Classes & Objects

MODULE NAME: OBJECT ORIENTED PROGRAMMING WITH JAVA

MODULE CODE: CST 05206

INSTRUCTORS: JAMILA L.

Introduction

A class is a blueprint or prototype from which objects are created.

An object is a software bundle of related state and behaviour.

- Software objects are often used to model the real-world objects that you find in everyday life.

*** *Before we learn classes and objects, let understand variables, access modifiers and how we can use them in objects and classes*

Access modifiers

The access modifiers in Java specifies the accessibility or scope of a field, method, constructor, or class.

We can change the access level of fields, constructors, methods, and class by applying the access modifier on it.

Access Modifier	within class	within package	outside package by subclass only	outside package
Private	Y	N	N	N
Default	Y	Y	N	N
Protected	Y	Y	Y	N
Public	Y	Y	Y	Y

Access modifiers cont...

There are four types of Java access modifiers:

Private: The access level of a private modifier is only within the class. It cannot be accessed from outside the class.

Default: The access level of a default modifier is only within the package. It cannot be accessed from outside the package. If you do not specify any access level, it will be the default. It provides more accessibility than private. But, it is more restrictive than protected, and public.

Protected: The access level of a protected modifier is within the package and outside the package through child class. If you do not make the child class, it cannot be accessed from outside the package.

Public: The access level of a public modifier is everywhere. It can be accessed from within the class, outside the class, within the package and outside the package.

Objects

Creating an object

- A keyword new is used to create an object of class.
- A statement that creates an object of a class declares the object, instantiates the object and calls a constructor of the class if any.
- When a constructor is called it initializes the member variables of a class.

Creating an object cont…

- Declaration: The code before an assignment operator is all variable declarations that associate a variable name with a reference or class type.
- Instantiation: The new keyword is a Java operator that creates the object (i.e. creating a memory space for a declared object).
- Initialization: The new operator is followed by a call to a constructor, which initializes the new object.

Life cycle of an object

- Use an object
  - Manipulation of attributes (myObject.myAttribute)
  - Invocation of methods (myObject.myMethod())
- Destroy the object
  - When *no references* are made to the object
  - Java handles this automatically – garbage collection

Declaring an object variable

- Declare a reference variable as follows:
- E.g.,
  - Bicycle mountainBike = new Bicycle();
  - Bicycle racingBike = new Bicycle();
  - Bicycle tandemBike = new Bicycle();
- There are three instances (objects) of the class Bicycle.

Classes

Attributes and behaviour of a class

- A Java class consists of two distinct types of information, attributes and behaviour.
- Attributes are data that differentiate one object from another.
- They can be used to determine the appearance, state, and other qualities of objects that belong to that class.
- In a class, attributes are defined by variables – places to store information in a computer program.
- An instance variable defines an attribute of one particular object.
- The object's class defines what kind of attribute it is, and each instance stores its own value for that attribute.

Attributes and behavior of a class cont...

- Instance variables are attributes that have values that differ from one object to another. e.g. int age;
- A class variable defines an attribute of an entire class.
- The variable applies to the class itself and to all of its instances, so only one value is stored no matter how many objects of that class have been created. e.g. String surname;

Attributes and behavior of a class cont...

- Behavior refers to the things that a class of object can do to themselves and other objects.
- Behavior can be used to change the attributes of an object, receive information from other objects, and send messages to other objects asking them to perform tasks.
- Behaviour for a class is implemented using methods.

Attributes and behaviour of a class cont…

- Methods are groups of related statements in a class of objects that handle a task.
- Objects communicate with each other using methods
- Instance methods are used when you are working with an object of the class.
- Class methods apply to class itself.
- These methods have *static* keyword modifier.
- Collectively, the methods and variables defined within a class are called members of a class

Example cont…

Example of Instance and class variable:

```
public class ClassDemo {
    static String surname;  // class variable
    String name;                    // instance variable
    int age;                            // instance variable
    ClassDemo(String surname1, String name1, int age1){
//constructor
        surname=surname1;    // variable initialization
        name=name1;
        age=age1;
    }

  void printMemberDetails(){ // method for printing

    System.out.println(name+" "+surname+" is "+age+" years old" );

  }
```

Example cont…

```
   // main method
public static void main(String args[]){
   System.out.println("The first family member");

// creating son object – son is object reference
   ClassDemo son=new ClassDemo("Malisa", "Amani",20);
      son.printMemberDetails();          // calling method
      System.out.println("\nThe second family member");
      ClassDemo dad=new ClassDemo("Malisa", "Alpha",63);
      dad.printMemberDetails();
  }
  }
```

Methods

Methods

- Define behaviour of objects
- Used for communication between objects

Syntax:

```
modifiers return-type methodName (parameterList)
{
    //Method body
}
```

Methods cont...

- Parameter list is a comma separated list of parameter declarations
- Each parameter identifies the parameter's type and a name by which the parameter can be referenced
- A parameter may be referenced anywhere within the method.
- Methods are used by invoking them with respect to an object reference

Syntax :

objectReference.methodName(argu-ment List);

e.g.

```
String treeName = tree.getName( );
```

Methods cont…

- Static methods apply to the class as a whole rather than an instance of the class
- Static methods can be invoked using the same notation as non-static methods
- However, it is more appropriate to invoke static methods using the class name rather than a variable name.

Constructors

- A constructor is a special method which is called at the creation of an object from a class.
- Used for the initialization of an object of a class
- If not specified in a class, the default constructor is used, usually provided by a Java compiler
- Constructor declarations look like method declarations – except that they use the name of the class and have no return type.

Constructors cont…

- For example, creating a constructor for Bicycle class

```
public Bicycle( int startSpeed, int startGear) {
        speed = startSpeed;
    gear = startGear;
}
```

Constructors cont…

- To create a new Bicycle object called myBike, a constructor is called by the *new* operator:
- Bicycle myBike = new Bicycle(0, 1);
- new Bicycle(0, 1) creates space in memory for the object and initializes its fields
- More than one constructor is allowed
- Bicycle yourBike = new Bicycle(); invokes the no-argument constructor to create a new Bicycle object called yourBike.
- If no constructor created, the compiler automatically provides a no-argument default constructor for any class without constructors.

Constructors cont...

Constructors look a lot like regular methods, with three basic differences:

- They always have the same name as the class.
- They do not have a return type.
- They cannot return a value in the method by using the return statement.

set and get methods

- A class's private fields can be manipulated only by methods of that class.
- So a client of an object – that is, any class that calls the object's methods – calls the class's public methods to manipulate the private fields of an object of the class.
- Classes often provide public methods to allow clients of the class to set (i.e., assign values to) or get (i.e., obtain the values of) private instance variables.
- The names of these methods need not begin with set or get, but this naming convention is highly recommended in Java.

set and get methods cont…

```
 /*program demonstrates the use of set and get
methods */

    public class Rectangle {

    private double length, width;
Rectangle(double l, double w){ setLength(l);

    setWidth(w);

 }
```

set and get methods cont...

```
//set method for length
    public void setLength(double l){
     length=l;
    }
    //get method for length
     public double getLength(){
    return length;
    }
```

set and get methods cont…

```
//set method for width
    public void setWidth(double w){
    width=w;
    }
    //get method for width
    public double getWidth(){
    return width;
    }
```

set and get methods cont…

```
 //method for calculating area
 public double calculateArea(){ double
area;//local variable
     //use of get methods
     area=getLength()*getWidth();
     return area;
     } //end method calculateArea
 }//end class Rectangle
```

set and get methods cont…

```
class RectangleTest{

    public static void main(String[] args){

        Rectangle rect=new Rectangle(5,3);

        System.out.println("Initial area  of
the rectangle is     "+rect.calculateArea());
```

set and get methods cont…

```
 /*changing values of length and width by using set
methods*/

     rect.setLength(10);

     rect.setWidth(6);

     //display new value of area
System.out.println("\nNew area of the rectangle is "
+rect.calculateArea()); }//end method main

 }//end class RectangleTest
```

Thank you!

for listen