Swapnali Aitwade

[Company name]  [Company address]

NAME :-MAYUR KAMBLE PRN NO:-2020BTECS00110 BATCH:-S7 ASSIGNMENT NO:-2

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

|  | **Method Overloading** | **Method Overriding** |
| --- | --- | --- |
| **1.** | **Method overloading is a compile-time polymorphism.** | **Method overriding is a run-time polymorphism.** |
| **2.** | **It helps to increase the readability of the program.** | **It is used to grant the specific implementation of the method which is already provided by its parent class or superclass.** |
| **3.** | **It occurs within the class.** | **It is performed in two classes with inheritance relationships.** |
| **4.** | **Method overloading may or may not require inheritance.** | **Method overriding always needs inheritance.** |
| **5.** | **In method overloading, methods must have the same name and different signatures.** | **In method overriding, methods must have the same name and same signature.** |
| **6.** | **In method overloading, the return type can or can not be the same, but we just have to change the parameter.** | **In method overriding, the return type must be the same or co-variant.** |

1. **Implement all string functions in java.**

**package Assignment\_No\_2;**

**public class q2 {**

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

**String s="Mayur";**

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

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

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

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

**System.out.println(s.concat(" Kamble"));**

**System.out.println(s.indexOf('r'));**

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

**}**

**}**

**OUTPUT:-**

**5**

**y**

**mayur**

**MAYUR**

**Mayur Kamble**

**4**

**Mayur**

**3)Implement all stringbuffer functions in java.**

**package Assignment\_No\_2;**

**public class q3 {**

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

**StringBuffer s=new StringBuffer("MAYUR");**

**//s.append("MAYUR");**

**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());**

**}**

**}**

**OUTPUT:-**

**MAYUR**

**21**

**0**

**Y**

**1**

**AYUR**

**RUYA**

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

We can create String without using **new** key word like,

String s=”MAYUR”;

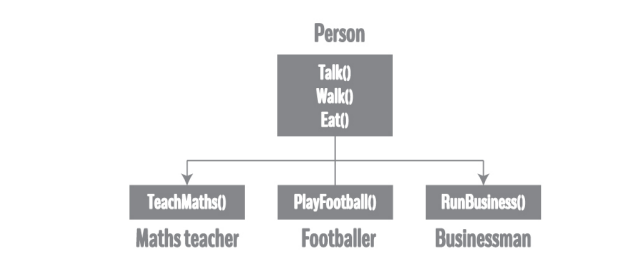
And also we can create String with new Key word like,

String s=new String(“MAYUR”);

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.**
2. package Assignment\_No\_2;
3. class Shape{
4. public  void displayShape()
5. {
6. System.out.println("This is shape.");
7. }
8. }
9. class Rectangle extends Shape{
10. public  void displayRectangle()
11. {
12. System.out.println("This is rectangular shape.");
13. }
14. }
15. class Circle extends Shape{
16. public  void displayCircle()
17. {
18. System.out.println("This is circular shape.");
19. }
20. }
21. class Square extends Rectangle{
22. public  void displaySquare(){
23. System.out.println("Square is rectangle.");
24. }
25. }
26. public class q5{
27. public static void main(String[] args)
28. {
29. Square sq=new Square();
30. sq.displayShape();
31. sq.displayRectangle();
32. }
33. }
34. OUTPUT:-
35. This is shape.
36. This is rectangular shape.
37. **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.**

package Assignment\_No\_2;

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 q6{

    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:-

He can eat.

He can walk.

He can talk.

He can teach maths.

He can eat.

He can walk.

He can talk.

He can do buisness.

He can eat.

He can walk.

He can talk.

He can play football.

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]**

package Assignment\_No\_2;

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 q7 {

    public static void main(String[] args) {

        Allowance allowance=new Allowance();

        allowance.calculateateAllowance();

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

    }

}

OUTPUT:-

Enter employee id:-1

Enter employee's first name:-Mayur

Enter employee's last name:-KAMBLE

Enter gender of the employee:-MALE

Enter he's/her age:-21

Enter he's/her experience:-1

1)Buisness

2)Marketing

3)DESIGNING

4)MAINTAINENCE

Chosse your depatment no

1

Total allowance is :-

95000/-

1. **Write a Java Program to demonstrate StringBuilder class methods.**
2. package Assignment\_No\_2;
3. public class q8 {
4. public static void main(String[] args) {
5. StringBuilder s=new StringBuilder();
6. s.append("MAYUR");
7. System.out.println(s);
8. System.out.println(s.capacity());
9. System.out.println(s.indexOf("M"));
10. System.out.println(s.charAt(2));
11. System.out.println(s.lastIndexOf("A"));
12. System.out.println(s.delete(0,1));
13. System.out.println(s.reverse());
14. System.out.println(s.length());
15. System.out.println(s.substring(2));
16. System.out.println(s.replace(0, s.length(), "KAMBLE"));
17. }
18. }
19. OUTPUT:-
20. MAYUR
21. 16
22. 0
23. Y
24. 1
25. AYUR
26. RUYA
27. 4
28. YA
29. KAMBLE
30. **Write a Java Program to demonstrate Method overriding.( create class Result with method result(). Override method result() in UGResult and PGResult class).**
31. package Assignment\_No\_2;
32. class Result{
33. public void result()
34. {
35. System.out.println("This is Result class");
36. }
37. }
38. class UGResult{
39. public void result(){
41. System.out.println("This is UGResult class.");
42. }
43. }
44. class PGResult{
45. public void result()
46. {
47. System.out.println("This is PGResult class.");
49. }
50. }
51. public class q9 {
52. public static void main(String[] args) {
53. PGResult pgResult=new PGResult();
54. pgResult.result();
56. }
57. }
58. OUTPUT:-
59. This is PGResult class.
60. **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.**
61. **Implement hybrid inheritance using all access specifiers (public, private, protected).**
62. package Assignment\_No\_2;
63. class Animal{
64. private String animalType;
66. public void setAnimalType(String type)
67. {
68. this.animalType=type;
69. }
70. public String getAnimalType() {
71. return animalType;
72. }
73. }
74. class Dog extends Animal{
75. protected String breedName;
76. protected String gender;
77. Dog()
78. {
79. setAnimalType("Dog");
80. }
82. public void setBrideName(String brideName) {
83. this.breedName = brideName;
84. }
85. public void setGender(String gender) {
86. this.gender = gender;
87. }
88. public String getBrideName() {
89. return breedName;
90. }
91. public String getGender() {
92. return gender;
93. }
95. }
96. class GermenShefred extends Dog{
97. public String name;
98. public String color;
99. GermenShefred()
100. {
101. setBrideName("GermenShfred");
102. }
103. public void setName(String name) {
104. this.name = name;
105. }
106. public void setColor(String color) {
107. this.color = color;
108. }
109. public String getName() {
110. return name;
111. }
112. public String getColor() {
113. return color;
114. }
115. }
116. class Cat extends Animal{
117. protected String brideName;
118. protected String gender;
119. Cat()
120. {
121. setAnimalType("Cat");
122. }
123. public void setBrideName(String brideName) {
124. this.brideName = brideName;
125. }
126. public void setGender(String gender) {
127. this.gender = gender;
128. }
130. }
131. class Persian extends Cat{
133. public String name;
134. public String color;
135. Persian()
136. {
137. setBrideName("Persian");
138. }
139. public void setName(String name) {
140. this.name = name;
141. }
142. public void setColor(String color) {
143. this.color = color;
144. }
145. public String getName() {
146. return name;
147. }
148. public String getColor() {
149. return color;
150. }
151. public String getBreedeName() {
152. return null;
153. }
154. }
155. public class q11 {
156. public static void main(String[] args) {
157. GermenShefred dog=new GermenShefred();
158. Persian cat=new Persian();
159. dog.setName("Max");
160. dog.setColor("Black");
161. cat.setName("XYZ");
162. cat.setColor("Orange");
163. System.out.println(dog.getAnimalType()+":-\n"+dog.getBrideName()+"\n"+dog.getName()+"\n"+dog.getColor()+"\n");
164. System.out.println(cat.getAnimalType()+":-\n"+cat.getBreedeName()+"\n"+cat.getName()+"\n"+cat.getColor());
165. }
166. }

OUTPUT:-

Dog:-

GermenShfred

Max

Black

Cat:-

null

XYZ

Orange

**12) 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.**

package Assignment\_No\_2;

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 q12{

    public static void main(String[] args) {

        Teacher t=new Teacher();

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

        t.displyData();

    }

}

OUTPUT:-

College name:-Walchand College Of Engginearing Sangli

Teacher nameMAYUR

Qualification:-B.tech

Department no.:-1

Department name:-COMPUTER SCIENCE