ICS 102 LAB #11 Classes, Constructors and Visibility Modifiers

# Objectives:

* Learn how to declare classes and how to use the **private**, **public** and default **visibility modifiers**

Java is an Object-Oriented Language. As a language that has the Object Oriented feature, Java supports the following fundamental concepts:

* Classes
* Objects
* Message passing: Objects can send and receive messages by method invocation (i.e., method calls)

* **Object -** Objects have states and behaviors. Example: A dog has states - color, name, breed as well as behaviors -wagging, barking, and eating. An object is an instance of a class.
* **Class -** A class can be defined as a template/blue print that describes the behaviors/states that object of its type support.

# Defining classes and Visibility Modifiers

A Java program consists of one or more classes. A class is a template for objects and an object is an instance of a class. A class is declared using the class keyword. The general form of a class definition is:

access-specifier class ClassName

{

class variable declarations // discussed in Lab11 and 12

instance variable declarations

Constructors

class methods // discussed in Lab11 and 12

instance methods

}

Collectively the methods, class variables, and instance variables defined within a class are called members of the class.

Three of Java **access-specifiers (**Visibility Modifiers) are **public,** **private**, and if no specifier is present a default level access. An access-specifier determines how a class or a class member can be accessed.

* **public**: Visible to any class that can see the class, i.e., grants access to anyone in the same package and other packages that import the class.
* package (default): Visible to all classes of the same package.
* **private**: Visible only to the class itself.

**3. Types of variables**

A class can contain any of the following variable types:

* **Local variables:** Variables defined inside methods, constructors or blocks are called local variables. The variable will be declared and initialized within the method and will be destroyed when the method has completed. A local variable cannot have access modifier.
* **Instance variables:** Instance variables are variables within a class but outside any method. These variables are instantiated when the class is loaded. Instance variables can be accessed from inside any method, constructor or blocks of that particular class.
* **Class variables:** Class variables are variables declared within a class, outside any method, with the **static** keyword.

A class can have any number of methods to access the value of various kinds of variables.

# 4. Constructors and the *this* keyword

A *constructor* in Java is a block of code that is called when an instance of an object is created. Since a constructor is used to create objects, instance variable initialization code is normally placed in a constructor. Here are the key differences between a constructor and a method:

* A constructor does not have a return type.
* The name of the constructor must be the same as the name of the class.
* Unlike methods, constructors are not considered members of a class.
* A constructor is called automatically when a new instance of an object is created using the **new** keyword.
* A constructor cannot be static or final.

The syntax of a constructor is:

access-specifier ClassName (parameter-list) [throws exception...]

{

statements...

}

**Note:**

* A constructor can throw exceptions if it encounters error situations that it cannot recover from.
* Every Class in Java has constructor, if no explicit constructor is specified in a class by a programmer, Java Compiler inserts a no argument constructor inside the class.

# The word *this* is an implicit reference variable that refers to the current object. One use of *this* is to refer to the instance variables of the class, especially when their names happen to be the same with parameter names or local variable names.

Example:

**public class Student{**

**private final int id;**

**private String name;**

**private String major;**

**private double gpa;**

**public Student(int id, String name, String major, double gpa){**

**this.id = id;**

**this.name = name;**

**this.major = major;**

**this.gpa = gpa;**

**}**

**// . . .**

**}**

**4. Object Creation**

An object is created by calling a constructor using the **new** operator.

Examples:

**String str = new String("ICS 102");**

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

**Scanner fileScanner = new Scanner(new FileInputStream("input.txt"));**

**Student st1 = new Student(980000, "Qasim Yusuf", "COE", 3.5);**

Once created, an object’s reference variable can be used to access a method of the object by using the dot operator and a method call:

**objectReferenceVariable.methodName(parameterList)**

Examples:

**int strLength = str.length();**

**String str2 = str.toUpperCase();**

**double number = kbScanner.nextDouble();**

**int studentID = st1.getID();**

**st1.setGPA(4.0);**

Non-private variables of an object can also be accessed by using:

**objectReferenceVariable.variableName**

**4. Driver classes**

Note: It is better to implement a class Y and its driver class (a class in which we create and use objects of Y) as public classes in different Java files; however a Java file may contain more than one class provided only one class is declared as public. This is usually the driver class that contains the main method.

Example# 1:

**class Access{**

**public int x;**

**private int y;**

**int z; // instance variable with default or package access**

**Access(int x, int y, int z){**

**this.x = x;**

**this.y = y;**

**this.z = z;**

**}**

**public int getY(){**

**return y;**

**}**

**public void setY(int newY){**

**y = newY;**

**}**

**}**

**public class AccessDriver {**

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

**Access obj = new Access(6, 8, 3);**

**/\* A public instance variable can be accessed directly from other classes in its package**

**and from classes in other packages that import the class**

**\*/**

**System.out.println("x = " + obj.x);**

**/\* A public variable can be modified directly from other classes in its package and**

**from classes in other packages. THIS IS DANGEROUS because everyone is able to**

**modify the variable.**

**\*/**

**obj.x= 15;**

**System.out.println("x = " + obj.x);**

**/\* An instance variable with default or package access can be accessed directly from**

**other classes in its package; but not from classes in other packages.**

**\*/**

**System.out.println("z = " + obj.z);**

**/\* An instance variable with default or package access can be modified directly from**

**other classes in its package; but not from other packages.**

**\*/**

**obj.z = 20;**

**System.out.println("z = " + obj.z);**

**// A private instance variable cannot be accessed directly from outside its class**

**// System.out.println(obj.y); // this generates compile-time error**

**// The only way to access a private variable is to use a get- or accessor-method:**

**System.out.println("y = " + obj.getY( ));**

**// The only way to modify a private variable is to provide a set-method (or mutator-**

**method)**

**// obj.y = 12; // this generates compile-time error**

**// This is the correct way to modify a private variable:**

**obj.setY(12);**

**System.out.println("y = " + obj.getY( ));**

**}**

**}**

Example# 2:

The following shows two classes. A class Box and a class, BoxDriver, which creates two Box objects.

**class Box {**

**private double length , width , height;**

**public Box(double boxLength , double boxWidth , double boxHeight)**

**throws IllegalArgumentException{**

**if(boxLength <= 0 || boxWidth <= 0 || boxHeight <= 0)**

**throw new IllegalArgumentException("Error: Dimension <= 0");**

**length = boxLength;**

**width = boxWidth;**

**height = boxHeight;**

**}**

**public double volume( ){**

**return length \* width \* height;**

**}**

**public double surfaceArea( ) {**

**return 2\*(length\*width + length\*height + width\*height);**

**}**

**}**

**public class BoxDriver {**

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

**Box box1 = null, box2 = null; // Create two Box objects**

**try{**

**box1 = new Box(20.0, 10.0, 15.0); // normally the values will be read**

**box2 = new Box(6.0, 4.0, 2.0);**

**}**

**catch(IllegalArgumentException e){**

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

**System.exit(1);**

**}**

**double volume, area;**

**// Get and display volume and surface area of box1 and box2**

**volume = box1.volume( );**

**area = box1.surfaceArea( );**

**System.out.println("The volume of box1 is " + volume + " cubic cm");**

**System.out.println("The surface area of box1 is "+area+" square cm\n");**

**System.out.println("The volume of box2 is " + box2.volume( ) + " cubic cm");**

**System.out.println("The surface area of box2 is "+ box2.surfaceArea() +**

**" square cm\n");**

**}**

**}**

**Note:** In case of invalid parameters, the Box constructor throw an **IllegalArgumentException** rather than display an error message using:

System.out.println("Error: Invalid parameter");

This is good programming practice; the decision on what is to be done in case of error should be done by the driver program. To enable this, the driver program should call these methods in a try-catch block. Example:

**Box box1 = null, box2 = null;**

**try{**

**box1 = new Box(20.0, 10.0, 15.0); // normally the values will be read**

**box2 = new Box(6.0, 4.0, 2.0);**

**}catch(IllegalArgumentException e){**

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

**System.exit(1);**

**}**

Example#3:

public class Student{

**private final int studentID;** // a final variable must have no corresponding set method

private String studentName;

private String major;

private double gpa;

public Student(int id, String name, String theMajor, double theGpa){

studentID = id;

studentName = name;

major = theMajor;

gpa = theGpa;

}

// Accessor or get methods:

public String getName( ){

return studentName;

}

public String getMajor( ){

return major;

}

public double getGPA( ){

return gpa;

}

public int getID(){

return studentID ;

}

// Mutator or set methods:

public void setStudentName(String theName){

studentName = theName;

}

public void setStudentMajor(String theMajor){

major = theMajor;

}

public void setStudentGPA(double theGPA) throws IllegalArgumentException{

if(theGPA < 0 || theGPA > 4)

throw new IllegalArgumentException("Error: Invalid GPA");

gpa = theGPA;

}

}

public class StudentDriver {

public static void main(String[ ] args){

Student student1 = new Student(20301111, "Muhsin Qasim", "MIS", 3.6);

System.out.println("Student1: ID: " + student1.getID() + ", Name: "

+ student1.getName() + ", Major: " + student1.getMajor() + ", GPA: " + student1.getGPA());

System.out.println();

Student student2 = new Student(2000650, "Yusuf Ahmad", "COE", 3.5);

System.out.println("Student2 before GPA modification: ");

System.out.println("Student2: ID: " + student2.getID() + ", Name: " + student2.getName()

+ ", Major: " + student2.getMajor() + ", GPA: " + student2.getGPA());

// Modify GPA of student2

student2. setStudentGPA (4.0);

System.out.println("\nStudent2 after GPA modification: ");

System.out.println("Student2: ID: " + student2.getID() + ", Name: " + student2.getName()

+ ", Major: " + student2.getMajor() + ", GPA: " + student2.getGPA());

}

}

**Example#4:**

public class Point{

private int x;

private int y;

private String position;

public Point(int x, int y){

this.x = x;

this.y = y;

position = initializePosition( );

}

public int getX( ){

return x;

}

public int getY( ){

return y;

}

public String getPosition( ){

return position;

}

public void setX(int x){

this.x = x;

position = initializePosition( );

}

public void setY(int y){

this.y = y;

position = initializePosition( );

}

public double distance(Point point){

return Math.hypot(this.x - point.x , this.y - point.y);

}

public void translate(int dx, int dy){

x += dx;

y +=dy;

position = initializePosition( );

}

public void swapCoordinates( ){

int temp = x;

x = y;

y = temp;

position = initializePosition( );

}

**private String initializePosition( ){**

String pstn;

if(x == 0 && y == 0)

pstn = "origin";

else if(x > 0 && y == 0)

pstn = "positive x-axis";

else if(x < 0 && y == 0)

pstn = "negative x-axis";

else if(y > 0 && x == 0)

pstn = "positive y-axis";

else if(y < 0 && x == 0)

pstn = "negative y-axis";

else if(x > 0 && y > 0)

pstn = "first quadrant";

else if(x < 0 && y > 0)

pstn = "second quadrant";

else if(x < 0 && y < 0)

pstn = "third quadrant";

else

pstn = "fourth quadrant";

return pstn;

}

}

public class PointDriver {

public static void main(String[ ] args) {

Point point1 = new Point(3, -2);

Point point2 = new Point(5, 4);

System.out.println("Point1: x = " + point1.getX( ) + ", y = " + point1.getY( ));

System.out.println("Point2: x = " + point2.getX( ) + ", y = " + point2.getY( ));

double dist = point1.distance(point2);

System.out.printf("The distance between Point1 and Point2 is %.2f cm%n", dist);

System.out.println("\nPoint1 before swapping coordinates:");

System.out.println("Point1: x = " + point1.getX( ) + ", y = " + point1.getY( ));

System.out.println("Point1 after swapping coordinates:");

point1.swapCoordinates( );

System.out.println("Point1: x = " + point1.getX( ) + ", y = " + point1.getY( ));

System.out.println("\nPoint2 before translation:");

System.out.println("Point2: x = " + point2.getX( ) + ", y = " + point2.getY( ));

System.out.println("Point2 after translation:");

point2.translate(-5, -6);

System.out.println("Point2: x = " + point2.getX( ) + ", y = " + point2.getY( ));

}

}

**Lab Tasks: Note: Your solutions for these tasks are required in Lab 12**

**Task#1:**

1. Modify Example#2 **Box** class as follows:

* Modify the given constructor such that each Constructor or method parameter has the same name as the corresponding instance variable.

* Add appropriate set- and get-methods. The set methods must throw IllegalArgumentException. The parameters to the set-methods must have the same names as the instance variables.

1. Modify Example#2 **BoxDriver** class such that it calls an appropriate methods to display the following output:

The volume of box1 is 3000.0 cubic cm

The surface area of box1 is 1300.0 square cm

The volume of box2 is 48.0 cubic cm

The surface area of box2 is 88.0 square cm

Box1 before length modification:

Length: 20.0 cm

Box1 after length modification:

Length: 40.0 cm

**Task#2:**

1. Implement a **Sphere** class that has:

* private instance variables: radius, xCenter, yCenter each of type double
* Appropriate constructor that throws IllegalArgumentException if radius < 0
* Appropriate accessor (or get) methods
* Appropriate set methods one of which throws IllegalArgumentException if

radius < 0

* calculateVolume method.
* calculateSurfaceArea method.

Note: where r is the radius

1. Implement **SphereDriver** class such that it creates a **Sphere** object, referenced by **sphere1**, with radius 5.00 cm and center (1.00, 3.00). It then modifies the **Sphere** object such that its radius is multiplied by **2.0**, its center x-coordinate is translated by **+3.0** and its center

y-coordinate is translated by **-12.0**.

The **main** method must handle **InputMismatchException** and **IllegalArgumentException**. The output of your program must be in the form:

**Sample run:**

Enter sphere 1 radius: 5.0

Enter x, y coordinates of sphere 1 center: 1.0 3.0

Sphere 1 before modification:

Center (1.00, 3.00)

Radius = 5.00 cm

Volume = 523.60 cubic cm

Surface Area = 314.16 square cm

Sphere 1 After modification:

Center (4.00, -9.00)

Radius = 10.00 cm

Volume = 4188.79 cubic cm

Surface Area = 1256.64 square cm

**Task#3:**

Implement a **Student** class with the following instance variables, constructors and instance methods:

**Instance Variables:**

private String name;

private double totalScore;

private int numberOfQuizzes;

**Constructor:**

public Student(String name) throws IllegalArgumentException

*// throws exception if name is null or is a string with zero or more blank characters.*

**Instance Methods:**

public String getName( )

public double getAverage( ) throws UnsupportedOperationException *if no quiz has*

*been taken.*

public double getTotalScore( )

public void addQuiz(double score) throws IllegalArgumentException if score < 0

Write an application **TestStudent** that prompts for and reads a student name. It then creates a Student object. It then prompts for and reads the number of quizzes **n** that the student has taken. It then prompts for and reads the scores of the student in **n** quizzes and adds each to the ***totalScore*** of the student using ***addQuiz( )*** method. Finally, the application prints the student name and his quiz average.

The **main** method must handle java.util.InputMismatchException, java.lang.IllegalArgumentException and java.lang.UnsupportedOperationException.

**Sample program runs:**

|  |
| --- |
|  |
|  |
|  |
|  |