NAME – AMAN MUKESH AGRAWAL

PRN – 2020BTECS00074

PROGRAMMING LAB – II(JAVA)

ASSIGNMENT - 2

1. Explain difference between method overloading and method overriding.

|  |  |
| --- | --- |
| Method Overloading | Method Overriding |
| * Method overloading is a compile-time polymorphism. | * Method Overriding is a runtime polymorphism. |
| * It increases the readability of the program. | * It is used to grant the specific implementation of the method which is already provided by the parent class or base class. |
| * It occurs within the class. | * It is performed in 2 class in inheritance relationship. |
| * It do not need inheritance | * It need to be in between inheritance relationship. |
| * In this, the return type can or cannot be same but we have to change parameter(s). | * In this, the return type must be same or co-varient. |

1. Implement all string functions in java.

Program:

import java.util.\*;

import java.lang.\*;

public class Strings{

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

        String str1 = "Hello";

        String str2 = new **String**("World");

        System.out.**println**(str1 + " " + str2);

*//StringBuffer*

        StringBuffer str3 = new **StringBuffer**("Walchand College");

        System.out.**println**(str3);

*//StringBuilder*

        StringBuilder str4 = new **StringBuilder**();

        str4.**append**(str3 + " says " + str1 + " " + str2);

        System.out.**println**(str4);

*//String Constructors*

*//1. byte array to string*

        byte[] byt = {119,97,108,99,104,97,110,100};

        String byt\_str = new **String**(byt);

        System.out.**println**(byt\_str);

*//2. char array to string*

        char[] ch = {'W','a','l','c','h','a','n','d'};

        String ch\_str = new **String**(ch);

        System.out.**println**(ch\_str);

*//length*

        System.out.**println**(str4.**length**());

*//indexOf*

        System.out.**println**(str3.**indexOf**(ch\_str));

*//charAt*

        System.out.**println**(str3.**charAt**(4));

*//replace*

        System.out.**println**(str3.**replace**(0, 3, str2));

*//toLowerCase*

        System.out.**println**(str1.**toLowerCase**());

*//toUpperCase*

        System.out.**println**(str1.**toUpperCase**());

*//Compare*

        System.out.**println**(str1.**compareTo**(str2));

*//Concat*

        System.out.**println**(str1.**concat**(str2));

*//equalto*

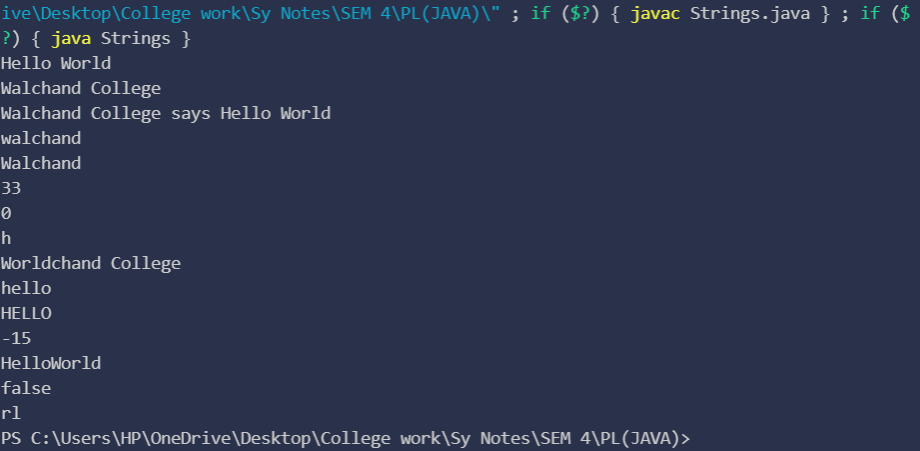
        System.out.**println**(str1.**equals**(str2));

*//substring*

        System.out.**println**(str3.**substring**(2,4));

    }

}

Output: 

1. Implement all stringbuffer functions in java.

Program:

public class StringBuff {

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

*//StringBuffer*

         StringBuffer str1 = new **StringBuffer**("Walchand College");

         System.out.**println**(str1);

*//Append*

         System.out.**println**(str1.**append**(" of Engineering"));

*//insert*

         System.out.**println**(str1.**insert**(0,"The "));

*//Delete*

         System.out.**println**(str1.**delete**(0, 4));

*//Reverse*

         System.out.**println**(str1.**reverse**());

*//Replace*

         System.out.**println**(str1.**replace**(0, 3, "THE"));

*//length*

         System.out.**println**(str1.**length**());

*//subString*

         System.out.**println**(str1.**substring**(4));

*//indexOf*

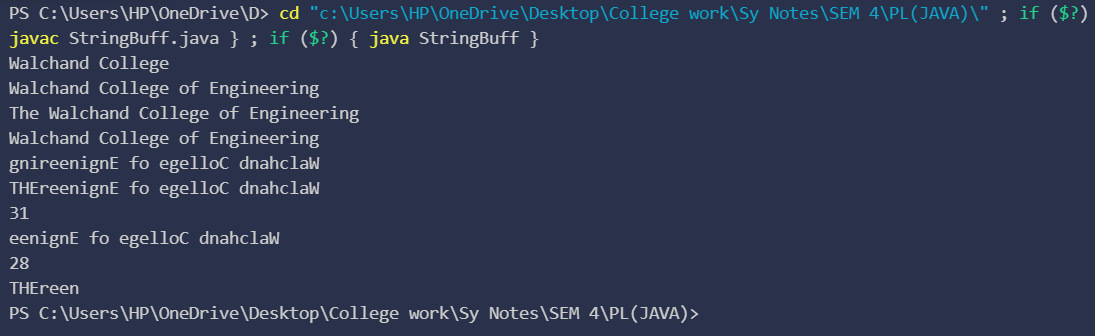
         System.out.**println**(str1.**indexOf**("laW"));

*//subSequence*

         System.out.**println**(str1.**subSequence**(0, 7));

    }

}

Output: 

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

String can be created by string literals,

String s=”Aman”;

And String can also be created with new Key word like,

String s=new String(“Aman”);

1. Create a class named 'Shape' with a method to print "This is This is shape". Then create two other classes named 'Rectangle', 'Circle' inheriting the Shape class, both having a method to print "This is rectangular shape" and "This is circular shape" 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.

Program:

class Shape{

    public void **print\_sh**(){

        System.out.**println**("This is a Shape");

    }

}

class Rectangle extends Shape{

    public void **print\_rect**(){

        System.out.**println**("This is rectangular shape");

    }

}

class Circle extends Shape{

    public void **print\_circle**(){

        System.out.**println**("This is circular shape");

    }

}

class Square extends Rectangle{

    public void **print\_square**(){

        System.out.**println**("Square is a rectangle");

    }

}

public class PrintShape {

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

        Square sq = new **Square**();

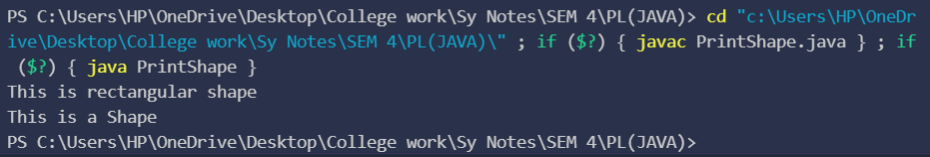
        sq.**print\_rect**();

        sq.**print\_sh**();

    }

}

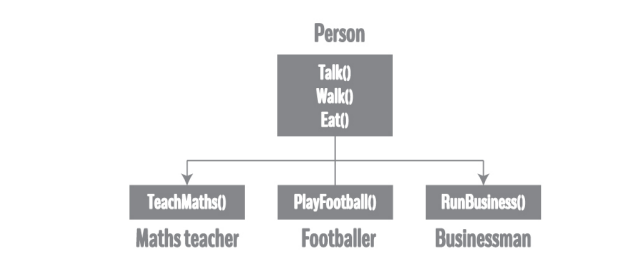
Output:



1. Create game characters using the concept of inheritance. Suppose, in your game, you want three characters - a maths teacher, a footballer and a businessman. Since, all of the characters are persons, they can walk and talk. However, they also have some special skills. A maths teacher can teach maths, a footballer can play football and a businessman can run a business. You can individually create three classes who can walk, talk and perform their special skill as shown in the figure below.



In each of the classes, you would be copying the same code for walk and talk for each character. If you want to add a new feature - eat, you need to implement the same code for each character. This can easily become error prone (when copying) and duplicate codes. It'd be a lot easier if we had a Person class with basic features like talk, walk, eat, sleep, and add special skills to those features as per our characters. This is done using inheritance.



Using inheritance, now you don't implement the same code for walk and talk for each class. You just need to inherit them. So, for Maths teacher (derived class), you inherit all features of a Person (base class) and add a new feature TeachMaths. Likewise, for a footballer, you inherit all the features of a Person and add a new feature PlayFootball and so on.

Program:

class Person{

    public void **talk**()

    {

        System.out.**println**("He can talk.");

    }

    public void **walk**()

    {

        System.out.**println**("He can walk.");

    }

    public void **eat**(){

        System.out.**println**("He can eat.");

    }

}

class MathTeacher extends Person{

    public void **tachMath**()

        {

            System.out.**println**("He can teach maths.");

        }

}

class FootBaller extends Person{

    public void **playFootBall**(){

        System.out.**println**("He can play football.");

    }

}

class Buisnessman extends Person{

    public void **doBuisness**()

    {

        System.out.**println**("He can do buisness.");

    }

}

class Ass3\_6{

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

        MathTeacher teacher=new **MathTeacher**();

        teacher.**eat**();

        teacher.**walk**();

        teacher.**talk**();

        teacher.**tachMath**();

        Buisnessman man=new **Buisnessman**();

        man.**eat**();

        man.**walk**();

        man.**talk**();

        man.**doBuisness**();

        FootBaller footBaller=new **FootBaller**();

        footBaller.**eat**();

        footBaller.**walk**();

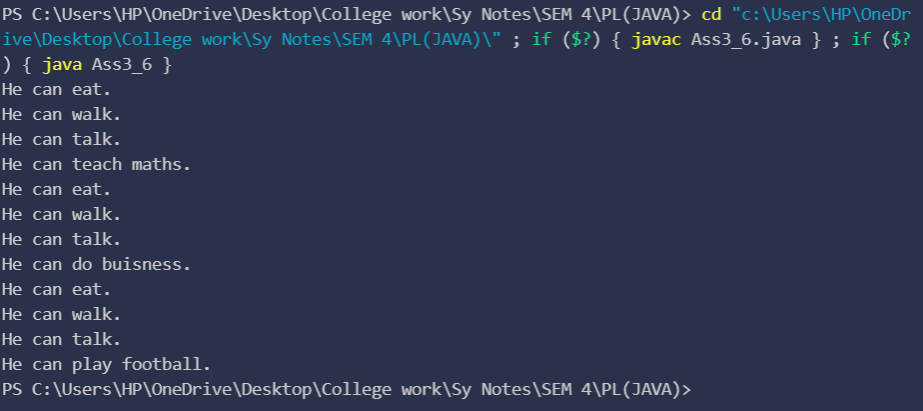
        footBaller.**talk**();

        footBaller.**playFootBall**();

    }

}

Output:



1. WAP to manage the employee allowance from a specific department by creating class structure as follow,

InheritanceEx2

|

|

InheritanceEx2Main.java

|

|- dept | Deparment.java

|

|

|

|- emp | Employee.java extends Department

|

|

|

|- allowance | Allowance.java extends Employee

|

| [Multilevel Inheritance]

Program:

import java.util.Scanner;

class Department {

    protected String departmentName;

    protected int department\_no;

}

class Employee extends Department {

    protected int emp\_id;

    protected String employeeFirstName;

    protected String employeeLastName;

    protected String gender;

    protected byte age;

    protected int experience;

**Employee**()

    {

        Scanner sc=new **Scanner**(System.in);

        System.out.**print**("Enter employee id:-");

        emp\_id=sc.**nextInt**();

        System.out.**print**("Enter employee's first name:-");

        employeeFirstName=sc.**next**();

        System.out.**print**("Enter employee's last name:-");

        employeeLastName=sc.**next**();

        System.out.**print**("Enter gender of the employee:-");

        gender=sc.**next**();

        System.out.**print**("Enter he's/her age:-");

        age=sc.**nextByte**();

        System.out.**print**("Enter he's/her experience:-");

        experience=sc.**nextInt**();

        System.out.**println**("1)Buisness\n2)Marketing\n3)DESIGNING\n4)MAINTAINENCE");

        System.out.**println**("Chosse your depatment no");

        department\_no=sc.**nextInt**();

        switch(department\_no)

        {

            case 1:departmentName="BUISNESS";

            break;

            case 2:departmentName="MARKETING";

            break;

            case 3:departmentName="DESIGNING";

            break;

            case 4:departmentName="MAINTAINENCE";

            break;

            default:departmentName="null";

        }

        sc.**close**();

    }

}

class Allowance extends Employee {

    protected int homeAllowance=0;

    protected int healthAllowance=0;

    protected int travellAllowance=0;

    private int totalAllowance = 0;

**Allowance**()

    {

        super();

    }

    public void **calculateateAllowance**() {

        if (departmentName == "BUISNESS") {

            homeAllowance = 20000;

            healthAllowance = 25000;

            travellAllowance = 50000;

        } else if (departmentName == "MARKETING") {

            homeAllowance = 18000;

            healthAllowance = 25000;

            travellAllowance = 35000;

        } else if (departmentName == "DESIGNING") {

            homeAllowance = 16000;

            healthAllowance = 23000;

            travellAllowance = 30000;

        } else if (departmentName == "MAINTAINENCE") {

            homeAllowance = 15000;

            healthAllowance = 20000;

            travellAllowance = 25000;

        }

        else{

            System.out.**println**("SOMETHING WRONG!");

        }

        totalAllowance = homeAllowance + healthAllowance + travellAllowance;

    }

    public int **getAllowance**() {

        System.out.**println**("Total allowance is :-");

        return this.totalAllowance;

    }

}

public class Ass3\_7 {

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

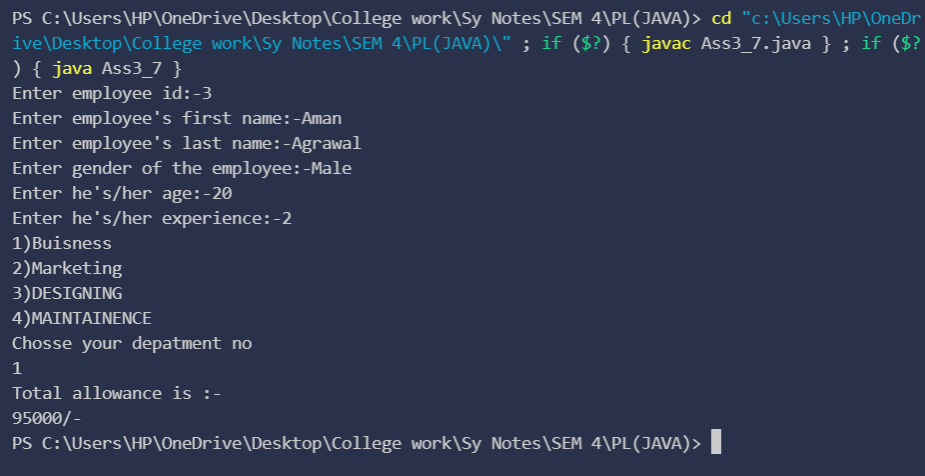
        Allowance allowance=new **Allowance**();

        allowance.**calculateateAllowance**();

        System.out.**println**(allowance.**getAllowance**()+"/-");

    }

}

Output: 

1. Write a Java Program to demonstrate StringBuilder class methods.

Program:

public class Ass3\_8 {

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

        StringBuilder s = new **StringBuilder**();

        s.**append**("AMAN");

        System.out.**println**(s);

        System.out.**println**(s.**capacity**());

        System.out.**println**(s.**indexOf**("M"));

        System.out.**println**(s.**charAt**(2));

        System.out.**println**(s.**lastIndexOf**("A"));

        System.out.**println**(s.**delete**(0, 1));

        System.out.**println**(s.**reverse**());

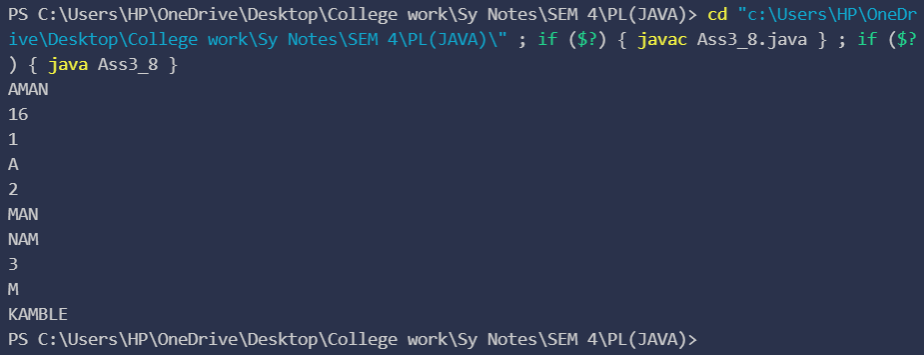
        System.out.**println**(s.**length**());

        System.out.**println**(s.**substring**(2));

        System.out.**println**(s.**replace**(0, s.**length**(), "KAMBLE"));

    }

}

Output: 

1. Write a Java Program to demonstrate Method overriding.( create class Result with method result(). Override method result() in UGResult and PGResult class)

Program:

class Result {

    public void **result**() {

        System.out.**println**("This is Result class");

    }

}

class UGResult {

    public void **result**() {

        System.out.**println**("This is UGResult class.");

    }

}

class PGResult {

    public void **result**()

        {

            System.out.**println**("This is PGResult class.");

        }

}

public class Ass3\_9{

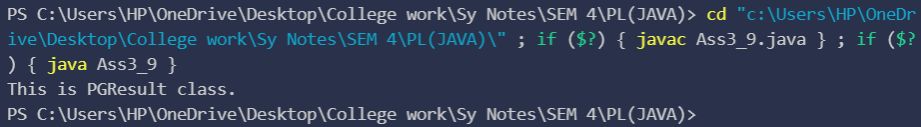
    public static void **main**(String[] *args*) {

                    PGResult pgResult=new **PGResult**();

            pgResult.**result**();

        }

}

Output: 

1. 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.

Program:

class Result {

    public void **result**() {

        System.out.**println**("This is Result class");

    }

}

class UGResult {

    public void **result**() {

        System.out.**println**("This is UGResult class.");

    }

}

class PGResult{

        public void **result**()

        {

            System.out.**println**("This is PGResult class.");

        }

    }

public class Ass2\_10{

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

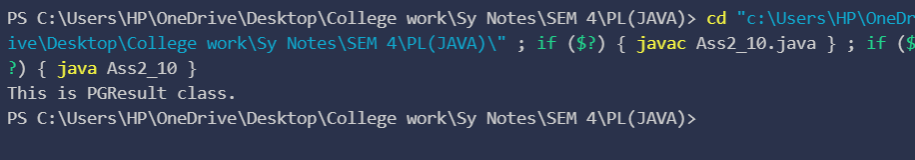
        PGResult pgResult = new **PGResult**();

        pgResult.**result**();

    }

}

Output:



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

Program:

class Animal{

            private String animalType;

            public void **setAnimalType**(String *type*)

            {

                this.animalType=type;

            }

            public String **getAnimalType**() {

                return animalType;

            }

        }

        class Dog extends Animal{

            protected String breedName;

            protected String gender;

**Dog**()

            {

**setAnimalType**("Dog");

            }

            public void **setBrideName**(String *brideName*) {

                this.breedName = brideName;

            }

            public void **setGender**(String *gender*) {

                this.gender = gender;

            }

            public String **getBrideName**() {

                return breedName;

            }

            public String **getGender**() {

                return gender;

            }

        }

        class GermenShefred extends Dog{

        public String name;

            public String color;

**GermenShefred**()

            {

**setBrideName**("GermenShfred");

            }

            public void **setName**(String *name*) {

                this.name = name;

            }

            public void **setColor**(String *color*) {

                this.color = color;

            }

            public String **getName**() {

                return name;

            }

            public String **getColor**() {

                return color;

            }

        }

        class Cat extends Animal{

            protected String brideName;

            protected String gender;

**Cat**()

            {

**setAnimalType**("Cat");

            }

            public void **setBrideName**(String *brideName*) {

                this.brideName = brideName;

        }

        public void **setGender**(String *gender*) {

            this.gender = gender;

        }

    }

    class Persian extends Cat{

        public String name;

        public String color;

**Persian**()

            {

**setBrideName**("Persian");

            }

            public void **setName**(String *name*) {

                this.name = name;

            }

            public void **setColor**(String *color*) {

                this.color = color;

            }

            public String **getName**() {

                return name;

            }

            public String **getColor**() {

                return color;

            }

            public String **getBreedeName**() {

                return null;

            }

        }

        public class Ass2\_11 {

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

                GermenShefred dog=new **GermenShefred**();

                Persian cat=new **Persian**();

                dog.**setName**("Max");

                dog.**setColor**("Black");

                cat.**setName**("XYZ");

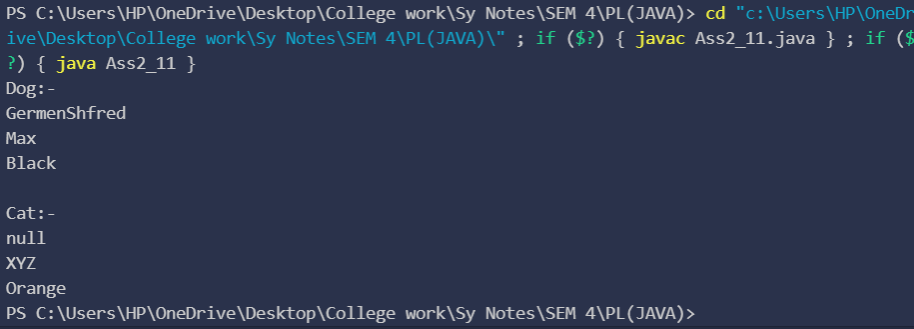
                cat.**setColor**("Orange");

        System.out.**println**(dog.**getAnimalType**()+":-\n"+dog.**getBrideName**()+"\n"+dog.**getName**()+"\n"+dog.**getColor**()+"\n");

        System.out.**println**(cat.**getAnimalType**()+":-\n"+cat.**getBreedeName**()+"\n"+cat.**getName**()+"\n"+cat.**getColor**());

    }

}

Output: 

1. 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.

Program:

interface College{

    String name="Walchand College Of Engginearing Sangli";

}

class Departmet{

   public int dept\_no;

    public String dept\_name;

}

class Teacher extends Departmet implements College{

    String name;

    String qualification;

    void **setInfo**(String *name*,String *qualification*,int *dept\_no*,String *dept\_name*)

    {

            this.name=name;

            this.dept\_name=dept\_name;

            this.dept\_no=dept\_no;

            this.qualification=qualification;

    }

    void **displyData**()

    {

        System.out.**println**("College name:-"+College.name);

        System.out.**println**("Teacher name"+name);

        System.out.**println**("Qualification:-"+qualification);

        System.out.**println**("Department no.:-"+dept\_no);

        System.out.**println**("Department name:-"+dept\_name);

    }

}

 class Ass2\_12{

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

        Teacher t=new **Teacher**();

        t.**setInfo**("Aman","B.tech",1,"COMPUTER SCIENCE");

        t.**displyData**();

    }

}

Output: 