Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Academic Year: 2022-2023

JAVA

EXPERIMENTS

Name:	Prerna Sunil Jadhav
Sap ID:	60004220127
Branch:	Computer Engineering
Semester:	III (DSE)
Course Code:	DJ19CEL306
Course:	Programming Laboratory – I (Java)
Experiments:	6 to 10

Name – Prerna Sunil Jadhav **SAP ID - 60004220127**

Experiment No – 06(A)

AIM: To implement Functions, recursive functions and overloading (CO1)

THEORY:

- Functions/ Methods
  - Java Method is a collection of statements that perform some specific task and return the result to the caller.
  - A Java method can perform some specific task without returning anything.
  - Methods in Java allow us to reuse the code without retyping the code.
  - In Java, every method must be part of some class that is different from languages like C, C++, and Python.
    1. A method is like function i.e. used to expose behavior of an object.
    2. it is a set of codes that perform a particular task.
- Recursion
  - Recursion is the technique of making a function call itself. This technique provides a way to break complicated problems down into simple problems which are easier to solve.
  - Just as loops can run into the problem of infinite looping, recursive functions can run into the problem of infinite recursion. Infinite recursion is when the function never stops calling itself. Every recursive function should have a halting condition, which is the condition where the function stops calling itself.
- Method Overloading
  - If a class has multiple methods having same name but different in parameters, it is known as Method Overloading.
  - If we have to perform only one operation, having same name of the methods increases the readability of the program.

PROGRAM 1: Write A Program to check if 2 strings are Meta strings or not. Meta strings are the strings which can be made equal by exactly one swap in any of the strings. Equal string are not considered here as Meta strings.
Example: str1 = "geeks", str2 = "keegs"
By just swapping 'k' and 'g' in any of string, both will become same.
Example: str1 = "Converse", str2 = "Conserve"
By just swapping 'v' and's' in any of string, both will become same.

CODE (i): WAP to display area of square and rectangle using the concept of overloaded functions

Code1_MethodOverloading.java

```java
package Exp6;

class Shape
{
    void area(float x)
    {
        System.out.println("The area of the square is "+Math.pow(x, b: 2)+" sq units");
    }
    void area(float x, float y)
    {
        System.out.println("The area of the rectangle is "+x*y+" sq units");
    }
    void area(double x)
    {
        double z = 3.14 * x * x;
        System.out.println("The area of the circle is "+z+" sq units");
    }
}

public class Code1_MethodOverloading {
    Run | Debug
    public static void main(String[] args) {
        System.out.println(x: "Prerna Sunil Jadhav - 60004220127\n");

        Shape ob = new Shape();
        ob.area(x: 5);
        ob.area(x: 11,y: 12);
        ob.area(x: 2.5);
    }
}
```

OUTPUT:

```
Prerna Sunil Jadhav - 60004220127

The area of the square is 25.0 sq units
The area of the rectangle is 132.0 sq units
The area of the circle is 19.625 sq units
```

(ii) Write menu driven program to implement recursive functions for following tasks

CODE (ii)(a): To find GCD and LCM

Code2_GCD.java

```java
package Exp6;

public class Code2_GCD {
    Run | Debug
    public static void main(String[] args) {
        System.out.println(x: "Prerna Sunil Jadhav - 60004220127\n");
        int n1 = 4, n2 = 22;
        int gcd = gcd(n1, n2);
        int lcm = lcm(n1, n2);
        System.out.printf(format: "G.C.D of %d and %d is %d.\n", n1, n2, gcd);
        System.out.printf(format: "L.C.M of %d and %d is %d.", n1, n2, lcm);
    }
    static int lcm(int a, int b)
    {
        return (a / gcd(a, b)) * b;
    }

    public static int gcd(int n1, int n2)
    {
        if (n2 != 0)
            return gcd(n2, n1 % n2);
        else
            return n1;
    }
}
```

OUTPUT:

```
Prerna Sunil Jadhav - 60004220127

G.C.D of 4 and 22 is 2.
L.C.M of 4 and 22 is 44.
```

CODE (ii)(b): To find X^y

Code3_Power.java

```java
package Exp6;

class Code3_Power {
    Run | Debug
    public static void main(String[] args) {
        System.out.println(x: "Prerna Sunil Jadhav - 60004220127\n");
        int base = 3, powerRaised = 4;
        int result = power(base, powerRaised);
        System.out.println(base + "^" + powerRaised + "=" + result);
    }

    public static int power(int base, int powerRaised) {
        if (powerRaised != 0) {
            // recursive call to power()
            return (base * power(base, powerRaised - 1));
        } else {
            return 1;
        }
    }
}
```

OUTPUT:

```
Prerna Sunil Jadhav - 60004220127

3^4=81
```

CODE (ii)(c): To print n Fibonacci numbers

Code4_Fibo.java

Exp6 > Code4_Fibo.java > Code4_Fibo > main(String[])

```java
package Exp6;

public class Code4_Fibo {
    public static int fibonacciRecursion(int n) {
        if (n == 0) {
            return 0;
        }
        if (n == 1 || n == 2) {
            return 1;
        }
        return fibonacciRecursion(n - 2) + fibonacciRecursion(n - 1);
    }

    Run | Debug
    public static void main(String args[]) {
        System.out.println(x: "Prerna Sunil Jadhav - 60004220127\n");

        int maxNumber = 16;
        System.out.print("Fibonacci Series of " + maxNumber + " numbers: ");
        for (int i = 0; i < maxNumber; i++) {
            System.out.print(fibonacciRecursion(i) + " ");
        }
    }
}
```

OUTPUT:

```
Prerna Sunil Jadhav - 60004220127

Fibonacci Series of 16 numbers: 0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610
```

CODE (ii)(d): To find reverse of number

Code5_Reverse.java

Exp6 > Code5_Reverse.java > Code5_Reverse > main(String[])

```java
package Exp6;

class Code5_Reverse {
    public static void Reverse(int num)
    {
        if (num < 10) {
            System.out.println(num);
            return;
        }
        else {
            System.out.print(num % 10);
            Reverse(num / 10);
        }
    }

    Run | Debug
    public static void main(String args[])
    {
        System.out.println(x: "Prerna Sunil Jadhav - 60004220127\n");
        int num = 870341009;
        System.out.print(s: "Reversed Number: ");
        Reverse(num);
    }
}
```

OUTPUT:

```
Prerna Sunil Jadhav - 60004220127

Reversed Number: 900143078
```

CODE (ii)(e): To 1+2+3+4+…….+ (n-1)+n

Code6_AddNumbers.java

```java
package Exp6;

public class Code6_AddNumbers {

    Run | Debug
    public static void main(String[] args) {
        System.out.println(x: "Prerna Sunil Jadhav - 60004220127\n");

        int number = 250;
        int sum = addNumbers(number);
        System.out.println("Sum of first " + number + " numbers = " + sum);
    }

    public static int addNumbers(int num) {
        if (num != 0)
            return num + addNumbers(num - 1);
        else
            return num;
    }
}
```

OUTPUT:

```
Prerna Sunil Jadhav - 60004220127

Sum of first 250 numbers = 31375
```

CODE (ii)(f): Calculate sum of digits of a number

Code7_SumOfDigits.java

```java
package Exp6;

public class Code7_SumOfDigits {
    private static int sumOfDigits(int num) {
        if (num == 0) {
            return 0;
        }
        return num % 10 + sumOfDigits(num / 10);
    }

    Run | Debug
    public static void main(String[] args) {
        System.out.println(x: "Prerna Sunil Jadhav - 60004220127\n");

        int result = sumOfDigits(num: 1234);
        System.out.println(result);
    }
}
```

OUTPUT:

```
Prerna Sunil Jadhav - 60004220127

10
```

CONCLUSION:

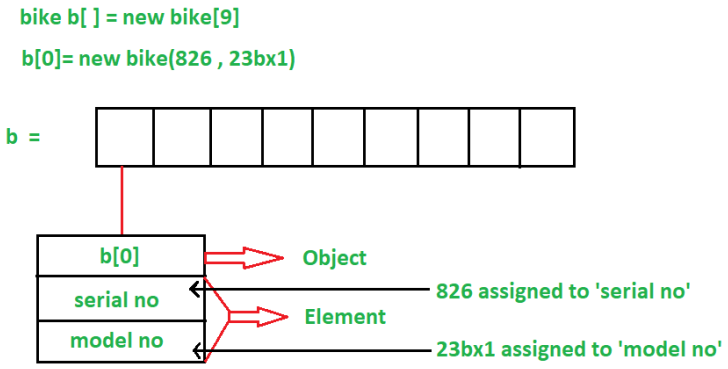
- Method overloading refers to the creation of more than one methods with the same name in the same class. Method overloading is a way that demonstrates polymorphism(or more accurately static polymorphism) in Java.
- Recursion is the process in which a function calls itself and the method that calls itself is known as a recursive function. This means that the method call statement is present in the body of the method itself.

Name – Prerna Sunil Jadhav **SAP ID - 60004220127**

Experiment No – 06(B)

AIM: To implement Array of Objects (CO2)

THEORY:



- The array of Objects the name itself suggests that it stores an array of objects.
- Unlike the traditional array stores values like String, integer, Boolean, etc an Array of Objects stores objects that mean objects are stored as elements of an array.
- Note that when we say Array of Objects it is not the object itself that is stored in the array but the reference of the object.

CODE (i): WOOP to arrange the names of students in descending order of their total marks, input data consists of students details such as names, ID.no, marks of maths, physics, chemistry. (Use array of objects)

Code1_Student.java

Exp6B > Code1_Student.java > Code1_Student > display(Student[])

```java
package Exp6B;
import java.util.Scanner;
public class Code1_Student {
    Run | Debug
    public static void main(String[] args) {
        System.out.println(x: "Prerna Sunil Jadhav - 60004220127\n");
        Scanner sc = new Scanner(System.in);
        System.out.print(s: "\nEnter the number of Students: ");
        int n = sc.nextInt();
        Student[] studArray = new Student[n];
        for (int i = 0; i < n; i++) {
            System.out.println("Details of Student " + (i + 1));
            System.out.print(s: "Enter ID: ");
            int id = sc.nextInt();
            System.out.print(s: "Enter name: ");
            String name = sc.next();
            System.out.print(s: "Enter Maths Marks: ");
            int maths = sc.nextInt();
            System.out.print(s: "Enter Physics Marks: ");
            int phy = sc.nextInt();
            System.out.print(s: "Enter Chemistry Marks: ");
            int chem = sc.nextInt();
            studArray[i] = new Student(id, maths, phy, chem, name);
        }
        System.out.println(x: "Marks in descending order:");
        display(studArray);
        sc.close();
    }
```

Code1_Student.java

Exp6B > Code1_Student.java > Student > Student(int, int, int, int, String)

```java
    public static void display(Student[] arr2) {
        int arr[] = new int[arr2.length];
        for(int i = 0; i<arr.length; i++){
            arr[i] = arr2[i].total;
        }
        for(int i = 1;i < arr2.length; i++) {
            int j = i;
            while(j > 0 && arr[j] < arr[j-1]) {
                int temp = arr[j];
                arr[j] = arr[j-1];
                arr[j-1] = temp;
                j--;
            }
        }
        int[] b = new int[arr2.length];
        int j = arr2.length;
        for (int i = 0; i < arr2.length; i++) {
            b[j - 1] = arr[i];
            j = j - 1;
        }

        for (int i: b) {
            for (Student s: arr2){
                if (s.total == i){
                    System.out.println(s.name+" "+i);
                }
            }
        }
    }
}

class Student {
    int total, id, maths, physics, chemistry;
    String name;

    Student(int id, int maths, int phy, int chem, String name) {
        this.id = id;
        this.maths = maths;
        this.physics = phy;
        this.chemistry = chem;
        this.name = name;
        this.total = this.id + this.maths + this.physics + this.chemistry;
    }
}
```

OUTPUT:

```
Prerna Sunil Jadhav - 60004220127


Enter the number of Students: 3
Details of Student 1
Enter ID: 1
Enter name: Prerna
Enter Maths Marks: 78
Enter Physics Marks: 98
Enter Chemistry Marks: 88
Details of Student 2
Enter ID: 2
Enter name: Diksha
Enter Maths Marks: 92
Enter Physics Marks: 70
Enter Chemistry Marks: 89
Details of Student 3
Enter ID: 3
Enter name: Krishna
Enter Maths Marks: 69
Enter Physics Marks: 89
Enter Chemistry Marks: 88
Marks in descending order:
Prerna 265
Diksha 253
Krishna 249
```

CONCLUSION:

- An object represents a single record in memory, and thus for multiple records, an array of objects must be created.
- It must be noted that the arrays can hold only references to the objects, and not the objects themselves.

Name – Prerna Sunil Jadhav **SAP ID - 60004220127**

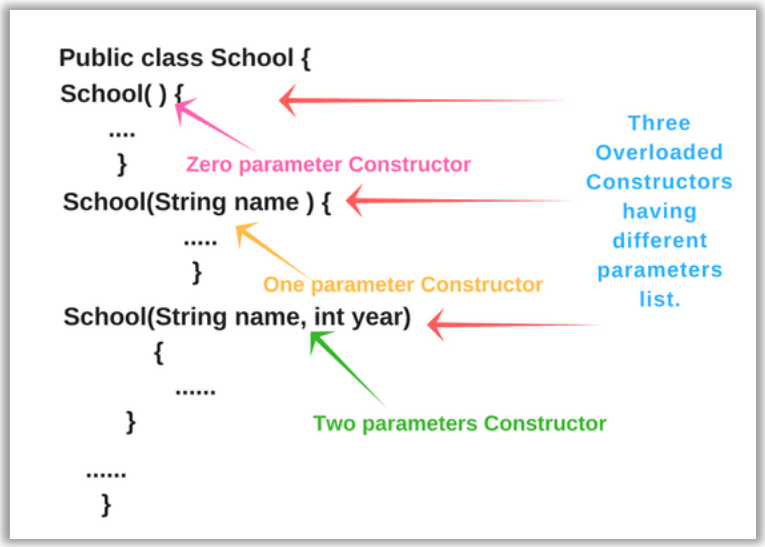
Experiment No – 07

AIM: To implement Constructors and overloading (CO2)

THEORY:

Constructor:

- In Java, a constructor is a block of codes similar to the method. It is called when an instance of the class is created. At the time of calling constructor, memory for the object is allocated in the memory.
- It is a special type of method which is used to initialize the object.
- Every time an object is created using the new() keyword, at least one constructor is called.
- It calls a default constructor if there is no constructor available in the class. In such case, Java compiler provides a default constructor by default.



Constructor Overloading:

- In Java, we can overload constructors like methods.
- The constructor overloading can be defined as the concept of having more than one constructor with different parameters so that every constructor can perform a different task.

CODE (i): WAP find area of square and rectangle using overloaded constructor

Code1_ConstructorOverloading.java

```
Exp7 > Code1_ConstructorOverloading.java > ...
1   package Exp7;
2
3   public class Code1_ConstructorOverloading {
        Run | Debug
4       public static void main(String[] args) {
5           System.out.println(x: "Prerna Sunil Jadhav - 60004220127\n");
6
7           Shape rect = new Shape(length: 12, breadth: 9);
8           System.out.println("Area of Rectangle: "+rect.area+" sq.units.");
```

```
10          Shape square = new Shape(side: 122);
11          System.out.println("Area of Square: "+square.area+" sq.units.");
12      }
13  }
14
15  class Shape{
16      int area;
17
18      Shape(int length, int breadth){
19          this.area = length*breadth;
20      }
21
22      Shape(int side){
23          this.area = side*side;
24      }
25  }
```

OUTPUT:

```
Prerna Sunil Jadhav - 60004220127

Area of Rectangle: 108 sq.units.
Area of Square: 14884 sq.units.
```

CODE (ii): Create Rectangle and Cube class that encapsulates the properties of a rectangle and cube i.e. Rectangle has default and parameterized constructor and area() method. Cube has default and parameterized constructor and volume() method. They share no ancestor other than Object.

Implement a class Size with size() method. This method accepts a single reference argument z. If z refers to a Rectangle then size(z) returns its area and if z is a reference of Cube, then z returns its volume. If z refers to an object of any other class, then size(z) returns -1. Use main method in Size class to call size(z) method.

Code2_Shape.java

Exp7 > Code2_Shape.java > ...

```java
package Exp7;
import java.util.*;
class Rect {
    private int l, b;
    Rect(int l, int b) {
        this.l = l;
        this.b = b;
    }
    int area() {
        return l * b;
    }
}
class Cube {
    private int side;
    Cube(int side) {
        this.side = side;
    }
    int volume() {
        return side * side * side;
    }
}
class Size {
    public static int size(Object o) {
        if (o instanceof Rect) {
            return ((Rect) o).area();
        } else if (o instanceof Cube) {
            return ((Cube) o).volume();
        } else {
            return -1;
        }
    }
}
public class Code2_Shape {
    Run | Debug
    public static void main(String[] args) {
        System.out.println(x: "Prerna Jadhav - 60004220127");
        Scanner sc = new Scanner(System.in);
        Rect r = new Rect(l: 5, b: 6);
        Cube c = new Cube(side: 4);
        System.out.println("Area of Rectangle : " + Size.size(r));
        System.out.println("Volume of Cube : " + Size.size(c));
        System.out.println("Other objects : " + Size.size(sc));
    }
}
```

OUTPUT:

```
Prerna Jadhav - 60004220127
Area of Rectangle : 30
Volume of Cube : 64
Other objects : -1
```

CONCLUSION: Thus, we implemented programs on Constructor and Destructor.

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Academic Year: 2022-2023

Name – Prerna Sunil Jadhav **SAP ID - 60004220127**

Experiment No – 08

AIM: To implement Abstract classes (CO4)

THEORY:

In this below given program we have implemented concepts like data encapsulation , constructor overloading. Encapsulation in Java is a mechanism of wrapping the data (variables) and code acting on the data (methods) together as a single unit. In encapsulation, the variables of a class will be hidden from other classes and can be accessed only through the methods of their current class. Abstract class called Shape has three subclasses say Triangle, Rectangle, Circle. Method area() in the abstract class and override this area() in these three subclasses to calculate for specific object i.e., Area() of Triangle subclass should calculate area of triangle etc. Same for Rectangle and Circle. An abstract class is like a blueprint/format about the minimum required functions. A method which is declared as abstract and does not have implementation is known as an abstract method.

CODE (i): Write a abstract class program to calculate area of circle, rectangle and triangle

Code1_Abstract.java

Exp8 > Code1_Abstract.java > ...

```java
package Exp8;
import java.lang.Math;
abstract class Shape {
    abstract void area();
    double area;
}
class Triangle extends Shape {
    double b = 50, h = 15;
    void area() {
        area = (b * h) / 2;
        System.out.println("area of Triangle -->" + area);
    }
}
class Rectangle extends Shape {
    double w = 70, h = 20;
    void area() {
        area = w * h;
        System.out.println("area of Rectangle -->" + area);
    }
}
```

```java
class Circle extends Shape {
    double r = 5;
    void area() {
        area = Math.PI * r * r;
        System.out.println("area of Circle -->" + area);
    }
}
class Exp8_Abstract {
    Run | Debug
    public static void main(String[] args) {
        System.out.println(x: "Prerna Jadhav 60004220127");
        Triangle t = new Triangle();
        Rectangle r = new Rectangle();
        Circle c = new Circle();
        t.area();
        r.area();
        c.area();
    }
}
```

OUTPUT:

```
Prerna Jadhav 60004220127
area of Triangle -->375.0
area of Rectangle -->1400.0
area of Circle -->78.53981633974483
```

CONCLUSION: Thus, we implemented Abstract classes.

Name – Prerna Sunil Jadhav **SAP ID - 60004220127**

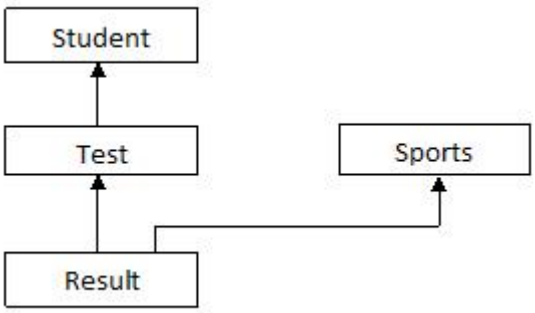
Experiment No – 09

AIM: To implement Inheritance, interfaces and method Overriding

THEORY:

In this program we created, and interface named sports which consists of score function and created 3 classes namely student, text by extending the student class and Result by extending student class and implementing the interface sports. Lastly, we created class multiple and executed all the functions. Inheritance in Java is a mechanism in which one object acquires all the properties and behaviors of a parent object. It is an important part of OOPs (Object Oriented programming system). An interface is a reference type, similar to a class. That can contain only constants, method signatures, default methods, static methods, and nested types.

CODE (i): WAP to implement three classes namely Student, Test and Result. Student class has member as roll no, Test class has members as sem1_marks and sem2_marks and Result class has member as total. Create an interface named sports that has a member score (). Derive Test class from Student and Result class has multiple inheritances from Test and Sports. Total is formula based on sem1_marks, sem2_mark and score.



Code1_OOPs.java

Exp9 > Code1_OOPs.java > ...

```java
package Exp9;
import java.util.Scanner;
interface Sports {
    int score = 100;
    void member_score();
}
class Student {
    int roll_no;
    void read(int n) {
        roll_no = n;
    }
    void display() {
        System.out.println(roll_no);
    }
}
class Test extends Student {
    int sem1_marks, sem2_marks;
    void read1(int n) {
        sem1_marks = n;
    }
    void read2(int n) {
        sem2_marks = n;
    }
    void display() {
        System.out.println(sem1_marks + sem1_marks);
    }
}
class Result extends Test implements Sports {
    public void member_score() {
        int total;
        total = sem1_marks + sem2_marks + score;
        System.out.println("The total score is " + total);
    }
}
public class Code1_OOPs {
    Run | Debug
    public static void main(String args[]) {
        System.out.println(x: "Prerna Jadhav - 60004220127");
        Scanner s = new Scanner(System.in);
        Result r = new Result();
        System.out.println(x: "Enter roll no.");
        int roll = s.nextInt();
        System.out.println(x: "Enter sem1.");
        int sem1 = s.nextInt();
        System.out.println(x: "Enter sem2.");
        int sem2 = s.nextInt();
        r.read(roll);
        r.read1(sem1);
        r.read2(sem2);
        r.member_score();
        s.close();
    }
```

Output:

```
Prerna Jadhav - 60004220127
Enter roll no.
23
Enter sem1.
122
Enter sem2.
234
The total score is 456
```

Conclusion:
Thus we implemented Inheritance, interfaces and method Overriding

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Academic Year: 2022-2023

Name – Prerna Sunil Jadhav **SAP ID - 60004220127**

Experiment No – 10

AIM: To implement Package (CO2)

THEORY:

Package in Java is a mechanism to encapsulate a group of classes, sub packages and interfaces. To create package in Java: - First create a directory within the name of the package. - Create a java file in the newly created directory. - In this java file you must specify the package name with the help of package keyword. - Save this file with same name of public class Note: only one class in a program can declare as public.

CODE (i): WAP to create a user defined package & import the package in another program.

Greetings.java

Exp10 > UserDefinedPackage > Greetings.java > ...

```java
package Exp10.UserDefinedPackage;

public class Greetings {
    public void greet(){
        System.out.println(x: "Prerna Sunil Jadhav - 60004220127\n");
        System.out.println(x: "You are welcome to the JAVA world");
    }
}
```

English.java

Exp10 > English.java > ...

```java
package Exp10;
import Exp10.UserDefinedPackage.Greetings;

public class English {
    Run | Debug
    public static void main(String[] args) {
        Greetings g = new Greetings();
        g.greet();
    }
}
```

Output:

```
Prerna Sunil Jadhav - 60004220127

You are welcome to the JAVA world
```

Conclusion:
Thus we implemented Package