ASSIGNMENT 2

Name: Aaditya Khot

PRN: 21610051

Batch: S5

Branch: IT

Course: Java Programming Lab

Q.1 Explain difference between method overloading and method overriding.

→

- **Method Overloading:** In Java it is possible to define two or more methods within the same class that share the same name, if their parameter declarations are different. When this is the case, the methods are said to be overloaded, and the process is referred to as method overloading. Method overloading is one of the ways that Java supports polymorphism. When an overloaded method is invoked, Java uses the type and/or number of arguments as its guide to determine which version of the overloaded method to actually call. Thus, overloaded methods must differ in the type and/or number of their parameters. While overloaded methods may have different return types, the return type alone is insufficient to distinguish two versions of a method. When Java encounters a call to an overloaded method, it simply executes the version of the method whose parameters match the arguments used in the call.

- **Method Overriding:** In a class hierarchy, when a method in a subclass has the same name and type signature as a method in its superclass, then the method in the subclass is said to override the method in the superclass. When an overridden method is called from within a subclass, it will always refer to the version of that method defined by the subclass. Method overriding occurs only when the names and the type signatures of the two methods are identical. If they are not, then the two methods are simply overloaded. Overridden methods allow Java to support run-time polymorphism. Polymorphism is essential to object-oriented programming for one reason: it allows a general class to specify methods that will be common to all its derivatives, while allowing subclasses to define the specific implementation of some or all of those methods. Overridden methods are another way that Java implements the "one interface, multiple methods" aspect of polymorphism

Q.2 Implement all string functions in Java.

→

```
import java.util.*;

public class StringFunctions {
   public static void main(String[] args) {
      String str1 = "Hello World!";
      String str2 = "hello world";
      String str3 = "Java is awesome";

      // charAt(int index)
      char ch = str1.charAt(4);
      System.out.println("Character at index 4 of str1: " + ch);

      // concat(String str)
      String str4 = str1.concat(" How are you?");
```

```
System.out.println("Concatenated string: " + str4);

// contains(CharSequence sequence)
boolean result = str3.contains("is");
System.out.println("Does str3 contain 'is'? " + result);

// endsWith(String suffix)
result = str1.endsWith("!");
System.out.println("Does str1 end with '!'? " + result);

// equals (Object obj)
result = str1.equals(str2);
System.out.println("Are str1 and str2 equal? " + result);

// equalsIgnoreCase (String str)
result = str1.equalsIgnoreCase(str2);
```

```
System.out.println("Are str1 and str2 equal (ignoring case)? " + result);

// indexOf (int ch)
int index = str1.indexOf('o');
System.out.println("Index of 'o' in str1: " + index);

// isEmpty ()
result = str1.isEmpty();
System.out.println("Is str1 empty? " + result);

// length ()
int length = str3.length();
System.out.println("Length of str3: " + length);

// replace (char oldChar, char newChar)
String str5 = str1.replace('o', 'e');
```

```
        System.out.println("Replaced 'o' with 'e");
    }
}
```

```
D:\2nd Year\SEM 4\Java Porgramming>javac StringFunctions.java

D:\2nd Year\SEM 4\Java Porgramming>java StringFunctions
Character at index 4 of str1: o
Concatenated string: Hello World! How are you?
Does str3 contain 'is'? true
Does str1 end with '!'? true
Are str1 and str2 equal? false
Are str1 and str2 equal (ignoring case)? false
Index of 'o' in str1: 4
Is str1 empty? false
Length of str3: 15
Replaced 'o' with 'e

D:\2nd Year\SEM 4\Java Porgramming>
```

Q.3 Implement all String Buffer functions in Java.

→

```
import java.util.*;
public class StringBuffers {
    public static void main(String[] args){
```

```
StringBuffer sb = new StringBuffer("Hello World!");

System.out.println("Buffer = " + sb);
System.out.println("Length = " + sb.length());
System.out.println("Capacity = " + sb.capacity());

char ch = sb.charAt(1); // returns 'e'
sb.setCharAt(1, 'E'); // changes the StringBuffer to "HEllo, World!"
System.out.println(sb);

sb.append(", World!"); // changes the StringBuffer to "Hello, World!, World"
System.out.println(sb);

sb.insert(7, "there, "); // changes the StringBuffer to "Hello, there, World!"
System.out.println(sb);
```

```
        sb.delete(7, 13); // changes the StringBuffer to "Hello,!"
        System.out.println(sb);

        sb.reverse(); // changes the StringBuffer to "!dlroW ,olleH"
        System.out.println(sb);

        String str = sb.toString(); // returns "Hello, World!"
        System.out.println(str);
    }
}
```

```
D:\2nd Year\SEM 4\Java Porgramming>java StringBuffers
Buffer = Hello World!
Length = 12
Capacity = 28
HEllo World!
HEllo World!, World!
HEllo Wthere, orld!, World!
HEllo W orld!, World!
!dlroW ,!dlro W ollEH
!dlroW ,!dlro W ollEH

D:\2nd Year\SEM 4\Java Porgramming>
```

Q.4 Explain with example declaration of string using string literal and new keyword.

→

In Java, a string is a sequence of characters, which can be declared in two different ways: using string literals or using the new keyword.

1. **Declaration using string literals**:

A string literal is a sequence of characters enclosed within double quotes (''). Here's an example of declaring a string using string literals in Java:

String myString = "Hello, world!";

2. **Declaration using new keyword:**

In Java, strings are objects and can be created using the new keyword. Here is an example of declaring a string using new keyword in Java:

String myString = new String ("Hello, world!");

In the second example, a new instance of the String class is created using the new keyword and the constructor that takes a string as its argument. This creates a new string object on the heap, separate from the string pool, with its own memory allocation. While it is less common to declare strings using new in Java, it can be useful in certain situations where a new object is needed.

Q.5 Create a class named 'Shape' with a method to print "This is shape." .Then create two other classes named 'Rectangle', 'Circle' inheriting the Shape class, both having a method to print "This is rectangular " and "This is circular" respectively. Create a subclass 'Square' of 'Rectangle' having a method to print "Square is a rectangle". Now call the method of 'Shape' and 'Rectangle' class by the object of 'Square' class.

→

class Shape{

```
    public void printShape(){
        System.out.println("This is Shape.");
    }
}

class Rectangle extends Shape{
    public void printShape(){
        super.printShape();
        System.out.println("This is
Rectangle.");
    }
}

class Circle extends Shape{
    public void printShape(){
        super.printShape();
        System.out.println("This is Circle.");
    }
}
```

```
class Square extends Rectangle{
   public void printShape(){
      super.printShape();
      System.out.println("Square is
Rectangle.");
   }
}

public class Main{
   public static void main(String[] args){
      Square sq = new Square();
      sq.printShape();
   }
}
```

```
D:\2nd Year\SEM 4\Java Porgramming>javac Main.java

D:\2nd Year\SEM 4\Java Porgramming>java Main
This is Shape.
This is Rectangle.
Square is Rectangle.

D:\2nd Year\SEM 4\Java Porgramming>
```

Q.6

```java
class Person {
    public void talk() {
        System.out.println("I can talk");
    }

    public void walk() {
        System.out.println("I can walk");
    }

    public void eat() {
        System.out.println("I can eat");
    }

    public void sleep() {
        System.out.println("I can sleep");
    }
}

class MathsTeacher extends Person {
    public void teachMaths() {
        super.talk();
        super.walk();
        super.eat();
        super.sleep();
        System.out.println("I can teach Maths\n");
    }
}

class Footballer extends Person {
    public void playFootball() {
        super.talk();
        super.walk();
        super.eat();
        super.sleep();
        System.out.println("I can play football\n");
    }
```

```
}

class Businessman extends Person {
    public void runBusiness() {
        super.talk();
        super.walk();
        super.eat();
        super.sleep();
        System.out.println("I can run a business\n");
    }
}

public class Multiple {
    public static void main(String[] args){
        MathsTeacher m = new MathsTeacher();
        System.out.println("Maths Teacher can do following things :
");
        m.teachMaths();

        Footballer f = new Footballer();
        System.out.println("Footballer can do following things : ");
        f.playFootball();

        Businessman b = new Businessman();
        System.out.println("Businessman can do following things :
");
        b.runBusiness();
    }
}
```

```
D:\2nd Year\SEM 4\Java Porgramming>javac Multiple.java

D:\2nd Year\SEM 4\Java Porgramming>java Multiple
Maths Teacher can do following things :
I can talk
I can walk
I can eat
I can sleep
I can teach Maths

Footballer can do following things :
I can talk
I can walk
I can eat
I can sleep
I can play football

Businessman can do following things :
I can talk
I can walk
I can eat
I can sleep
I can run a business
```

Q.7

→

```
public class InheritanceEx2Main {

    static class Department {
        protected String deptName;

        public Department(String deptName)
{
```

```
        this.deptName = deptName;
    }
  }

  static class Employee extends
Department {
    protected String empName;
    protected int empID;
    protected double salary;

    public Employee(String deptName,
String empName, int empID, double
salary) {
        super(deptName);
        this.empName = empName;
        this.empID = empID;
        this.salary = salary;
    }

    public double getSalary() {
```

```
        return salary;
    }
}

static class Allowance extends Employee {
    protected double allowance;

    public Allowance(String deptName, String empName, int empID, double salary, double allowance) {
        super(deptName, empName, empID, salary);
        this.allowance = allowance;
    }

    public double getAllowance() {
        return allowance;
    }
```

```
    public double getTotalSalary() {
        return salary + allowance;
    }
  }

  public static void main(String[] args) {
    Allowance emp1 = new
Allowance("IT", "Parth", 101, 50000,
2000);
    Allowance emp2 = new
Allowance("CSE", "Shreeya", 102, 60000,
2500);

    System.out.println("Employee 1:");
    System.out.println("Department: " +
emp1.deptName);
    System.out.println("Name: " +
emp1.empName);
    System.out.println("ID: " +
emp1.empID);
```

```
        System.out.println("Salary: " +
emp1.getSalary());
        System.out.println("Allowance: " +
emp1.getAllowance());
        System.out.println("Total Salary: " +
emp1.getTotalSalary());


        System.out.println("Employee 2:");
        System.out.println("Department: " +
emp2.deptName);
        System.out.println("Name: " +
emp2.empName);
        System.out.println("ID: " +
emp2.empID);
        System.out.println("Salary: " +
emp2.getSalary());
        System.out.println("Allowance: " +
emp2.getAllowance());
        System.out.println("Total Salary: " +
emp2.getTotalSalary());
    }
```

}

```
PROBLEMS 1   OUTPUT   DEBUG CONSOLE   TERMINAL                                Code

PS C:\Users\asus\Desktop\Java ASG> cd "c:\Users\asus\Desktop\Java ASG\" ; if ($?) { javac InheritanceEx2Main.java }
 ; if ($?) { java InheritanceEx2Main }
Employee 1:
Department: IT
Name: Parth
ID: 101
Salary: 50000.0
Allowance: 2000.0
Total Salary: 52000.0
Employee 2:
Department: CSE
Name: Shreeya
ID: 102
Salary: 60000.0
Allowance: 2500.0
Total Salary: 62500.0
PS C:\Users\asus\Desktop\Java ASG>
```

Q8 Write a Java Program to demonstrate StringBuilder class methods.

→

```
public class StringBuilderDemo {
    public static void main(String[] args) {
        StringBuilder sb = new StringBuilder("Hello");
        sb.append(" World");
        sb.reverse();
        System.out.println(sb);
        sb.insert(6, " there,");
        sb.delete(5, 7);
        System.out.println(sb);
```

```
        sb.replace(0, 5, "Hi");
        System.out.println(sb);
        String str = sb.toString();
        System.out.println(str);
        StringBuilder sb2 = new
StringBuilder(10);
        sb2.append("Java");
        System.out.println("Length of sb2: "
+ sb2.length());
        System.out.println("Capacity of sb2:
" + sb2.capacity());
    }
}
```

```
PROBLEMS 2   OUTPUT   DEBUG CONSOLE   TERMINAL                    Code

PS C:\Users\asus\Desktop\Java ASG> cd "c:\Users\asus\Desktop\Java ASG\" ; if ($?) { javac StringBuilderDemo.java }
; if ($?) { java StringBuilderDemo }
dlroW olleH
dlroWthere,olleH
Hithere,olleH
Hithere,olleH
Length of sb2: 4
Capacity of sb2: 10
PS C:\Users\asus\Desktop\Java ASG>
```

Q9 Write a Java Program to demonstrate Method overriding.(create class

Result with method result(). Override method result() in UGResult and

PGResult class)

→

```
class Result {
   public void result() {
      System.out.println("This is the result of the exam.");
   }
}

class UGResult extends Result {
   @Override
   public void result() {
      System.out.println("This is the result of the UG exam.");
   }
}

class PGResult extends Result {
```

```
    @Override
    public void result() {
        System.out.println("This is the result
of the PG exam.");
    }
}

public class Overriding {
    public static void main(String[] args) {
        Result r = new Result();
        UGResult ugr = new UGResult();
        PGResult pgr = new PGResult();

        r.result();
        ugr.result();
        pgr.result();
    }
}
```

```
PROBLEMS 2   OUTPUT   DEBUG CONSOLE   TERMINAL                                   Code

PS C:\Users\asus\Desktop\Java ASG> cd "c:\Users\asus\Desktop\Java ASG\" ; if ($?) { javac Overriding.java } ; if ($
?) { java Overriding }
This is the result of the exam.
This is the result of the UG exam.
This is the result of the PG exam.
PS C:\Users\asus\Desktop\Java ASG>
```

Q 10 Write a java program to create a class called STUDENT with data members PRN, Name and age. Using inheritance, create a classes called UGSTUDENT and PGSTUDENT having fields as semester, fees and stipend. Enter the data for at least 5 students. Find the semester wise average age for all UG and PG students separately.

→

```
import java.util.Scanner;


class STUDENT {
    int PRN;
    String Name;
    int age;
```

```
    void getData() {
        Scanner sc = new
Scanner(System.in);
        System.out.println("Enter PRN,
Name and Age: ");
        PRN = sc.nextInt();
        Name = sc.next();
        age = sc.nextInt();
    }
}

class UGSTUDENT extends STUDENT {
    int semester;
    double fees;

    void getUGData() {
        getData();
        Scanner sc = new
Scanner(System.in);
```

```
        System.out.println("Enter Semester
and Fees: ");
        semester = sc.nextInt();
        fees = sc.nextDouble();
    }
}

class PGSTUDENT extends STUDENT {
    int semester;
    double fees;
    double stipend;

    void getPGData() {
        getData();
        Scanner sc = new
Scanner(System.in);
        System.out.println("Enter Semester,
Fees, and Stipend: ");
        semester = sc.nextInt();
        fees = sc.nextDouble();
```

```
        stipend = sc.nextDouble();
    }
}

public class example {
    public static void main(String[] args) {
        Scanner sc = new
Scanner(System.in);
        int n, m;
        System.out.println("Enter the
number of UG students: ");
        n = sc.nextInt();
        UGSTUDENT[] ug = new
UGSTUDENT[n];
        double[] sumAgeUG = new
double[8];
        double[] countUG = new double[8];
        for (int i = 0; i < n; i++) {
            ug[i] = new UGSTUDENT();
            ug[i].getUGData();
```

```
        sumAgeUG[ug[i].semester] +=
ug[i].age;
        countUG[ug[i].semester]++;
    }
    System.out.println("Enter the
number of PG students: ");
    m = sc.nextInt();
    PGSTUDENT[] pg = new
PGSTUDENT[m];
    double[] sumAgePG = new
double[8];
    double[] countPG = new double[8];
    for (int i = 0; i < m; i++) {
        pg[i] = new PGSTUDENT();
        pg[i].getPGData();
        sumAgePG[pg[i].semester] +=
pg[i].age;
        countPG[pg[i].semester]++;
    }
    System.out.println("Semester-wise
average age of UG students:");
```

```
        for (int i = 1; i <= 7; i++) {
            if (countUG[i] != 0) {
                System.out.println("Semester " +
i + ": " + sumAgeUG[i] / countUG[i]);
            }
        }
        System.out.println("Semester-wise
average age of PG students:");
        for (int i = 1; i <= 7; i++) {
            if (countPG[i] != 0) {
                System.out.println("Semester " +
i + ": " + sumAgePG[i] / countPG[i]);
            }
        }


    }
}
```

```
PROBLEMS 6   OUTPUT   DEBUG CONSOLE   TERMINAL
Enter PRN, Name and Age:
1 Vivek 19
Enter Semester, Fees, and Stipend:
3 100000 20000
Enter PRN, Name and Age:
2 Jyoti 20
Enter Semester, Fees, and Stipend:
4
100000
20000
Enter PRN, Name and Age:
3 Kaustubh 19
Enter Semester, Fees, and Stipend:
4 100000 20000
Semester-wise average age of UG students:
Semester 4: 19.0
Semester-wise average age of PG students:
Semester 3: 19.0
Semester 4: 19.5
PS C:\Users\asus\Desktop\Java ASG>
```

Q11 Implement hybrid inheritance using all access specifiers (public, private, protected).

→

```
class Person {
    protected String name;
    protected int age;
    private String address;

    public Person(String name, int age, String address) {
        this.name = name;
        this.age = age;
        this.address = address;
    }
```

```
    protected void walk() {
        System.out.println(name + " is
walking");
    }

    private void speak() {
        System.out.println(name + " is
speaking");
    }
}

class Student extends Person {
    protected int grade;

    public Student(String name, int age,
String address, int grade) {
        super(name, age, address);
        this.grade = grade;
    }
```

```
    protected void study() {
        System.out.println(name + " is
studying");
    }
}

class Teacher extends Person {
    private String subject;

    public Teacher(String name, int age,
String address, String subject) {
        super(name, age, address);
        this.subject = subject;
    }

    public void teach() {
        System.out.println(name + " is
teaching " + subject);
    }
```

```
}

public class AccessSpecifier {
    public static void main(String[] args) {
        Student student = new
Student("Himanshu", 19, "Hinganghat ",
69);
        student.walk();
        student.study();
        Teacher teacher = new
Teacher("Rupesh", 19, "Pune",
"Backend");
        teacher.walk();
        teacher.teach();
    }
}
```

```
PROBLEMS 8   OUTPUT   DEBUG CONSOLE   TERMINAL                    Code

PS C:\Users\asus\Desktop\Java ASG> cd "c:\Users\asus\Desktop\Java ASG\" ; if ($?) { javac AccessSpecifier.java } ;
if ($?) { java AccessSpecifier }
Himanshu is walking
Himanshu is studying
Rupesh is walking
Rupesh is teaching Backend
PS C:\Users\asus\Desktop\Java ASG>
```

Q12 Write a program to implement a class Teacher contains two fields Name and Qualification. Extend the class to Department, it contains Dept. No and Dept. Name. An Interface named as College it contains one field Name of the College. Using the above classes and Interface get the appropriate information and display it.

→

```
interface College {
   String COLLEGE_NAME = "XYZ College"; // field of the interface

   String getCollegeName(); // method to get the college name
}

class Teacher {
   String name;
   String qualification;
```

```
    public Teacher(String name, String
qualification) {
        this.name = name;
        this.qualification = qualification;
    }

    public String getName() {
        return name;
    }

    public String getQualification() {
        return qualification;
    }
}

class Department extends Teacher {
    int deptNo;
    String deptName;
```

```
    public Department(String name, String
qualification, int deptNo, String
deptName) {
        super(name, qualification);
        this.deptNo = deptNo;
        this.deptName = deptName;
    }

    public int getDeptNo() {
        return deptNo;
    }

    public String getDeptName() {
        return deptName;
    }
}

class CollegeImpl extends Department
implements College {
```

```
    public CollegeImpl(String name, String
qualification, int deptNo, String
deptName) {
        super(name, qualification, deptNo,
deptName);
    }

    public String getCollegeName() {
        return COLLEGE_NAME;
    }
}

public class Interfaces {
    public static void main(String[] args) {
        CollegeImpl college = new
CollegeImpl("ABC", "PG", 123, "IT");
        System.out.println("College Name: "
+ college.getCollegeName());
        System.out.println("Teacher Name: "
+ college.getName());
```

```
        System.out.println("Teacher Qualification: " + college.getQualification());
        System.out.println("Department No: " + college.getDeptNo());
        System.out.println("Department Name: " + college.getDeptName());
    }
}
```

```
PROBLEMS 8   OUTPUT   DEBUG CONSOLE   TERMINAL                                   Code

PS C:\Users\asus\Desktop\Java ASG> cd "c:\Users\asus\Desktop\Java ASG\" ; if ($?) { javac Interfaces.java } ; if ($
?) { java Interfaces }
College Name: XYZ College
Teacher Name: ABC
Teacher Qualification: PG
Department No: 123
Department Name: IT
PS C:\Users\asus\Desktop\Java ASG>
```