical (tangible and intangible). The example of an intangible object is the banking system.

An object has three characteristics:

State: represents the data (value) of an object.

Behavior: represents the behavior (functionality) of an object such as deposit, withdraw, etc.

Identity: An object identity is typically implemented via a unique ID. The value of the ID is not visible to the external user. However, it is used internally by the JVM to identify each object uniquely.

An object is an instance of a class. A class is a template or blueprint from which objects are created. So, an object is the instance(result) of a class.

Object Definitions:
An object is *a real-world entity*.
An object is *a runtime entity*.
The object is *an entity which has state and behavior*.
The object is *an instance of a class*.

METHODS IN JAVA

In Java, a method is like a function which is used to expose the behaviour of an object.

Advantage of Method

Code Reusability

Code Optimization

OBJECT AND CLASS EXAMPLE: main Within the Class

In this example, we have created a Student class which has two data members id and name. We are creating the object of the Student class by new keyword and printing the object's value.

Here, we are creating a main() method inside the class.

File: Student.java

```
//Defining a Student class.
class Student
{
    //defining fields
    int id;    //field or data member or instance variable
    String name;
     public static void main(String args[]) //creating main method inside the Student class
    {
            Student s1=new Student();//creating an object of Student
                    //Printing values of the object
            System.out.println(s1.id);//accessing member through reference variable
            System.out.println(s1.name);
    }
}
```

Packages in JAVA

A package in Java is used to group related classes. Think of it as **a folder in a file directory**. We use packages to avoid name conflicts, and to write a better maintainable code. Packages are divided into two categories:

Built-in Packages (packages from the Java API)

User-defined Packages (create your own packages)

Built-in Packages

The Java API is a library of prewritten classes, that are free to use, included in the Java Development Environment.

The library contains components for managing input, database programming, and much more. The complete list can be found at Oracles website: https://docs.oracle.com/javase/8/docs/api/.

The library is divided into **packages** and **classes**. Meaning you can either import a single class (along with its methods and attributes), or a whole package that contain all the classes that belong to the specified package.

To use a class or a package from the library, you need to use the import keyword:

```
import package.name.Class;  // Import a single class
import package.name.*;  // Import the whole package
```

Pre-defined packages

java.applet	java.awt	java.beans
java.io	java.lang	java.lang.ref
java.math	java.net	java.nio
java.sql	java.text	java.util
java.util.zip	javax.sql	javax.swing

Algorithm:

Step 1: Start

Step 2: Declare and Define Class Student with required variables and methods addData and printData.

Step 3: Define main method

Step 4: Create objects of class Student S1, S2

Step 5: Access the methods with S1, S2.

Step 6: Stop

Code:

```
import java.util.Scanner;



class Student {

    private String name;

    private String city;

    private int age;

    Scanner scanner = new Scanner(System.in);



    public void addData() {

        System.out.println("Enter the Name, City and Age of student: ");

        String name = scanner.nextLine();

        String city = scanner.nextLine();

        int age = scanner.nextInt();

        this.name = name;

        this.city = city;

        this.age = age;

    }



    public void printData()

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

        this.name);
```

```
        System.out.println("City: " + this.city);
        System.out.println("Age: " + this.age);
    }
}


public class Assignment4 {
    public static void main(String[] args)
        {Student s1 = new Student();
        s1.addData();
        Student s2 = new Student();
        s2.addData();


        System.out.println("Details of Student 1:");
        s1.printData();
        System.out.println("Details of Student 2:");
        s2.printData();
    }
}
```

Output:

```
PS C:\SOURAV\CODE\Java\Java Assignments> cd "c:\SOURAV\CODE\Java\Ja
Enter the Name, City and Age of student:
Sourav
Ahmednagar
20
Enter the Name, City and Age of student:
Naman
Ahmedabad
18
Details of Student 1:
Name: Sourav
City: Ahmednagar
Age: 20
Details of Student 2:
Name: Naman
City: Ahmedabad
Age: 18
PS C:\SOURAV\CODE\Java\Java Assignments>
```

Conclusion:

Thus we implemented a Java program that accepts student data and display with the help of objects.

FAQ's:

MIT SCHOOL OF ENGINEERING
Rajbaug, Loni-Kalbhor, Pune

MIT-ADT UNIVERSITY
PUNE, INDIA

Name : Sourav Tashniwal
Class : SY-1 (B)
Roll No : 2213047
Subject : IPL

Assignment- 4

1) What are the access specifiers / modifiers that can be added to member of a class?

Ans. i) Public : Members declared within the public access modifier can be accessed from anywhere in the program.
ii) Private : Members declared with the private access modifier can only be accessed within the class.
iii) Protected : Members declared with the protected access modifier can be accessed within the same class as well as in subclass & classes in the same package.
iv) Default : Members declared with the default access modifier can be accessed within the same class and classes in the same package, but not in subclasses.

2) Differentiate between Abstract class and Interface?

Ans.

Abstract Class	Interface
- Doesn't support multiple inheritance	- Support multiple inheritance
- Contains data members.	- Does not contain data members
- Contains Constructors	- Does not contain constructors.
- Only Complete members can be static	- Member of interface cannot be static.

3) Write the difference between constructors and methods?

Ans

Constructors	Methods
- It is used to intialize the state of an object.	- It is used to expose behaviour of an object.
- Constructor must not have return type.	- Methods must have return type.
- Constructor is invoked implicitly	- Method is invoked explicitly.
- Constructor name must be same as the class name.	- Method name may or may not be same as class name

4) What is interface in Java?

Ans An interface is java is a blueprint for a class that defines a set of methods that can be used by multiple classes, regardless of their implementation. Interfaces cannot be instantiated and are implemented by classes uses the "implements" keyword.

5) What are the predefined packages in Java?

Ans i) java.lang: Provides classes for fundamental data types, strings, arrays and other common data type and utilities

ii) java.util: Provides utility classes for data structures, random number generation and other common utility functions.

iii) java.net: Provides classes for working with networks protocols, such as sockets, URLs and datagrams.

Assignment No. 05

Assignment Title:

Multithreading in java

Aim:

Write a java program that implements a multi-thread application that has three threads. First thread generates a random integer every 1 second and if the value is even, the second thread computes the square of the number and prints. If the value is odd, the third thread will print the value of the cube of the number.

Pre-Requisites: C/C++

Objective:

To achieve multithreading in java by implementing Runnable Interface or extending the Thread class.

Outcomes:

Students are able to Understand and Apply Multithreading concepts with java programming

Theory:

Multithreading is a Java feature that allows concurrent execution of two or more parts of a program for maximum utilization of CPU. Each part of such program is called a thread. So, threads are light-weight processes within a process.

Threads can be created by using two mechanisms:

1. Extending the Thread class
2. Implementing the Runnable Interface

Thread creation by extending the Thread class

We create a class that extends the java.lang.Thread class. This class overrides the run() method available in the Thread class. A thread begins its life inside

run() method. We create an object of our new class and call start() method to start the execution of a thread. Start() invokes the run() method on the Thread object.

```
// Java code for thread creation by extending

// the Thread class

class MultithreadingDemo extends Thread

    {public void run()

    {

        try {

            // Displaying the thread that is running

            System.out.println("Thread " + Thread.currentThread().getId() + " is
                                                    running");

        }

        catch (Exception e) {

            // Throwing an exception

            System.out.println("Exception is caught");

        }

    }

}

 // Main Class

public class Multithread {

    public static void main(String[] args)

    {

        int n = 8; // Number of threads
```

```
        for (int i = 0; i < n; i++) {
            MultithreadingDemo object = new MultithreadingDemo();
            object.start();
        }
    }
}
```

Output:

Thread 15 is running

Thread 14 is running

Thread 16 is running

Thread 12 is running

Thread 11 is running

Thread 13 is running

Thread 18 is running

Thread 17 is running

Thread creation by implementing the Runnable Interface

We create a new class which implements java.lang.Runnable interface and override run() method. Then we instantiate a Thread object and call start() method on this object.

```
// Java code for thread creation by implementing

// the Runnable Interface

class MultithreadingDemo implements Runnable

    {public void run()

    {

        try {

            // Displaying the thread that is running

                System.out.println( "Thread " + Thread.currentThread().getId()

                                                + " is running");

        }

        catch (Exception e) {

            // Throwing an exception

            System.out.println("Exception is caught");

        }

    }

}

 // Main Class

class Multithread {

    public static void main(String[] args)

    {
```

```
        int n = 8; // Number of threads
        for (int i = 0; i < n; i++) {
            Thread object = new Thread(new MultithreadingDemo());
            object.start();
        }
    }
}
```

Output

Thread 13 is running

Thread 11 is running

Thread 12 is running

Thread 15 is running

Thread 14 is running

Thread 18 is running

Thread 17 is running

Thread 16 is running

Algorithm/Steps:

1. Start the program
2. Create a class with the name “even implements Runnable” and “odd implementsRunnable”.

3. Create thread objects and Random class objects.

4. Pass the objects of our class to the Thread class.

5. Call the start method.

Code:

```
class FirstThread implements Runnable
    {@Override
    public void run()
        {while (true) {
            int number = (int) (Math.random() * 100 + 1);
            System.out.println("Generated number: " + number);
            if (number % 2 == 0) {
                new Thread(new SecondThread(number)).start();
            } else {
                new Thread(new ThirdThread(number)).start();
            }
            try
                { Thread.sleep(1000)
                ;
            } catch (InterruptedException e)
                {e.printStackTrace();
            }
        }
    }
}
```

```
class SecondThread implements Runnable
    {private int number;


    public SecondThread(int number)
        {this.number = number;
    }


    @Override
    public void run() {
        System.out.println("Square of " + this.number + " is " +
(this.number * this.number));
    }
}


class ThirdThread implements Runnable
    {private int number;


    public ThirdThread(int number)
        {this.number = number;
    }


    @Override
    public void run() {
```

```
        System.out.println("Cube of " + this.number + " is " +
(this.number * this.number * this.number));
    }
}


public class Assignment5 {
    public static void main(String[] args)
        { new Thread(new
        FirstThread()).start();
    }
}
```

Output:

```
PS C:\SOURAV\CODE\Java\Java Assignments> cd "c:\SOU
Generated number: 99
Cube of 99 is 970299
Generated number: 53
Cube of 53 is 148877
Generated number: 37
Cube of 37 is 50653
Generated number: 90
Square of 90 is 8100
Generated number: 25
Cube of 25 is 15625
Generated number: 49
Cube of 49 is 117649
```

Conclusion:

Hence, we studied multithreading in java.

Frequently Asked Questions:

MIT SCHOOL OF ENGINEERING
Rajbaug, Loni-Kalbhor, Pune

MIT-ADT UNIVERSITY
PUNE, INDIA

Name: Sourav Shailesh Toshniwal
Class: SY-1 (B)
Roll no: 2213047
Subject: JPL.

Assignment - 5.

1. Define Thread class and Runnable interface?

Ans Thread class provide constructors and methods to create and perform operations on a thread. The Runnable interface should be implemented by any class whose instances are intended to be executed by a thread.

2. What are thread methods?

Ans These are some thread methods:
- start() - starts the thread.
- getState() - It returns the state of thread.
- sleep() - stop the thread for the specified time.
- Thread methods are methods performed on the thread, to make it easier to use.

3. What is alive method?

Ans The isAlive() method of thread class tests if the thread is alive. A thread is considered alive when the start() method of thread class has be called and the thread is not yet dead.

4. What is thread interrupt?

Ans It is an indication to a thread that it should stop what it is doing and do something else. It's up to the programmer

to decide exactly how a thread responds to an interrupt.

5. What are the states associated in the thread?

Ans - NEW: A thread that hasn't yet started is in this state.
- RUNNABLE : A thread executing in the java virtual machine is in this state.
- BLOCKED: A thread that is blocked waiting for a monitor lock is in this state.
- WAITING: A thread that is waiting indefinitely for another thread to perform a particular action is in this state.
- TIMED WAITING: A thread that is waiting for another thread to perform an action for up to a specified waiting time is in this state.
- TERMINATED: A thread that has exited is in this state.

6. What is the difference between process and thread?

Ans

Process	Thread
- It means any program is in execution	- Thread means a segment of a process.
- It takes more time for creation	- It takes less time for creation
- The process is isolated	- Threads share memory
- The process doesn't share data with each other.	- Threads share data with each other.

Assignment No. 06

Assignment Title:

Use of Abstract class

Aim:

Write a java program to create an abstract class named Shape that contains two integers and an empty method named printArea(). Provide three classes named Rectangle, Triangle and Circle such that each one of the classes extends the class Shape. Each one of the classes contains only the method printArea() that prints the area of the given shape.

Pre-Requisites: C/C++

Objective:

To learn how to write abstract class and use of abstract class along with the understanding of the types of classes present in java.

Outcomes:

Students are able to Understand and Apply abstract class concept in java.

Theory:

The class is a blueprint from which we can create an individual object. Java provides a keyword named class by which we can declare a class. Inside the class, we define class members and functions. It is not possible to create Java programs without class. We can also refer to a class as a user-defined data type because an object-oriented paradigm allows us to model real-world objects.

Abstraction:

Abstraction is a process of hiding the implementation details and showing only functionality to the user.

Another way, it shows only essential things to the user and hides the internal details, for example, sending SMS where you type the text and send the message. You don't know the internal processing about the message delivery.

There are two ways to achieve abstraction in java

1. Abstract class
2. Interface

Abstract Class:

A class which is declared as abstract is known as an abstract class. It can have abstract and non-abstract methods. It needs to be extended and its method implemented. It cannot be instantiated.

Points to Remember:

1. An abstract class must be declared with an abstract keyword.
2. It can have abstract and non-abstract methods.
3. It cannot be instantiated.
4. It can have constructors and static methods also.
5. It can have final methods which will force the subclass not to change the body of the method.

Abstract class syntax:

```
abstract class Animal

{
}
```

Abstract Method in Java:

```
1.
2. abstract void printData();//no method body and abstract
3.
```

Example:

```
package bbatchsy1;

abstract class Animal
{
abstract void drinking();
static void walking()
{
System.out.println("Animal walking");
}
final void sleeping()
{
System.out.println("Animal Sleeping");
}

}

class Dog extends Animal
{

@Override
void drinking() {
```

```
System.out.println("Dog drinking");

}
/*Can not override final method
void sleeping()
{
System.out.println("Dog sleeping");
}
can not override static method
void walking ()
{
System.out.println("Dog walking");
}*/


}

public class Calling {

public static void main(String[] args)
 {Dog d=new Dog();
 d.drinking();
 d.sleeping();
// d.walking();  Wrong
 Animal.walking();
```

```
}

}
```

Output:

```
Dog drinking
Animal Sleeping
Animal walking
```

Algorithm/Steps:

1. Create an abstract class Shape.
2. Declare two members of type integer.
3. Declare method name as printArea()
4. Declare three classes Rectangle, Triangle and Circle who extends Superclass Shape.
5. In SubClasses override the method printArea() which calculates the area of a given shape.
6. Ask the user to enter the shape to calculate the area, and use the switch case for choice.
7. Print area for given shape.

Code:

```
import java.util.Scanner;


abstract class Shape {
```

```
    protected int width;
    protected int height;
    public abstract void printArea();
}


class Rectangle extends Shape {
    public Rectangle(int width, int height)
        {this.width = width;
        this.height = height;
    }

    public void printArea() {
        int area = width * height;
        System.out.println("Rectangle Area: " + area);
    }
}


class Triangle extends Shape {
    public Triangle(int width, int height)
        {this.width = width;
        this.height = height;
    }

    public void printArea() {
        int area = (width * height) / 2;
```

```
        System.out.println("Triangle Area: " + area);
    }
}


class Circle extends Shape
    {protected int radius;


    public Circle(int radius)
        {this.radius = radius;
    }

    public void printArea() {
        double area = Math.round(Math.PI * radius * radius);
        System.out.println("Circle Area: " + area);
    }
}


public class Assignment6 {
    public static void main(String[] args)
        {Scanner sc = new Scanner(System.in);
        System.out.println("Which shape's area do you want to
calculate? ");
        System.out.println("1. Rectangle\n2. Triangle\n3.
Circle.\nEnter your choice: ");
        int choice = sc.nextInt();
```

```
        int width, height, radius;
        switch (choice) {
            case 1:
                System.out.println("Enter the width and height of
rectangle: ");
                width = sc.nextInt();
                height = sc.nextInt();
                Shape rectangle = new Rectangle(width, height);
                rectangle.printArea();
                break;


            case 2:
                System.out.println("Enter the width and height of
triangle: ");
                width = sc.nextInt();
                height = sc.nextInt();
                Shape triangle = new Triangle(width, height);
                triangle.printArea();

            case 3:
                System.out.println("Enter the radius of circle: ");
                radius = sc.nextInt();
                Shape circle = new Circle(radius);
                circle.printArea();


            default:
```

```
                break;
            }
        }
}
```

Output:

```
PS C:\SOURAV\CODE\Java\Java Assignments> cd "c:\SOURAV\CO
Which shape's area do you want to calculate?
1. Rectangle
2. Triangle
3. Circle.
Enter your choice:
1
Enter the width and height of rectangle:
20
12
Rectangle Area: 240
PS C:\SOURAV\CODE\Java\Java Assignments>
```

Conclusion:

Hence, we studied Abstract class in java.

Frequently Asked Questions:

Name: Sourav Shailesh Toshniwal
class: SY CSE-1 (B)
Roll no.: 2213047
Subject: JPL.

Assignment - 6

1. What is the difference between abstract class and concrete class?

Ans 1. Abstract class can have both an abstract as well as concrete methods where concrete class can only have concrete methods.
2. Abstract class can not be instantiated using new keyword whereas concrete class can be instantiated.
3. Abstract class may or may not have abstract methods whereas concrete class cannot have abstract method.
4. Abstract class cannot be declared as a final class whereas concrete class can be declared as final class.
5. Abstract class declared using abstract keyword whereas concrete class don't have any keyword.

2. Explain Abstraction and Encapsulation.

Ans - Abstraction is a design level process which is concerned about what a class instance can do, instead of the implementation of the class.
- Encapsulation is an implementation level process which helps in data binding and control over maintaining the transparency of the data.

3. What is the difference between static and instance variable?

Ans 1) Static variable is property of a class whereas instance variable is property of an instance.
2) A static variable is created only once when the classloader loads the class whereas instance variable is created everytime an instance is created.
3) Static variable is used when you want to store a value that represents all the instances like count, sum, average etc whereas instance variable is used to store a value that represents property of single instance.

4. Write a Java program to create a vehicle having two abstract methods getData() and printData(). Write two classes TwoWheeler and FourWheeler which extends the class vehicle having its own attribute color and price. Get choice from the user and accordingly get and print data for two wheeler and four wheeler.

Ans
```
import java.util.Scanner;
abstract class Vehicle{
    abstract void getData();
    abstract void printData();
}
Class TwoWheeler extends vehicle{
    String color;
    int price;
    @override
    void getData(){
```

```
    Scanner sc = new Scanner (System.in);
    system.out.println ("Enter the color of two wheeler: ");
    color = sc.nextline ();
    system.out.println ("Enter the price of two wheeler:");
    price = sc.nextInt ();
  }

  @override
  void printData () {
    system.out.println ("Color of two wheeler is: "+color);
    system.out.println ("Price of two wheeler is: " + price);
  }
}

Class FourWheeler extends Vehicle {
  String color;
  int price;
  @Override
  void getData() {
    Scanner sc = new Scanner (System.in);
    System.out.println ("Enter the color of four wheeler:");
    color = sc.nextline();
    System.out.println ("Enter the price of four wheeler: ");
    price = sc.nextInt ();
  }

  @override
  void printData() {
    System.out.println ("Color of four wheeler is : "+ color);
    System.out.println ("Price of four wheeler is : "+ price);
  }
}
```

```
public class Main{
    public static void main (string[] args){
            — // —
    }
}
```

5. What is the advantage of Abstract class in Java?

Ans- Abstract class in Java enables the best way to execute the process of data abstraction by providing the developers with the option of hiding the code implementation. It also presents the end-user with a template that explains the method involved.

Assignment No. 07

Assignment Title: create a calculator using single function add(), sub() and multi().

Aim: Write a java program to create a calculator which performs addition, subtraction and multiplication of numbers for different types like integer, float and complex numbers using single function add(),sub() and multi().

Pre-Requisites: C/C++ Programming

Objective: We will learn how to build a simple calculator using Java AWT. This calculator has some simple functionality like all the basic mathematical operations and some special *addon features* using Java Swing components to create a simple calculator with only +, -, /, * operations.

Outcomes:

For creating a calculator, we can have the following different sets of input and output.

1. To Perform Addition :

When the addition expression "98+102" is typed,

it is expected that the result is displayed as "98+102=200.0".

2. To Perform Subtraction :

When the subtraction expression "200.0-58.75" is typed,

it is expected that the result is displayed as "200.0-58.75=141.25".

3. To Perform Multiplication :

When an multiplication expression "141.25*20" is typed,

it is expected that the result is displayed as "141.25*20=2825.0".

4. To Perform Division : When the denominator is non-zero

When an division expression "2825.0/5" is typed,

it is expected that the result is displayed as "2825.0/5=565.0".

5. To Perform Division : When the denominator is zero

When an division expression "565.0/0" is typed,

it is expected that the error is displayed as "565.0/0=Zero Divison Error".

Theory:

AWT(Abstract Window Toolkit) is an API that helps in building GUI (Graphical User Interface) based java applications. GUI helps in user interactions using some graphics. It primarily consists of a set of classes and methods that are required for creating and managing the GUI in a simplified manner such as buttons,windows,frame,textfield,RadioButton etc

In this article we will use Java AWT components to create a simple calculator with only +, -, /, * operations.

methods used :

add(Component c) : adds component to container.

addActionListenerListener(ActionListener d) : add actionListener for specified component

setBackground(Color c) : sets the background color of the specified container

setSize(int a, int b) : sets the size of container to specified dimensions.

setText(String s) : sets the text of the label to s.

getText() : returns the text of the label.

Algorithm/Steps:

LOGIC PART -I

FOR NUMERIC BUTTON. if(e. getSource()==b1){ //b1 for number 1 zt=l1. ...
FOR AIRTHMETIC BUTTON. if(e. getSource()==badd){ //FOR ADDITION num1=Double. ...
FOR EQUALS BUTTON. if(e. getSource()==bcalc){ num2=Double. ...
FOR CLEAR BUTTON. if(e. ...
FOR BACKSPACE BUTTON.

LOGIC PART -II

1.FOR NUMERIC BUTTON

```
if(e.getSource()==b1){ //b1 for number 1
 zt=l1.getText();
  z=zt+"1";// 1 will merged at the end of the previous value
  l1.setText(z);
}
```

when any of the numeric button pressed
whatever value in label l1 will be stored in a variable zt and then concatenated with the corresponding number and and then displayed in the label l1
for NEGATIVE and DECIMAL PTS Button we did it similarly

2.FOR AIRTHMETIC BUTTON

```
if(e.getSource()==badd){  //FOR ADDITION
  num1=Double.parseDouble(l1.getText());
  z="";
  l1.setText(z);
  check=1; // 1 for the adddition
}
```

NOW here we store the value of label l1 into a variable num1 after converting into double type which will be technically 1st number
and then and set label l1 to null
we will just use a check variable for getting that this particular airthmetic button(here +) was clicked so we can do this operation in our = button

3.FOR EQUALS BUTTON

```
if(e.getSource()==bcalc){ num2=Double.par
  seDouble(l1.getText()); if(check==1)
    xd =num1+num2;
  if(check==2)
    xd =num1-num2;
  if(check==3)
    xd =num1*num2;
  if(check==4)
    xd =num1/num2;
  if(check==5)
```

```
    xd =num1%num2;
  l1.setText(String.valueOf(xd));
}
```

NOW again store the value of l1 into num2 variable which will be techincally 2nd number and then check the value of variable check and then do corresponding operation and after that display result in label l1

4.FOR CLEAR BUTTON

```
if(e.getSource()==bclr){num
  1=0;
  num2=0;
  check=0;
  xd=0;
  z="";
   l1.setText(z);
   }
```

here updated all the variable we use to its Default value 0
and set label l1 to null so that we can start our new calculation afterward

5.FOR BACKSPACE BUTTON

```
if(e.getSource()==bback){  // FOR  BACKSPACE
  zt=l1.getText();
  try{
    z=zt.substring(0, zt.length()-1);
    }catch(StringIndexOutOfBoundsException f){return;}
  l1.setText(z);
}
```

here just updates the value in l1 by removing last digits using substring function
and handled one StringIndexOutOfBoundsException which occur when our value in label is null and still pressing back Button

sample Output:

Program Explanation:

1. When any button on the calculator is clicked, the function actionPerformed is called. Get the button clicked using getActionCommand.

2. If the button clicked is a digit or decimal point, then either of the following is executed :

a) The digit is concatenated to the the second number if no operator has been encountered.

b) Otherwise, the digit is concatenated to the first number.

3. If the button clicked is the clear button, then clear the input field.

4. If the button clicked is the equals operator, then either of the following is executed :

a) If neither the first number nor the second number is empty, then compute the result and display it in the frame

b) Otherwise, clear the input field.

5. If any of the button {+,-,*,/} is clicked, then either of the following is executed :

a) If either the operator or the second number is empty, then set the button clicked as the operator.

b) Otherwise, compute the result and display it in the frame.

Code:

```
import java.awt.*;
import java.awt.event.*;



public class Assignment7 extends Frame
    {Assignment7()
    {
        setLayout(null);


        Label l1 = new Label("First Number");
        Label l2 = new Label("Second Number");
        add(l1);
        add(l2);


        TextField t1 = new TextField(15);
        TextField t2 = new TextField(15);
```

```
add(t1);
add(t2);

Button b1 = new Button("Addition");
Button b2 = new Button("Difference");
Button b3 = new Button("Multiplication");
Button b4 = new Button("Division");
add(b1);
add(b2);
add(b3);
add(b4);

Label l3 = new Label("Result");
TextField t3 = new TextField(15);
add(l3);
add(t3);

l1.setBounds(50,50,200,30);
l2.setBounds(50,125,200,30);
t1.setBounds(250,50,150,40);
t2.setBounds(250,125,150,40);
b1.setBounds(20,200,100,50);
b2.setBounds(125,200,100,50);
b3.setBounds(235,200,100,50);
```

```
b4.setBounds(345,200,100,50);
l3.setBounds(50,280,200,30);
t3.setBounds(250,280,150,40);


b1.addActionListener(new ActionListener()
    { public void actionPerformed(ActionEvent ae)
    {
        String n1 = t1.getText();
        String n2 = t2.getText();


        Float num1 = Float.parseFloat(n1);
        Float num2 = Float.parseFloat(n2);


        Float result;
        result = num1+num2;


        String ans = String.valueOf(result);
        t3.setText(ans);
    }
});


b2.addActionListener(new ActionListener()
    { public void actionPerformed(ActionEvent ae)
```

```
    {
        String n1 = t1.getText();
        String n2 = t2.getText();

        Float num1 = Float.parseFloat(n1);
        Float num2 = Float.parseFloat(n2);

        Float result;
        result = num1-num2;

        String ans = String.valueOf(result);
        t3.setText(ans);
    }
});

b3.addActionListener(new ActionListener()
    { public void actionPerformed(ActionEvent ae)
    {
        String n1 = t1.getText();
        String n2 = t2.getText();

        Float num1 = Float.parseFloat(n1);
        Float num2 = Float.parseFloat(n2);
```

```
        Float result;
        result = num1*num2;

        String ans = String.valueOf(result);
        t3.setText(ans);
    }
});

b4.addActionListener(new ActionListener()
    { public void actionPerformed(ActionEvent ae)
    {
        String n1 = t1.getText();
        String n2 = t2.getText();

        Float num1 = Float.parseFloat(n1);
        Float num2 = Float.parseFloat(n2);

        Float result;
        result = num1/num2;

        String ans = String.valueOf(result);
```

```
            t3.setText(ans);
        }
    });



    addWindowListener(new WindowAdapter()
        { public void windowClosing(WindowEvent
        we)
        {
            System.exit(0);
        }
    });


    setVisible(true);
    setSize(500,500);
    setTitle("My Calculator");
    setBackground(Color .orange);
}




public static void main(String[] args)
    {new Assignment7();
}
}
```

Output:

Conclusion: Thus we have studied calculators in java with the help of AWT/Swing with event handling and perform addition, subtraction and multiplication .

Frequently Asked Questions:

MIT SCHOOL OF ENGINEERING
Rajbaug, Loni-Kalbhor, Pune

MIT-ADT UNIVERSITY
PUNE, INDIA

Name: Sourav Shailesh Toshniwal
Class: SY. CSE-1 (B)
Roll. No.: 2213047
Subject: JPL

Assignment - 7

1. What is the main purpose of AWT controls?

Ans - AWT controls provide a set of GUI components for building Java applications.
- They are used to create windows, buttons, text fields, checkboxes, menus, and the other graphical elements that enable user interaction with the application.
- AWT controls are part of the Java Foundation Classes and are included in the JDK.
- They provide a ~~platform~~ platform-independent way to create GUIs that work on different OS. and provide a consistent look and feel.

2. Which package in AWT is used for event handling?

Ans The package in AWT used for event handling in Java is called the "java.awt.event" package. Here are some key points about this package.
- It provides classes and interfaces for handling events in AWT and Swing components.
- It contains various event listener interfaces, such as ActionListener, MouseListener, KeyListener, etc., which can be implemented by the application to handle different types of events.

- An event Listener is a class that implements one or more of these interfaces and registers itself with a component to receive events.

3. Which controls are supported by AWT in java?

Ans
1. Button
2. Checkbox
3. Choice
4. Label
5. List
6. Panel
7. Scrollbar
8. Text Field
9. Text Area
10. Menus
11. Menu bar
12. Dialog
13. Frame
etc.

4. How are the events handled in AWT?

Ans In AWT, events are handled using following steps.

1. Event source registration: The component that generates the event needs to register the event with the event listener using the addXXXListener() method.
2. Event listener implementation: The event listener is responsible for implementing the corresponding listener interface.
3. Event notification: When an event occurs, the event source generates the event and notifies the event listener by invoking the corresponding listener method.
4. Event handling: The event listener handles the event by performing the required action or updating the state of the application.
5. Event dispatching: In AWT, event dispatching is handled by the event dispatch thread (EDT), which is a separate thread responsible for handling all events and dispatching them to appropriate event listener.

5. What are the three main components of event handling?

Ans 1. Event Source: The component that generates the event, such as a button or text field.

2. Event Listener: The object that receives and handles the event, by implementing the corresponding listener interface, such as ActionListener, MouseListener, or KeyListener.

3. Event Object: The object that encapsulates the event, providing information about the event such as its type, timestamp and any related data.

6. What are the limitations of AWT?

Ans 1. Platform dependency: It's appearance & behaviour can vary between different platform.

2. Limited controls: AWT provides a limited set of controls for building GUI compared to Swing and JavaFX.

3. Poor Performance: AWT's synchronous event model can cause performance issues in high event volume application.

4. Lack of customization: Makes it difficult to create a unique look and feel for the application.

5. Limited support for multimedia: It's features such as audio and video playback, which can be a drawback for multimedia.

6. Security issues: AWT has known security vulnerabilities, making it less secure than modern GUI toolkits.

7. Why AWT is abstract?

Ans 1. It provides a set of abstract classes and interfaces that define the basic building blocks for creating GUI.

2. It does not provide an implementation for these classes and interfaces, leaving it up to the underlying platform.

3. This allows AWT to be platform-independant, as the implementations can vary between platforms while maintaining a consistent API.

8. What is Swing used for?

Ans Swing is Graphical User Interface toolkit used for creating desktop applications in Java. It is used for creating rich and customizable desktop applications with consistent look and feel across multiple platforms, implementing advanced GUI features, and supporting advanced graphics capabilities.

Assignment No. 8

Assignment Title: Create simple application of Java AWT in which show an awt component button by setting its placement and window frame size..

Aim: Write a Java program that accepts four integers from the user and prints equal if all four are equal, and not equal otherwise.

Pre-Requisites: C/C++

Objective: The objective is to impart fundamentals of Java AWT (Abstract Window Toolkit) an API to develop Graphical User Interface (GUI) or windows-based applications in Java.

Outcomes: After learning this concept students will be able to,

1. Analyze and Work with Frame class, Colour, Fonts and layout managers

2. Design and develop interface components- Labels, Button, Text Components, Check Box, Check Box Group, Choice, List Box, Panels – Scroll Panel, Menu, Scroll Bar.

Theory:

JAVA AWT

Java AWT (Abstract Window Toolkit) is an API to develop Graphical User Interface (GUI) or windows-based applications in Java.

Java AWT components are platform-dependent i.e. components are displayed according to the view of operating system. AWT is heavy weight i.e. its components are using the resources of underlying operating system (OS).

The java.awt package provides classes for AWT API such as TextField, Label, TextArea, RadioButton, CheckBox, Choice, List etc.

The AWT tutorial will help the user to understand Java GUI programming in simple and easy steps.Java Operator Precedence

Why AWT is platform independent?

Java AWT calls the native platform calls the native platform (operating systems) subroutine for creating API components like TextField, ChechBox, button, etc.

For example, an AWT GUI with components like TextField, label and button will have different look and feel for the different platforms like Windows, MAC OS, and Unix. The reason for this

is the platforms have different view for their native components and AWT directly calls the native subroutine that creates those components.

In simple words, an AWT application will look like a windows application in Windows OS whereas it will look like a Mac application in the MAC OS.

JAVA AWT HIERARCHY

The hierarchy of Java AWT classes are given below.

There are four types of containers in Java AWT:

Window

Panel

Frame

Dialog

WINDOW

The window is the container that have no borders and menu bars. You must use frame, dialog or another window for creating a window. We need to create an instance of Window class to create this container.

PANEL

The Panel is the container that doesn't contain title bar, border or menu bar. It is generic container for holding the components. It can have other components like button, text field etc. An instance of Panel class creates a container, in which we can add components.

FRAME

The Frame is the container that contain title bar and border and can have menu bars. It can have other components like button, text field, scrollbar etc. Frame is most widely used container while developing an AWT application.

USEFUL METHODS OF COMPONENT CLASS

Method	Description
public void add(Component c)	Inserts a component on this component.
public void setSize(int width,int height)	Sets the size (width and height) of the component.
public void setLayout(LayoutManager m)	Defines the layout manager for the component.
public void setVisible(boolean status)	Changes the visibility of the component, by default false.

Code:

```
import java.awt.*;


public class Assignment8 {

public static void main(String[] args)

    { Frame frame = new Frame("AWT

    Example");Button button = new

    Button("Click me");frame.add(button);

    frame.setSize(200, 200);

    frame.setVisible(true);

 }

}
```

Output:

Assignment No. 09

Assignment Title: Java Swing

Aim: Create a student table with fields roll number, name, percentage. Insert values in the table. Display all the details of the student table in a tabular format on the screen (using swing).

Pre-Requisites: C/C++

Objective: To understand creation of table using swing

Outcomes: To display student details

Theory:

Swing has about four times the number of User Interface [UI] components as AWT and is part of the standard Java distribution. By today's application GUI requirements, AWT is a limited implementation, not quite capable of providing the components required for developing complex GUI's required in modern commercial applications. The AWT component set has quite a few bugs and really does take up a lot of system resources when compared to equivalent Swing resources. Netscape introduced its Internet Foundation Classes [IFC] library for use with Java. Its Classes became very popular with programmers creating GUI's for commercial applications.

- Swing is a Set Of API (API- Set Of Classes and Interfaces)
- Swing is Provided to Design Graphical User Interfaces
- Swing is an Extension library to the AWT (Abstract Window Toolkit)
- Includes New and improved Components that have been enhancing the looks and Functionality of GUIs'
- Swing can be used to build(Develop) The Standalone swing GUI Apps Also as Servlets And Applets
- It Employs model/view design architecture
- Swing is more portable and more flexible than AWT, The Swing is built on top of the AWT
- Swing is Entirely written in Java
- Java Swing Components are Platform-independent And The Swing Components are lightweight
- Swing Supports a Pluggable look and feels And Swing provides more powerful components
- such as tables, lists, Scrollpanes, Colourchooser, tabbedpane, etc
- Further Swing Follows MVC.

Many programmers think that JFC and Swing are one and the same thing, but that is not so.

JFC contains Swing [A UI component package] and quite a number of other items:

- Cut and paste: Clipboard support
- Accessibility features: Aimed at developing GUI's for users with disabilities
- The Desktop Colors Features Has been Firstly introduced in Java 1.1
- Java 2D: it has Improved colors, images, and also texts support

Features Of Swing Class

- Pluggable look and feel
- Uses MVC architecture
- Lightweight Components
- Platform Independent
- Advanced features such as JTable, JTabbedPane, JScollPane, etc.
- Java is a platform-independent language and runs on any client machine, the GUI look and feel, owned and delivered by a platform-specific O/S, simply does not affect an application's GUI constructed using Swing components

- **Lightweight Components:** Starting with the JDK 1.1, its AWT-supported lightweight component development. For a component to qualify as lightweight, it must not depend on any non-Java [O/s based) system classes. Swing components have their own view supported by Java's look and feel classes.
- **Pluggable Look and Feel:** This feature enables the user to switch the look and feel of Swing components without restarting an application. The Swing library supports components' look and feels that remain the same across all platforms wherever the program runs. The Swing library provides an API that gives real flexibility in determining the look and feel of the GUI of an application
- **Highly customizable –** Swing controls can be customized in a very easy way as visual appearance is independent of internal representation.
- **Rich controls**– Swing provides a rich set of advanced controls like Tree TabbedPane, slider, colorpicker, and table controls.

Swing Classes Hierarchy

The MVC Connection

- In general, a visual component is a composite of **three distinct aspects:**
  1. The way that the component looks when rendered on the screen
  2. The way such that the component reacts to the user
  3. The state information associated With the component
- Over the years, one component architecture has proven itself to be exceptionally effective:- **Model-View-Controller** or **MVC** for short.
- In MVC terminology, the **model** corresponds to the state information associated with the Component
- The **view** determines how the component is displayed on the screen, including any aspects of the view that are affected by the current state of the model.
- The **controller** determines how the component reacts to the user

The simplest Swing components have capabilities far beyond AWT components as follows:

- Swing buttons and labels can be displaying images instead of or in addition to text
- The borders around most Swing components can be changed easily. For example, it is easy to put a 1 pixel border around the outside of a Swing label
- Swing components do not have to be rectangular. Buttons, for example, can be round

- Now The Latest Assertive technologies such as screen readers can easily get information from Swing components. For example: A screen reader tool can easily capture the text that is displayed on a Swing button or label

Algorithm/Steps:

Step 1: Import necessary packages

Step 2: Create a window (JFrame)

Step 3: Create GUI Components (JLabel, JTextField and JButton)

Step 4: Create table and connect to jdbc driver

Step 5: execute sql query to insert and display records

Conclusion: Thus by performing this assignment we have successfully created table, inserted data and displayed output using swing

Frequently Asked Questions:

1) What is java swing?
2) How to create a window using the JFrame class.
3) How to use label using the JLabel class.
4) How to use text field using the JTextField class.
5) How to use button using the JButton class.
6) How to arrange components on the frame using FlowLayout - a simple layout manager.
7) How to handle the click event of a button.
8) How to show a message box using the JOptionPane class.
9) How to do connectivity with JDBC driver
10) How to use button using the `JTable` component.

Additional Problem statements

1) Create an employee table with fields emp_id, name, salary. Insert values in the table. Display all the details of the employee table in a tabular format on the screen (using swing).
2) Create a table marksheet with fields Student_enrollmentno,std _name,marks_subj1,marks_subj2,marks_subj3. Insert values in table. Display all the details of tis table on the screen as output

MIT SCHOOL OF ENGINEERING
Rajbaug, Loni-Kalbhor, Pune

MIT-ADT UNIVERSITY PUNE, INDIA

Name: Sourav Shailesh Toshniwal.
Class : SY CSE-1 (B)
Roll no.: 2213047
Subject: JPL

Assignment - 9

1. What is java swing?

Ans Java swing is GUI toolkit that allows developers to create destop applications with a consistent look and feel across multiple platforms. It provides a set of components that can be easily customized to create sophisticated and interactive user interfaces. It can be imported into project by:

```
import javax.swing.*;
```

2. How to create a window using the Jframe class.

Ans

```
import Javax.swing.Jframe;      //importing in project.
public class MyWindow {
    public static void main (String[] args){
        JFrame frame = new JFrame();   //create object.
        frame.setTitle("My Window");   //Game title to frame
        frame.setSize(500,500);        // set size.

        frame.setVisible(true);   //, By default it is false so we have to change it to true.

    }
```

3. How to use label using the JLabel class.

Ans : import javax.swing.JFrame;
import javax.swing.JLabel;
public class MyWindow{
public static void main (String[] args){
JFrame frame = new JFrame ();
JLabel label = new JLabel("Hello, world!");
frame.add (label);
label.setBounds (50, 50, 200, 50);
frame.setSize (300, 200);

frame.setVisible (true);
}
}

4. How to use text field using the JTextfield class.

Ans : Pseudocode : (only main function).
public static void maid (String[] args){
JFrame frame = new JFrame ();
JTextfield textfield = new JTextfield();
textfield.setBounds(50, 50, 200, 30);
frame.add(textfield);
frame.setSize (300, 150);

frame.setVisible (true);
}.

5. How to use button using the JButton class.

Ans : import javax.swing.JButton; //import.
JButton myButton = new JButton("Click me!); //Object
myButton.setBounds (50, 50, 100, 30);
myframe.add (myButton); // added to frame.
myButton.addactionlistener (new ActionListener() {
public void actionPerformed (ActionEvente) {
} // function to perform action after clicking.
};

6. How to arrange components on the frame using FlowLayout - a simple layout manager.

Ans 1. Create a JFrame object: Create a window that serves as main container to hold components.
2. Create a JPanel object and set its layout to FlowLayout using the constructor or setLayout() method.
3. Create your components and instanticate them using the appropriate constructor of the components class.
4. Add your components to the panel using the add() method of the container to add components to the panel.

7. How to handle the click event of a button.

Ans 1. Create an ActionListener object by either implementing the ActionListener interface or using anonymous inner class.
2. Override the actionPerformed() method of ActionListener interface. This method will be called when button is clicked.
3. Inside the actionPerformed() method, write the code to perform the action you went to take when button is clicked.
4. Add the ActionListener object to the JButton using addActionListener() method.

8. How to show a message box using the JOptionPane class.

Ans - The JOptionPane class provides a convenient way to display standard dialog boxes such as message dialogs, confirmation dialogs, input & option dialogs.
- Syntax:

```
JOptionPane.showMessageDialog(null, "Hello, World!");
```

9. How to do connectivity with JDBC driver.

Ans: 1. Load the JDBC connectivity driver using forName() method.

2. Establish a connection to the database using the DriverManager.getConnection() method.

3. Create a statement object using the connection.createStatement method.

4. Execute SQL statements using the statement object.

5. Process the results using a ResultSet object returned by the statement.executeQuery() method.

6. Close the connection using the connection.close() method.

10. How to use button using the JTable component.

Ans: 1. Create a custom cell renderer: To use a JButton in a JTable component, you need to create a custom cell renderer that returns a JButton for the cells that need to have a button.

2. Create a custom cell editor: To handle button clicks, you need to create a custom cell editor that returns a JButton for the cells that need to have a button. The cell editor should listen for button clicks and perform the required actions.

3. Set the cell renderer and cell editor: Once you have created the custom cell renderer and cell editor, you need to set them for the appropriate columns in JTable components.

Code:

```java
import java.awt.Dimension;
import java.sql.*;
import javax.swing.*;


public class Assignment9 {
    public static void main(String[] args) {
        Connection conn = null;
        Statement stmt = null;
        ResultSet rs = null;
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/mydatabase", "root", "Sourav@45");
            stmt = conn.createStatement();


            String createTableSQL = "CREATE TABLE IF NOT EXISTS student ("
                    + "roll_number INT PRIMARY KEY,"
                    + "name VARCHAR(50),"
                    + "percentage FLOAT"
                    + ")";
            stmt.executeUpdate(createTableSQL);


            String insertDataSQL = "INSERT INTO student (roll_number, name, percentage) VALUES "
                    + "(1, 'John Doe', 85.5), "+ "(2, 'Jane Smith', 92.0), "+ "(3, 'Bob Johnson', 77.3), "
                    + "(4, 'Alice Lee', 91.8)";
            stmt.executeUpdate(insertDataSQL);


            String querySQL = "SELECT * FROM student";
            rs = stmt.executeQuery(querySQL);


            String[] columnNames = {"Roll Number", "Name", "Percentage"};
            Object[][] data = new Object[4][3];
            int i = 0;
            while (rs.next()) {
                data[i][0] = rs.getInt("roll_number");
                data[i][1] = rs.getString("name");
                data[i][2] = rs.getFloat("percentage");
                i++;
            }
            JTable table = new JTable(data, columnNames);


            JScrollPane scrollPane = new JScrollPane(table);
            scrollPane.setPreferredSize(new Dimension(400, 200));


            JFrame frame = new JFrame("Student Table");
            JPanel panel = new JPanel();
            panel.add(scrollPane);
            frame.add(panel);
            frame.pack();
            frame.setVisible(true);
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            try {
                if (rs != null) {
                    rs.close();
                }
                if (stmt != null) {
                    stmt.close();
                }
                if (conn != null) {
                    conn.close();
                }
            } catch (Exception e) {
```

```
                e.printStackTrace();
            }
        }
    }
}
```

Output:

Assignment No. 10

Assignment Title: To study basic concepts of java swing and java applets

Aim: Write a java program that simulates a traffic light. The program lets the user select one of three lights: red, yellow, or green with radio buttons. On selecting a button, an appropriate message with "stop" or "ready" or "go" should appear above the buttons in a selected color. Initially there is no message shown

Pre-Requisites: C/C++

Objective: To demonstrate the concept of event handling used in GUI.

Outcomes: Design an object oriented system, AWT components and multithreaded

processes as per needs and specifications.

Student will be able to use knowledge of interfaces, packages and exception handling in their projects.

Theory:

Java Swing is a part of Java Foundation Classes (JFC) that is *used to create window-based applications*. It is built on the top of AWT (Abstract Windowing Toolkit) API and entirely written in java.

Unlike AWT, Java Swing provides platform-independent and lightweight components.

The javax.swing package provides classes for java swing API such as JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser etc.

Swing has about four times the number of User Interface [UI] components as AWT and is part of the standard Java distribution. By today's application GUI requirements, AWT is a limited implementation, not quite capable of providing the components required for developing complex GUI's required in modern commercial applications. The AWT component set has quite a few bugs and really does take up a lot of system resources when compared to equivalent Swing resources. Netscape introduced its Internet Foundation Classes [IFC] library for use with Java. Its Classes became very popular with programmers creating GUI's for commercial applications.

- Swing is a Set Of API (API- Set Of Classes and Interfaces)
- Swing is Provided to Design Graphical User Interfaces
- Swing is an Extension library to the AWT (Abstract Window Toolkit)
- Includes New and improved Components that have been enhancing the looks and Functionality of GUIs'
- Swing can be used to build(Develop) The Standalone swing GUI Apps Also as Servlets And Applets
- It Employs model/view design architecture
- Swing is more portable and more flexible than AWT, The Swing is built on top of the AWT
- Swing is Entirely written in Java
- Java Swing Components are Platform-independent And The Swing Components are lightweight
- Swing Supports a Pluggable look and feels And Swing provides more powerful components
- such as tables, lists, Scrollpanes, Colourchooser, tabbedpane, etc
- Further Swing Follows MVC.

Many programmers think that JFC and Swing are one and the same thing, but that is not so.

JFC contains Swing [A UI component package] and quite a number of other items:

Hierarchy of Java Swing classes

Commonly used Methods of Component class

The methods of Component class are widely used in java swing that are given below

Method	Description
public void add(Component c)	add a component on another component.
public void setSize(int width,int height)	sets size of the component.
public void setLayout(LayoutManager m)	sets the layout manager for the component.
public void setVisible(boolean b)	sets the visibility of the component. It is by default false.

Java Swing Examples

There are two ways to create a frame:

- By creating the object of Frame class (association)
- By extending Frame class (inheritance)

Features Of Swing Class

- Pluggable look and feel
- Uses MVC architecture
- Lightweight Components
- Platform Independent
- Advanced features such as JTable, JTabbedPane, JScollPane, etc.
- Java is a platform-independent language and runs on any client machine, the GUI look and feel, owned and delivered by a platform-specific O/S, simply does not affect an application's GUI constructed using Swing components

Components of Swing Classthe task's percentage

Class	Description
Component	A Component is the Abstract base class for about the non menu user-interface controls of SWING. Components are represents an object with a graphical representation
Container	A Container is a component that can container SWING Components
JComponent	A JComponent is a base class for all swing UI Components In order to use a swing component that inherits from JComponent, component must be in a containment hierarchy whose root is a top-level Swing container
JLabel	A JLabel is an object component for placing text in a container
JButton	This class creates a labeled button
JColorChooser	A JColorChooser provides a pane of controls designed to allow the user to manipulate and select a color
JCheckBox	A JCheckBox is a graphical(GUI) component that can be in either an on-(true) or off-(false) state
JRadioButton	The JRadioButton class is a graphical(GUI) component that can be in either an on-(true) or off-(false) state. in the group
JList	A JList component represents the user with the scrolling list of text items
JComboBox	A JComboBox component is Presents the User with a show up Menu of choices
JTextField	A JTextField object is a text component that will allow for the editing of a single line of text
JPasswordField	A JPasswordField object it is a text component specialized for password entry
JTextArea	A JTextArea object s a text component that allows for the editing of multiple lines of text
ImageIcon	A ImageIcon control is an implementation of the Icon interface that paints Icons from Images
JScrollbar	A JScrollbar control represents a scroll bar component in order to

	enable users to Select from range values
JOptionPane	JOptionPane provides set of standard dialog boxes that prompt users for a value or Something
JFileChooser	A JFileChooser it Controls represents a dialog window from which the user can select a file.
JProgressBar	As the task progresses towards completion, the progress bar displays the tasks percentage on its completion
JSlider	A JSlider this class lets the user graphically(GUI) select by using a value by sliding a knob within a bounded interval.
JSpinner	A JSpinner this class is a single line input where the field that lets the user select by using a number or an object value from an ordered sequence

Algorithm/Steps:

Create a Button Group such that it contains three radio buttons Red, Yellow, Green when red is clicked you want to display "Stop", when orange is clicked you want to display "ready", when the green button is clicked Display "Go" and also create a sample Traffic signal demo using rectangle and ovals using 2D graphics.

Conclusion: Hence, we have successfully studied java swing and java applets

Frequently Asked Questions:

1. What is EDT thread in Swing?
2. What is difference between invoke And Wait and invoke Later in Java?
3. What is difference between Swing and AWT in Java?
4. Why Swing components are called lightweight components?
5. Write code for JTable with custom cell editor and custom cell renderer?
6. What is the difference between a scrollbar and a scrollpane?
7. What is a container in a GUI?
8. What is a layout manager and what are different types of layout managers available in Java Swing?
9. What is the difference between the paint() and repaint() methods?
10. How can a GUI component handle its own events?

MIT SCHOOL OF ENGINEERING
Rajbaug, Loni-Kalbhor, Pune

MIT-ADT UNIVERSITY
PUNE, INDIA

Name : Sourav Shailesh Toshniwal
Class : SY CSE - 1 (B)
Roll no. : 2213047
Subject : JPL

Assignment - 10.

1. What is EDT thread in Swing?

Ans - EDT (Event Dispatch Thread) processes all user interface events.
- All updates to Swing Components should be performed on the EDT to ensure thread safety.
- Long-running tasks on the EDT can cause the UI to become unresponsive.
- To prevent blocking the EDT, time-consuming tasks should be executed on a separate thread, such as a worker thread or background thread.
- Swing provides several classes to support concurrency, such as SwingWorker.

2. What is difference between invoke & wait and invoke later in Java?

Ans 1. invokeAndWait() is a synchronous method while invokeLater() is an asynchronous method.
2. invokeAndWait() blocks the calling thread when called from EDT itself while invokeLater() doesn't.
3. invokeAndWait() should be used sparingly whereas invokeLater() should be used often.

3. What is difference between Swing and AWT in Java.

Ans 1. Architecture: AWT is based on native platform components while swing is built entirely in Java.

2. Component set: AWT provides basic set of UI components while swing provides more advanced set of components.

3. Customization: Swing is more flexible than AWT.

4. Performance: AWT components are faster and more lightweight than swing components.

5. Compatibity: Since swing is built entirely in Java, it more compatible across different platforms than AWT.

4. Why Swing components are called lightweight components?

Ans - Because they are implemented entirely in Java and do not rely on the underlying native OS for rendering. Instead draws directly onto a Java graphics surface using Java code.

- This lightweigh approach allows greater flexibility and ease to use in creating GUI.

5. Write code for JTable with custom cell editor and custom cell renderer?

Ans main method:

```
public static void main (String[] args){
    JTable table = new JTable (3,2);
    TableColumn c1 = table.getColumnModel().getColumn(0);
    TableColumn c2 = table.getColumnModel().getColumn(1);
    c1.setCellRender(new CustomCellRenderer());
    c2.setCellEditor(new CustomCellEditor();
    JFrame frame = new JFrame();
    frame.add(new JScrollPane(table));
    frame.pack();
    frame.setVisible(true);
}
```

```
static class CustomRender extends DefaultTableCellRenderer {
    @Override
    public void setValue(Object value){
        setBackground(Color.Red);
        setText((String) value);
    }
}
static class CustomCellEditor extends AbstractCellEditor
implements TableCellEditor { JSpinner spinner = new JSpinner();
    @Override
    public Object getCellEditorValue(){
        return spinner.getValue();
    }
    @Override
    public Component getTableCellEditorComponent (JTable table,
    Object value, boolean isSelected, int row, int column){
        spinner.setValue(value);
        return spinner;
    }
}
}
```

6. What is difference between scrollbar and scrollpane?

Ans Scrollbar allows scrolling in one direction whereas scrollpane allows in both directions.

7. What is container in a GUI?

Ans In a GUI, a container is a component that holds and manages other components within it. Containers are used to organize and layout components in a logical and visually appealing way, and they can be nested within other containers to create more complex GUIs.

8. What is a layout manager and what are different types of layout managers available in Java Swing?

Ans 1. A layout manager determines the size & position of components within a container.

2. The Layout Manager is responsible for laying out components in logical and visually appealing way.
3. Layout managers take into account factors such as component size, available space, etc.
4. There are several types of layout manages including FlowLayout, BorderLayout, GridLayout, etc.

9. What is difference between paint() & repaint() methods?

Ans: - paint() is method that is called whenever a component needs to be painted/repainted while repaint() is called to request that a component be repainted.
- paint() is called automatically by swing framework while repaint() is called directly by programmer.
- paint() should be used to perform the actual drawing operations while repaint() should be used to trigger the repaint process.

10. How can a GUI component handle its own events?

Ans: 1. Choose a listener interface for the event that the component should handle.
2. Implement the listener methods within the component's class.
3. Register the component as a listener for the event using the appropriate addXXXListener() method.
4. Write the code to handle the event within the listener method.

Code:

```java
import java.awt.Color;
import java.awt.*;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import javax.swing.*;
class App extends JFrame implements ItemListener{
    JFrame actualWindow;
    JPanel messageContainer, lightsContainer;
    JLabel message;
    ButtonGroup btn_group;
    JRadioButton rb_red, rb_yellow, rb_green;
    App() {
        Font myFont = new Font("Verdana",Font.BOLD, 30);
        actualWindow = new JFrame("Traffic Lights");
        messageContainer = new JPanel();
        lightsContainer = new JPanel();
        message = new JLabel("Select Light");
        btn_group = new ButtonGroup();
        rb_red = new JRadioButton("Red");
        rb_yellow = new JRadioButton("Yellow");
        rb_green = new JRadioButton("Green");

        actualWindow.setLayout(new GridLayout(2, 1));

        message.setFont(myFont);
        rb_red.setForeground(Color.RED);
        rb_yellow.setForeground(Color.YELLOW);
        rb_green.setForeground(Color.GREEN);

        btn_group.add(rb_red);
        btn_group.add(rb_yellow);
        btn_group.add(rb_green);

        rb_red.addItemListener(this);
        rb_yellow.addItemListener(this);
        rb_green.addItemListener(this);

        messageContainer.add(message);
        lightsContainer.add(rb_red);
        lightsContainer.add(rb_yellow);
        lightsContainer.add(rb_green);

        actualWindow.add(messageContainer);
        actualWindow.add(lightsContainer);

        actualWindow.setSize(300, 200);
        actualWindow.setVisible(true);
    }
    @Override
    public void itemStateChanged(ItemEvent ie) {
        JRadioButton selected = (JRadioButton) ie.getSource();
        String textOnButton = selected.getText();
        if(textOnButton.equals("Red")) {
            message.setForeground(Color.RED);
            message.setText("STOP");
        } else if(textOnButton.equals("Yellow")) {
            message.setForeground(Color.YELLOW);
            message.setText("READY");
        } else {
            message.setForeground(Color.GREEN);
            message.setText("GO");
        }
    }
}
public class Assignment10 {



    public static void main(String[] args) {
        new App();
    }
}
```

Output:

Traffic Lights
STOP
Red
Yellow
Green

Assignment No. 11

Assignment Title: A java program that connects to a database using JDBC .

Aim: Write a java program that connects to a database using JDBC and does add, deletes, modify, and retrieve operations.

Pre-Requisites: C++

Objective: The objective of the JDBC API is **to provide a connection to a relational database through with SQL queries can be performed**, and the results from those queries can be processed within a Java program.

Outcomes: After learning this concept student able to understand JDBC. Students also able to implement JDBC which makes it possible to do **establish a connection with a data source, send queries and update statements, and process the results**

Theory:

Java JDBC

JDBC stands for Java Database Connectivity. JDBC is a Java API to connect and execute the query with the database. It is a part of JavaSE (Java Standard Edition). JDBC API uses JDBC drivers to connect with the database. There are four types of JDBC drivers:

- JDBC-ODBC Bridge Driver,
- Native Driver,
- Network Protocol Driver, and
- Thin Driver

We can use JDBC API to access tabular data stored in any relational database. By the help of JDBC API, we can save, update, delete and fetch data from the database. It is like Open Database Connectivity (ODBC) provided by Microsoft.

Algorithm/Steps:

There are 5 steps to connect any java application with the database using JDBC. These steps are as follows:

- Register the Driver class
- Create connection
- Create statement
- Execute queries
- Close connection

1) Register the driver class

The **forName()** method of Class is used to register the driver class. This method is used to dynamically load the driver class.

Syntax of forName() method

public static void forName(String className)**throws** ClassNotFoundException

Example to register the OracleDriver class

Here, Java program is loading oracle driver to establish database connection.Class.forName("oracle.jdbc.driver.OracleDriver");

2) Create the connection object

The **getConnection()** method of DriverManager class is used to establish connection with the database.

Syntax of getConnection() method

1) **public static** Connection getConnection(String url)**throws** SQLException

2) **public static** Connection getConnection(String url,String name,String password) **throws** SQLException

Example to establish connection with the Oracle database

```
Connection con=DriverManager.getConnection(
"jdbc:oracle:thin:@localhost:1521:xe","system","password");
```

3) Create the Statement object

The createStatement() method of Connection interface is used to create statement. The object of statement is responsible to execute queries with the database.

Syntax of createStatement() method

public Statement createStatement()**throws** SQLException

Example to create the statement object

```
Statement stmt=con.createStatement();
```

4) Execute the query

The executeQuery() method of Statement interface is used to execute queries to the database. This method returns the object of ResultSet that can be used to get all the records of a table.

Syntax of executeQuery() method

public ResultSet executeQuery(String sql)**throws** SQLException

Example to execute query

```
ResultSet rs=stmt.executeQuery("select * from emp");

while(rs.next()){
System.out.println(rs.getInt(1)+" "+rs.getString(2));
}
```

5) Close the connection object

By closing connection object statement and ResultSet will be closed automatically. The close() method of Connection interface is used to close the connection.

Syntax of close() method

public void close()**throws** SQLException

Example to close connection

```
con.close();
```

Program Code:

1. **Code to add records to database**

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.util.Scanner;

public class ShowRecords {

public static void main(String[] args) {
// TODO Auto-generated method stub

//1. Load the driver
try {
Class.forName("com.mysql.cj.jdbc.Driver");
System.out.println("Driver Loaded");
//2. create the connection
Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/studentdb","root","Chinmay@12
3");
System.out.println("Connection Established");
PreparedStatement pstmt=con.prepareStatement("insert into student values(?,?,?)");
//take the input from user
Scanner sc=new Scanner(System.in);
System.out.println("Enter First Name");
String fname=sc.next();
System.out.println("Enter last name");
String lname=sc.next();
System.out.println("Enter city ");
String city=sc.next();
```

```
//add into database
pstmt.setString(1, fname);
pstmt.setString(2, lname);
pstmt.setString(3, city);
int r=pstmt.executeUpdate();
if(r>0)
{
System.out.println("Record Inserted Successfully");
}
else
{
System.out.println("Not Inserted the records");
}
} catch (ClassNotFoundException e) {
// TODO Auto-generated catch block
e.printStackTrace();
} catch (SQLException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}

}

}
```

2.Code to modify record

```
package Mypackage;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
public class UpdateDemo
 {
```

```
public static void main(String[] args)
{
try
{

//Load driver
Class.forName("com.mysql.cj.jdbc.Driver");
System.out.println("Driver Loaded");
//2. create the connection
Connection
conn=DriverManager.getConnection("jdbc:mysql://localhost:3306/studentdb","root","Chinmay@123");
System.out.println("Connection Established");
//update data in database
String query = "update student set firstname = ?, lastname = ?  where city = ?";
PreparedStatement preparedStmt = conn.prepareStatement(query);
   preparedStmt.setString(1,"Akshay");
   preparedStmt.setString(2,"Kokate");
   preparedStmt.setString(3,"Nagpur");

   preparedStmt.executeUpdate();
   conn.close();
  }
  catch (Exception e)
  {
   System.err.println("Got an exception! ");
   System.err.println(e.getMessage());
  }
}
}
```

3.Program to retrieve the record

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

```
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class RetriewData {
public static void main(String[] args)
{
try
{

//Load driver
Class.forName("com.mysql.cj.jdbc.Driver");
System.out.println("Driver Loaded");
//2. create the connection
Connection
conn=DriverManager.getConnection("jdbc:mysql://localhost:3306/studentdb","root","Chinmay@123");
System.out.println("Connection Established");
Statement stmt=conn.createStatement();       ResultSet res=stmt.executeQuery("select * from student");

while(res.next())
{
System.out.println(res.getString(1)+" "+res.getString(2)+" "+res.getString(3));
}
conn.close();
} catch (ClassNotFoundException e) {
// TODO Auto-generated catch block
e.printStackTrace();
} catch (SQLException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}

}
}
```

4. Program to delete record

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
public class Deleteoperation {

public static void main(String[] args)
{
try
{

//Load driver
Class.forName("com.mysql.cj.jdbc.Driver");
System.out.println("Driver Loaded");
//2. create the connection
Connection
conn=DriverManager.getConnection("jdbc:mysql://localhost:3306/studentdb","root","Chinmay@123");
System.out.println("Connection Established");
//update data in database
String query = "delete from student where city = ?";
PreparedStatement preparedStmt = conn.prepareStatement(query);
   preparedStmt.setString(1,"Pune");
   preparedStmt.executeUpdate();
   conn.close();
  }
  catch (Exception e)
  {
   System.err.println("Got an exception! ");
   System.err.println(e.getMessage());
  }
}

}
```

Conclusion: Thus we implemented JDBC

Frequently Asked Questions:

1. What is JDBC
2. How to create connection
3. How to get connection
4. Write syntax for close subject
5. Write syntax to register driver

MIT SCHOOL OF ENGINEERING
Rajbaug, Loni-Kalbhor, Pune

MIT-ADT UNIVERSITY PUNE, INDIA

Name: Sourav Shailesh Toshniwal.
Class : SY CSE-1 (B)
Roll no.: 2213047
Subject : JPL

Assignment - II.

1. What is JDBC?

Ans JDBC stands for Java Database Connectivity. It is a Java API to connect and execute the query with the database. It is a part of Java Standard Edition (JavaSE)
JDBC drivers are of 4 types:
1. JDBC - ODBC Bridge Driver.
2. Native Driver
3. Network Protocol Driver.
4. Thin Driver.

2. How to create connection?

Ans There are 7 steps to create connection.
1. Import the packages.
2. Register the drivers.
3. Establish a connection.
4. Create a statement.
5. Execute the query.
6. Retrieve results.
7. Close the connections.

3. How to get connection?

Ans The steps for connecting JDBC are:

1. Insert the database you want to access.
2. Include the JDBC library.
3. Ensure the JDBC driver you need is on your classpath.
4. Use JDBC library to obtain connection.

4. Write syntax for close subject?

Ans: JDBC statements object must always be explicitly closed by calling the close method on the object.

Syntax: close ()

```
try
{ connection close();
} catch (Exception e){
}
```

5. Write syntax to register driver?

Ans: Using the register driver() method:

The registerDevice() method of the Driver manager class accepts an object of the class as a parameter it with the JDBC driver manager.

```
Driver myDriver = newcon.mysql.JDBC.Driver()
Driver Manager.registerDriver(my Driver);
```

Code:

```java
import java.sql.*;


public class Assignment11 {
    static final String JDBC_DRIVER = "com.mysql.cj.jdbc.Driver";
    static final String DB_URL = "jdbc:mysql://localhost/employees";



    static final String USER = "root";
    static final String PASS = "Sourav@45";



    public static void main(String[] args) {
        Connection conn = null;
        Statement stmt = null;
        try{
            Class.forName(JDBC_DRIVER);



            System.out.println("Connecting to database...");
            conn = DriverManager.getConnection(DB_URL,USER,PASS);



            System.out.println("Creating statement...");
            stmt = conn.createStatement();



            String sql = "INSERT INTO employees (first_name, last_name, age) " +
                         "VALUES ('John', 'Doe', 25)";
            stmt.executeUpdate(sql);



            sql = "DELETE FROM employees WHERE last_name='Doe'";
            stmt.executeUpdate(sql);



            sql = "UPDATE employees SET age = 30 WHERE last_name='Smith'";
            stmt.executeUpdate(sql);



            sql = "SELECT * FROM employees";
            ResultSet rs = stmt.executeQuery(sql);



            while(rs.next()){
                int id  = rs.getInt("id");
                String first = rs.getString("first_name");
                String last = rs.getString("last_name");
                int age = rs.getInt("age");



                System.out.print("ID: " + id);
                System.out.print(", First Name: " + first);
                System.out.print(", Last Name: " + last);
                System.out.println(", Age: " + age);
            }



            rs.close();
            stmt.close();
            conn.close();
        }catch(SQLException se){
            se.printStackTrace();
        }catch(Exception e){
```

```
            e.printStackTrace();
        }finally{
            try{
                if(stmt!=null)
                    stmt.close();
            }catch(SQLException se2){
            }
            try{
                if(conn!=null)
                    conn.close();
            }catch(SQLException se){
                se.printStackTrace();
            }
        }
        System.out.println("Goodbye!");
    }
}
```

Output:

```
Creating statement...

1 rows added.

1 rows updated.

1 rows deleted.

Goodbye!


Process finished with exit code 0
```