**PRACTICAL # 03**

**OBJECT:**

**Java Classes and Objects:**

**Stack Class Implementation**

**THEORY:**

Class is the first main concept of object-oriented programming. It encapsulates attributes and functions of in a single object.

# **Class Declaration**

A class is a blue print or template for objects to be created.

**class MyClass {**

**// field, constructor, and**

**// method declarations/definitions**

**}**

The class body contains: constructors, declarations for the fields that provide the state of the class objects, and methods to implement the behavior of the class and objects.

The preceding class declaration is a minimal one. It contains only those components of a class declaration that are required.

Modifiers like public or private can be added at the very beginning and determine what other classes can access MyClass, a topic of inheritance.

In general, class declarations can include these components, in order:

1. Modifiers such as public, private.
2. The class name, with the initial letter capitalized by convention.
3. The name of the superclass (optional), preceded by the keyword **extends**. A class can only extend one parent class.
4. A comma-separated list of interfaces implemented by the class (optional), preceded by the keyword implements. A class can implement more than one interface.
5. The class body, surrounded by braces, {}.

**Member Variables**

The member variables declarations are composed of three components, in order:

1. Zero or one modifiers, such as public or private.
2. The field's type.
3. The field's name.

The fields of class **Person** for example, are name, age, height etc. The public keyword identifies these fields as public members, accessible by any object that can access the class.

**Access Modifiers**

* public modifier—the field is accessible from all classes.
* private modifier—the field is accessible only within its own class.

In the spirit of encapsulation, it is common to make fields private. This means that they can only be directly accessed from within the class. We still need access to these values which is done indirectly by adding public methods that return the field values.

**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 instantiated when the class is loaded. Instance variables can be accessed from inside any method, constructor or blocks of that particular class.

**Class variables:** Class variables are variables declared with in a class, outside any method, with the static keyword.

A class can have any number of methods to access the value of various kinds of methods.

**Constructors**

Every class has a constructor. If constructor is not defined, Java compiler builds a default constructor for that class.

When object is created, at least one constructor will be invoked. Constructors have the same name as the class. A class can have more than one constructor.

Example of a constructor is given below:

public class Circle{

private float radius;

public Circle(){

//default constructor

this.radius = 5.0;

}

public Circle(float radius){

// This constructor has one parameter.

this.radius = radius;

}

}

**Creating Object:**

A class is blueprints for objects. **new** keyword is used to create objects.

There are three steps when creating objects from a class:

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

**Instantiation:** The 'new' key word is used to create the object.

**Initialization:** The 'new' keyword is followed by a call to a constructor. Constructor initializes the new object.

Syntax:

**ClassName objectName = new ClassName(<arguments>);**

Example:

Circle circle = new Circle(5.2);

**Program:**

The program below creates a stack data structure.

public class Stack {

private int[] stack;

private int stackTop;

private int stackSize;

public Stack(int size){

stackSize = size;

stack = new int[size];

stackTop = 0;

}

public void push(int item){

if(stackTop < stackSize){

stack[stackTop++] = item;

}

else{

//overflow

}

}

public int pop(){

if(stackTop > 0 ){

stack[--stackTop] = item;

}

else{

//underflow

}

}

} //class Stack ends

Here is another example of Car

public class Car {

private int gear;

private int speed;

public Car(int startSpeed, int startGear) {

gear = startGear;

speed = startSpeed;

}

public int getGear() {

return gear;

}

public void setGear(int newValue) {

gear = newValue;

}

public int getSpeed() {

return speed;

}

public void applyBrake(int decrement) {

speed -= decrement;

}

public void speedUp(int increment) {

speed += increment;

}

}

**Types**

All variables must have a type. You can use primitive types such as int, float, boolean, etc. Or you can use reference types, such as strings, arrays, or objects.

**Defining Objects or Instantiating a Class**

Below is a class declaration for Point

public class Point {

public int x = 0;

public int y = 0;

//constructor

public Point(int a, int b) {

x = a;

y = b;

}

}

The new operator instantiates a class by allocating memory for a new object and returning a reference to that memory. The new operator also invokes the object constructor.

This reference to new object is usually assigned to a variable of the appropriate type, like:

Point originOne = new Point(23, 94);

**Source file declaration rules**

Essential rules when declaring classes, *import* statements and *package* statements in a source file.

* There can be only one public class per source file.
* A source file can have multiple non public classes.
* The name of public class should be the name of the source file **.java** extesion. Example,

*public class Employee{},* the source file should be saved as Employee.java.

* If the class is defined inside a package, then the package statement should be the first statement in the source file.
* If import statements are present then they must be written between the package statement and the class declaration. If there are no package statements then the import statement should be the first line in the source file.
* Import and package statements will apply to all the classes present in the source file. It is not possible to declare different import or package statements to different classes in source file.

**Java Package**

Package is a way of categorizing the classes and interfaces. In a non-trivial Java applications, hundreds of classes and interfaces will be written, therefore categorizing these classes is a must to make life much easier.

**Import statements**

In Java if a fully qualified name (includes the package and the class name) is given then the compiler can easily locate the source code or classes. Import statement is a way of giving the proper location for the compiler to find that particular class.

For example, the following line would ask compiler to load all the classes available in directory java\_installation/java/io :

import java.io.\*;

**ACTIVITIES**

**Activity 1**

Create a class Matrix and define operations of addition and multiplication. Define another class with main function and instantiate the Matrix object and perform the operations.

**Activity 2**

Create a Scientific Calculator class with all the typical operations. The main class should instantiate the class and perform the operations.

**REVIEW QUESTIONS:**

Q1: What is the syntax of class declaration?

1. Q2: What is a package?
2. Q3: How many public classes can be there in a single source file?
3. Q4: What is the purpose of constructor function in a class?

Q5: What is the purpose of modifiers?