# Department of Computing

# CS 212: Object Oriented Programming

# Class: BESE-11

# Lab 09: Inheritance

# Date: 5th May, 2021

# Instructor: Ms. Hania Aslam

**Learning Objectives**

The learning objective of this lab is to understand and practice the concept of inheritance, a very powerful feature of OOP which helps in code reusability.

**Activity #1.**

What is the output of running the class ActivityOne? Explain the output.

**Hint.** Remember only super-class data members (instance variables and instance methods) are inherited by the sub-class, and constructors are not included.

|  |
| --- |
| class A  **{**  public A**()**  **{**  System**.**out**.**println**(** "A's no-arg constructor is invoked"**);**  **}**  **}**  class B **extends** A **{}**  public class ActivityOne  **{**  public static void main**(** String**[]** args**)**  **{**  B b **=** **new** B**();**  **}**  **}**  **Explanation:**  When we make the object of child class B the default no arg constructor of class B is invoked which subsequently invokes the default constructor of Parent class A and hence we get the output. |

**Output Screenshot:**

****

**Activity #2.**

The following Java program compiles correctly. Show its output, and explain what is happening when an object of Class B is created?

|  |
| --- |
| class A  **{**  public A**()**  **{**  System**.**out**.**println**(** "A's constructor is invoked"**);**  **}**  **}**  class B **extends** A  **{**  public B**(**int t**)**  **{**  System**.**out**.**println**(** "B's constructor is invoked"**);**  **}**  **}**  public class ActivityTwo  **{**  public static void main**(** String**[]** args**)**  **{**  B b **=** **new** B**(**3**);**  **}**  **}**  **Explanation :**  When the object of child class B is created the parameterized constructor of class B is called which subsequently invokes the default costructor of parent class A and hence both the statements are printed. |

**Output Screenshot:**

****

**Activity #3.**

The program fails to compile. Identify the problem and propose a correction? Also explain the reason. Hint: If a constructor does not explicitly call a super constructor as its first statement, a call to super() is automatically added.

|  |
| --- |
| class A  **{**  public A**(** int x**)** **{}**  **}**  class B **extends** A  **{**  public B**()** **{}**  **}**  public class ActivityThree  **{**  public static void main**(** String**[]** args**)**  **{**  B b **=** **new** B**();**  **}**  **}**  **Explanation:**  When we create an object of sub class B it invokes the default constructor of class B which would further invoke the default constructor of the parent class A but there is no default constructor defined for parent class A and hence the compiler throws an error. This error can be rectified by **removing the parameter ‘int x’** from the parameterized constructor of class A hence converting it into a default constructor. |

**Activity #4.**

Identify and correct the problems in the following program so that it compiles and runs fine.

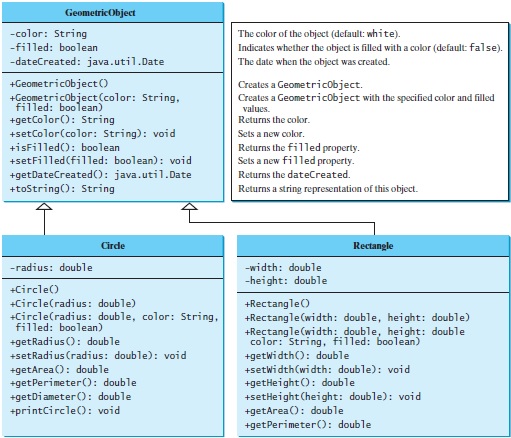
|  |
| --- |
| class Circle  **{**  private double radius**;**  public Circle**(** double radius**)**  **{**  radius **=** radius**;**  **}**  public double getRadius**()**  **{**  **return** radius**;**  **}**  public double getArea**()**  **{**  **return** radius **\*** radius **\*** Math**.**PI**;**  **}**  **}**  class B **extends** Circle  **{**  private double length**;**    B**(** double radius**,** double length**)**  **{**  Circle**(** radius**);**  length **=** length**;**  **}**  /\*\* Override getArea() \*/  public double getArea**()**  **{**  **return** getArea**()** **\*** length**;**  **}**  **}**  public class ActivityFour  **{**  public static void main**(** String**[]** args**)**  **{**  B b **=** **new** B**(** 5**,** 10**);**  System**.**out**.**println**(** "Area = " **+** b**.**getArea**());**  **}**  **}** |

**Rectified Code:**

package com.company;  
  
class Circle1  
{  
 private double radius;  
  
 Circle1(){}  
 public Circle1( double *radius*)  
 {  
 this.radius = *radius*;  
 }  
  
 public double getRadius()  
 {  
 return radius;  
 }  
  
 public double getArea()  
 {  
 return radius \* radius \* Math.*PI*;  
 }  
}  
  
class B extends Circle1  
{  
 private double length;  
  
 B( double *radius*, double *length*)  
 {  
 super( *radius*);  
 this.length = *length*;  
 }  
  
 */\*\* Override getArea() \*/* public double getArea()  
 {  
 return super.getArea() \* length;  
 }  
}  
  
  
  
public class ActivityFour  
{  
 public static void main( String[] *args*)  
 {  
 B b = new B( 5, 10);  
 System.*out*.println( "Area = " + b.getArea());  
 }  
}

**Task #1:**

The following UML class diagram illustrates an inheritance relationship, wherein the classes Circle and Rectangle have been extended from the class GeometricObject.



You’re required to implement the classes GeometricObject and Rectangle.

The Rectangle class contains:

* Two double data fields named width and height that specify the width and height of the rectangle. The default values are 1.0 for both width and height.
* A no-arg constructor that creates a default rectangle.
* A constructor that creates a rectangle with the specified width and height.
* A method named getArea() that returns the area of this rectangle.
* A method named getPerimeter() that returns the perimeter.
* A method named toString() that returns a string description for the rectangle.

The toString() method shall be implemented as follows:

return "Rectangle: width = " + width + " height = " + height;

Write a test program that prompts the user to enter width and height of the rectangle, a color, and a Boolean value to indicate whether the rectangle is filled. The program should create a Rectangle object and set the color and filled properties using the input. The program should display the area, perimeter, color, and true or false to indicate whether it is filled or not.

**Hint:**

In Java, getting the current date is as simple as instantiating the Date object from the Java package java.util:

java.util.Date date=new java.util.Date();

System.out.println(date);

// You may need to use Date class while implementing the getDateCreated() method of GeometricObject class.

**Code:**

**GeometricObject Class:**

package com.company;  
  
*//Creating our main class GeometricObject*public class GeometricObject {  
 *//Declaring fields* String color = "white";  
 boolean filled = false;  
 java.util.Date date = new java.util.Date();  
  
 *//Defining the no arg constructor* GeometricObject(){  
 }  
 *//Defining the parameterized constructor* GeometricObject(String *color*, boolean *filled*){  
 this.color = *color*;  
 this.filled = *filled*;  
 }  
  
 *//Color Setter* public void setColor(String *color*){  
 this.color = *color*;  
 }  
 *//Color Getter* public String getColor(){  
 return color;  
 }  
 *//Filled Setter* public void setFilled(boolean *filled*){  
 this.filled = *filled*;  
 }  
 *//Filled Getter* public boolean isFilled(){  
 return filled;  
 }  
 *//Date Getter* public java.util.Date getDateCreated(){  
 return date;  
 }  
 public String toString(){  
 return "Geometric Object: color = " + color + " filled = " + filled + " date = " + date;  
 }  
}

**Circle Class:**

package com.company;  
  
*//Creating sub class Circle of GeometricObject class*public class Circle extends GeometricObject{  
 *//Declaring fields* double radius;  
  
 *//Defining the no arg constructor* Circle(){  
 }  
 *//Defining the single parameterized constructor* Circle(double *radius*){  
 this.radius = *radius*;  
 }  
 *//Defining the triple parameterized constructor* Circle(double *radius*, String *color*, boolean *filled*){  
 this.radius = *radius*;  
 this.color = *color*;  
 this.filled = *filled*;  
 }  
  
 *//Radius Setter* public void setRadius(double *radius*){  
 this.radius = *radius*;  
 }  
 *//Radius Getter* public double getRadius(){  
 return radius;  
 }  
 *//Method for calculating Area* public double getArea(){  
 return Math.*PI* \* radius \* radius;  
 }  
 *//Method for calculating Perimeter* public double getPerimeter(){  
 return 2 \* Math.*PI* \* radius;  
 }  
 *//Method for calculating Diameter* public double getDiameter(){  
 return 2 \* radius;  
 }  
 *//Printing details of Circle* public void printCircle(){  
 System.*out*.println("The radius of circle is: " + radius);  
 System.*out*.println("The area of circle is: " + getArea());  
 System.*out*.println("The perimeter of circle is: " + getPerimeter());  
 }  
}

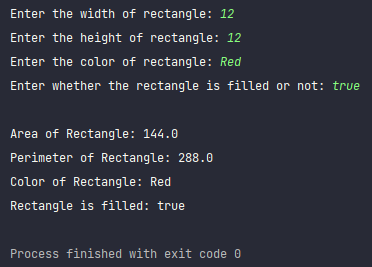
**Rectangle Class:**

package com.company;  
  
*//Creating sub class Rectangle of Geometric class*public class Rectangle extends GeometricObject{  
 *//Declaring fields* double width = 1.0;  
 double height = 1.0;  
  
 *//Defining the no arg constructor* Rectangle(){  
 }  
 *//Defining the dual parameterized constructor* Rectangle(double *width*, double *height*){  
 this.width = *width*;  
 this.height = *height*;  
 }  
 *//Defining the 4-parameter parameterized constructor* Rectangle(double *width*, double *height*, String *color*, boolean *filled*){  
 this.width = *width*;  
 this.height = *height*;  
 this.color = *color*;  
 this.filled = *filled*;  
 }  
  
 *//Width Setter* public void setWidth(double *width*){  
 this.width = *width*;  
 }  
 *//Width Getter* public double getWidth(){  
 return width;  
 }  
 *//Height Setter* public void setHeight(double *height*){  
 this.height = *height*;  
 }  
 *//Height Getter* public double getHeight(){  
 return height;  
 }  
 *//Method to calculate Area* public double getArea(){  
 return width \* height;  
 }  
 *//Method to calculate Perimeter* public double getPerimeter(){  
 return 2 \* (width \* height);  
 }  
 public String toString(){  
 return "Rectangle: width = " + width + " height = " + height;  
 }  
}

**Test Class:**

package com.company;  
  
import java.util.Scanner;  
  
*//Creating the Test class*public class TestTask01 {  
  
 *//Main method* public static void main(String[] *args*) {  
  
 *//Creating instance of Scanner class to take input* Scanner sc = new Scanner(System.*in*);  
  
 *//Taking inputs* System.*out*.print("Enter the width of rectangle: ");  
 double width = sc.nextDouble();  
 System.*out*.print("Enter the height of rectangle: ");  
 double height = sc.nextDouble();  
 System.*out*.print("Enter the color of rectangle: ");  
 String color = sc.next();  
 System.*out*.print("Enter whether the rectangle is filled or not: ");  
 boolean filled = sc.nextBoolean();  
  
 *//Creating object of Rectangle class* Rectangle rect = new Rectangle();  
 rect.setColor(color); *//Setting color* rect.setFilled(filled); *//Setting filled* rect.setWidth(width); *//Setting width* rect.setHeight(height); *//Setting height* System.*out*.println();  
  
 *//Printing required outputs* System.*out*.println("Area of Rectangle: " + rect.getArea());  
 System.*out*.println("Perimeter of Rectangle: " + rect.getPerimeter());  
 System.*out*.println("Color of Rectangle: " + rect.getColor());  
 System.*out*.println("Rectangle is filled: " + rect.isFilled());  
  
 }  
}

**Output Screenshot:**

****

**Task #2:**

Design a class named Person and its two subclasses named Student and Employee. Make Faculty and Staff subclasses of Employee.

A person has a name, address, phone number, and email address. A student has a class status (freshman, sophomore, junior, or senior). Define the status as a constant. An employee has an office, salary, and date hired. A faculty member has office hours and a rank. A staff member has a title. Override the toString() method in each class to display the class name and the person’s name.

Write a test program that creates a Person, Student, Employee, Faculty, and Staff, and invokes their toString() methods.

**Code:**

**Person Class:**

package com.company;  
  
*//Creating the main class Person*public class Person {  
 *//Declaring fields* String name;  
 String address;  
 long phone\_number;  
 String email;  
  
 *//Method to print class name and person name* public String toString(){  
 System.*out*.println("Class Name: Person");  
 System.*out*.println("Person Name: Ali");  
 return null;  
 }  
  
}

**Student Class:**

package com.company;  
  
*//Creating a sub class Student of Person class*public class Student extends Person{  
 *//Declaring fields* String status;  
  
 *//Method to print class name and person name* public String toString(){  
 System.*out*.println("Class Name: Student");  
 System.*out*.println("Person Name: Hamza");  
 return null;  
 }  
}

**Employee Class:**

package com.company;  
  
*//Creating sub class Employee of Person class*public class Employee extends Person{  
 *//Declaring fields* String office;  
 int salary;  
 String date\_hired;  
  
  
 *//Method to print class name and person name* public String toString(){  
 System.*out*.println("Class Name: Employee");  
 System.*out*.println("Person Name: Ahmed");  
 return null;  
 }  
}

**Faculty Class:**

package com.company;  
  
*//Creating sub class Faculty of Employee class*public class Faculty extends Employee{  
 *//Declaring fields* int office\_hrs;  
 String rank;  
  
 *//Method to print class name and person name* public String toString(){  
 System.*out*.println("Class Name: Faculty");  
 System.*out*.println("Person Name: Babar");  
 return null;  
 }  
}

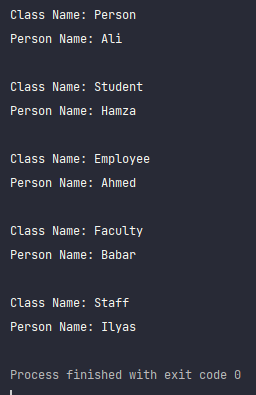
**Staff Class:**

package com.company;  
  
*//Creating sub class Staff of Employee class*public class Staff extends Employee {  
 *//Declaring fields* String title;  
  
 *//Method to print class name and person name* public String toString(){  
 System.*out*.println("Class Name: Staff");  
 System.*out*.println("Person Name: Ilyas");  
 return null;  
 }  
}

**Test Class:**

package com.company;  
  
*//Creating the Test class*public class TestTask02 {  
  
 *//Main method* public static void main(String[] *args*) {  
  
 *//Invoking toString method for each object of every class* Person person = new Person();  
 person.toString();  
  
 System.*out*.println();  
  
 Student student = new Student();  
 student.toString();  
  
 System.*out*.println();  
  
 Employee employee = new Employee();  
 employee.toString();  
  
 System.*out*.println();  
  
 Faculty faculty = new Faculty();  
 faculty.toString();  
  
 System.*out*.println();  
  
 Staff staff = new Staff();  
 staff.toString();  
 }  
}

**Output Screenshot:**

****

**Hand in**

Hand in the source code from this lab at the appropriate location on the LMS system. You should hand in a single file named Lab\_9\_<Your CMS\_ID. Your\_NAME >.docx (without angle brackets) that contains ONLY the following.

1. All completed java source code representing the work accomplished for this lab: The code snippets should contain author information in the comments at the top.

**To Receive Credit**

1. Comment your program heavily. Intelligent comments and a clean, readable formatting of your code account for 20% of your grade.
2. The lab time is not intended as free time for working on your programming/other assignments.