**Assignment: (Core Java)**

**4 : Classes and Objects**

**Que.1 Defining a Class and Object in Java**

**Ans.1 Class:** A class is a blueprint/template for creating objects. It defines properties and behaviors.

**Object:** An object is an instance of a class. It is created using the **new** keyword. Objects represent real-world entities.

// Define a class

class Student {

String name;

int age;

// method

void display() {

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

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

}

}

// Main class

public class classObject {

public static void main(String[] args) {

// Create an object of Student class

Student s1 = new Student();

// Assign values to object

s1.name = "Yami";

s1.age = 20;

// Call method

s1.display();

}

}

**Que.2 Constructors and Overloading**

**Ans.2**

* A constructor is a special method used to initialize objects.
* It has the same name as the class.
* It has no return type (not even void).
* It is called automatically when an object is created.

Constructor overloading in Java is a feature that allows a class to have multiple constructors with the same name but different parameter lists. This enables the creation of objects in various ways, providing flexibility in object initialization.

Key aspects of constructor overloading:

**Multiple Constructors**: A class can define more than one constructor.

**Same Name, Different Parameters**: All overloaded constructors must have the same name as the class, but their parameter lists must differ. This difference can be in:

* Number of parameters: For example, one constructor takes two parameters, while another takes three.
* Type of parameters: For example, one constructor takes an int and a String, while another takes a double and a String.
* Order of parameters: For example, one constructor takes (int, String), while another takes (String, int).

**Purpose:** It allows for different ways to initialize an object based on the available data or desired initial state. This provides convenience and flexibility to the user of the class.

public class Box {

double width;

double height;

double depth;

// Constructor with all dimensions

public Box(double w, double h, double d) {

width = w;

height = h;

depth = d;

}

// Constructor for a cube (all sides equal)

public Box(double side) {

width = height = depth = side;

}

// Constructor with no parameters (default box)

public Box() {

width = height = depth = 0; // default to 0

}

}

**Que.3 Object Creation, Accessing Members of the Class**

**Ans.3 Object Creation:** To create an object in Java, we use the new keyword along with the constructor.

**Syntax:**

ClassName objectName = new ClassName();

**Example:**

Car c1 = new Car(); // Object c1 created from class Car

**Accessing Members of a Class:** Members of a class = fields (variables) + methods (functions). We use the **dot operator (.)** to access them.

**Syntax:**

objectName.variableName;

objectName.methodName();

**Example: Object Creation & Accessing Members**

class Student {

String name;

int age;

void display() {

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

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

}

}

public class objectAccess {

public static void main(String[] args) {

// Create object of Student

Student s1 = new Student();

// Access and assign values to variables

s1.name = "Yami";

s1.age = 20;

// Access method using object

s1.display();

}

}

**Que.4 this Keyword**

**Ans.4 this Keyword** is a reference variable that refers to the current object. It is mainly used to **differentiate between class variables and local variables** when they have the same name.

Using "this" reference can improve code readability and reduce naming conflicts.

class Student {

String name;

int age;

void setDetails(String name, int age) {

this.name = name; // 'this.name' refers to class field

this.age = age; // 'age' is local variable

}

void display() {

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

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

}

}

public class ThisExample {

public static void main(String[] args) {

Student s1 = new Student();

s1.setDetails("Yami", 20);

s1.display();

}

}