# Chapter Two

# More on Classes and Objects

## Overview

When you faced with a problem domain and you have to architect a solution. Given that Java is an object-oriented language, you should perform an object-oriented design. In this process, you will end up with classes, objects, attributes, and behaviors/methods.

Object-oriented design process involves the following three tasks:

* Dividing the problem domain into types of objects/classes,
* Modeling the relationships between classes
* Modeling the attributes and behaviors /methods of each type.

These tasks are not listed in any particular order. Most likely, you will perform these tasks iteratively throughout the design process.

In an object-oriented design:

* You identify the fundamental objects of the problem domain, the "things" involved.
* You then classify the objects into types by identifying groups of objects that have common characteristics and behaviors.

The types of objects you identify in the problem domain become "types" in your solution. The program you write will create and manipulate objects of these types.

The fundamental task of abstraction in an object-oriented design is to identify classes and objects in the problem domain and then to identify attributes and methods.

* Classes combine data and methods.
* A class defines a data type.
* Advantages: Classes correspond to concepts in the problem domain.
* Classes reduce complexity by increasing *coherence*.

**Class = data + methods**.

Everything (data and methods) in Java is contained in classes. So far you've been using classes to hold methods (main and perhaps a few other static methods). These methods use local variables, which are completely private to the method, and which disappear when the method returns.

## Advantage of thinking a solution in terms of “*Classes*”

### Classes used to model problem.

One advantage of encapsulating data and methods in a class is to make programming objects that reflect "objects" in the problem domain. If your problem deals with**sales,** then you'll very likely have classes called *Order* and *Produc*t.

### Classes used to reduce complexity.

Another advantage of classes is reducing *complexity*. Complexity limits the size and reliability of programs. Complexity is reduced by **increasing *cohesion*** (**putting things together that belong together**) and **reducing *coupling*** (interconnections).

## Modeling real world problems:

The main task of OOP is to modeling the real-world problem using **classes** and **objects**. Classes and objects are models that used *to represent the real-world problem that supposed to be solved*.A model is a representation of simple important aspects of the real world.Model used in software development like, representation of input, objects, object interaction and classes.Abstraction is the main source for modeling. These models represent classes and objects. Thus, classes and objects are a representation of reality, abstracted from the complex real world.

**Finding classes and objects**

* The first step in finding classes and objects is studying the problem domain.
* Classes and objects should initially emerge from the problem domain itself.
* You should prepare to spend a good deal of time in investigation.
* Observe first hand documents by working with the end users.
* Take the specialist in that area and problem domain experts.
* Look for previous results on similar problem domain.
* Construct a **prototype** –simple skeleton of real system,using classes, objects and including expected interactions between.

After this is done, the classes and objects can be used to

1. Identify the potential/necessary attribute or fields
2. Identify methods required to access its data.

|  |
| --- |
| Class name |
| List of Attributes |
| List of Methods |

**Examples of systems that need OO solution**

* **Example 1:**Consider ***Cost Sharing System***, use abstraction feature and list possible classes, attributes, and methods
* **Example 2:**Consider ***Dormitory System***, use abstraction technique and list possible classes, attributes, and methods. Do the same for Car Registration System.

**Class is similar to database table**. If you are familiar with databases, a *class* is very similar to a *table definition*.

***A class has two main sections: Private and public***

* A **private** part canonly be accessible by member methods(by allmethods defined inside the class).
* A **public** part can be used by other methods and classes outside the given class.

## Examples of defining Classes and Objects in java

**Example 1:**it shows how to define private and public sections of a class

public class Myclass{

Private int x;// accessed only inside Myclass

Public float y; //can be accessed outside Myclass

Private method1( ){} //can’t access data outside this class

Public method2( ) { } //can access data from other class

}

**Example 2:**Here is the class again that represents information about a student.

// Purpose: Information about a student.

public class Student {

public String fName; // First name

public String lName; // Last name

private int id; // Student id

}

## 2.4 Use *new* operator to create a new object

A class defines what fields an *object* will have when it's created. When a new Student object is created, a block of memory is allocated, which is big enough to hold these three fields -- as well as some extra overhead that all objects have.

***Student stud;***

***stud = new Student();*** // Create Student object with **new** operator.

Or we can write in one line in the following way:

**Student stud=new Student ( );**

To create a new object, write new followed by the name of the class, followed by parentheses. Later we'll see that we can specify arguments in the parentheses when creating new objects.

### Access public fields with dot notation

All public fields can be referenced using **dot notation** (later we'll see better ways to access and set fields). To reference a field, write the object name, then a dot, then the field name or method name.

***Student stud;***// Declare a variable to hold a Student object.

**stud = new Student();** // Create a new Student object with default values.

**stud.firstName = "Rishan";** // Set values of the fields.

**stud.lastName = "Mezegeb";**

**stud.id = 1234;**

***A class definition is a template for creating objects.*** A class defines which fields an object has in it. You need to use new to create a new object from the class by allocating memory and assigning a default value to each field. A little later you will learn how to write a ***constructor*** to control the initialization.

## 2.5 Constructor

* **Constructor can be Default and Parameterized**

A constructor is a special initialization method that is called automatically whenever an instance of a class is created. The constructor member method has, by definition, the same name as the corresponding class. The constructor has no return value specification. The Java run-time system makes sure that the constructor of a class is called when instantiating an object.

Constructor Method

* Automatically invoked when a class instance is created using the *new* operator
  + Has same name as the class
  + Has no return type
  + Java creates a default constructor if no constructor has been explicitly written for a class

Constructors are used to *initialize the data members* of the class. If we have data members that must be initialized, we must have a constructor

### Default (Non-Parameterized) constructors

Used to initialize objects using default values, let us look at the following example.

public class Employee{

String empName;

String address;

int id;

public Employee( ) // Default constructor

{ empName = “ ”;

address = “ ”;

int=0;

}

}//end class

The constructor Employee () is a *default* or a *non-parameterized* constructor, because it doesn’t use parameters. Default parameter is used to initialize the object with default value

### Parameterized constructors

The parameterized constructors can accept values into the constructor that has different parameters in the constructor. The value would be transferred from the **main ( ) method to the constructor either with direct values or through variables.**

Parameterized Constructor has parameter list to receive arguments for populating the instance attributes.

When you create a new instance (a new object) of a class using the new keyword, a *constructor* for that class is called. Constructors are used to initialize the instance variables (fields) of an object. Constructors are similar to methods, but with some important differences.

* **Constructor name is class name** i.e., they must have the *same name as the class.*
* **Default constructor**. If you don't define a constructor for a class, a *default (parameter less) constructor* is automatically created by the compiler. The default constructor calls the default parent constructor (super ()) and initializes all instance variables to default value (zero for numeric types, null for object references, and false for Booleans).

**Differences between methods and constructors**

* There is ***no return type*** given in a constructor signature (header). The value is this object itself so there is no need to indicate a return value.
* There is *no return statement* in the body of the constructor.

There can be more than one constructor in a class, distinguished by their parameters. When the class is initialized, Java will call the right constructor with matching arguments and type.

**Example 1: //without constructors’**

public class Circle {

public int x, y; // centre of the circle

public int r; // radius of circle

//Methods to return circumference and area

public double circumference() {

return 2\*3.14\*r;

}

public double area() {

return 3.14 \* r \* r;

}

public static void main(String args[])

{

Circle aCircle; // creating reference

aCircle = new Circle(); // creating object

aCircle.x = 10; // assigning value to data field

aCircle.y = 20;

aCircle.r = 5;

double area = aCircle.area(); // invoking method

double circumf = aCircle.circumference();

System.out.println("Radius="+aCircle.r+" Area="+area);

System.out.println("Radius="+aCircle.r+" Circumference ="+circumf);

}}

**Example 2: // Write the above code using constructors**

public class Circle {

public int x, y; // centre of the circle

public int r; // radius of circle

Circle( ){

x=0; y=0; r=0;

}

Circle(int valx , int valy , int valr)

{

x=valx; y=valy ; r=valr;

}

//Methods to return circumference and area

public double circumference( ) {

return 2\*3.14\*r;

}

public double area( ) {

return 3.14 \* r \* r;

}

public static void main (String args[]){

Circle c1=new Circle();

System.out.println("The value of x is " + c1.x);

System.out.println("The value of yis "+ c1.y);

System.out.println("The value of r is " + c1.r);

Circle c2=new Circle(4,2,5);

System.out.println("The value of x is " + c2.x);

System.out.println("The value of y is " + c2.y);

System.out.println("The value of r is " + c2.r);

System.out.println("Area of c2 is" +c2.area( ));

System.out.println("Circumference of c2 is "+c2. circumference( ));

}

}

**Example 3: It contains two constructors - Default and Parameterized. This example is about a polygon, Cube**

|  |
| --- |
| public class Cube1 {  int length;  int width;  int height;  public int getVolume() {  return (length \*width \* height);  }  Cube1() {  length = 0;  width = 0;  height = 0;  }  Cube1(int l, int b, int h) {  length = l;  width = b;  height = h;  }  public static void main(String[] args) {  Cube1 cubeObj1, cubeObj2;  cubeObj1 = new Cube1();  cubeObj2 = new Cube1(10, 10, 10);  System.out.println("Volume of Cube is : " + cubeObj1.getVolume());  System.out.println("Volume of Cube is : " + cubeObj2.getVolume());  }} |

If such a class requires a default constructor, its implementation must be provided. Any attempt to call the default constructor will be a compile time error if an explicit default constructor is not provided in such a case.

***Summary***

* Constructor is a special method that gets invoked “automatically” at the time of object creation.
* Constructor is normally used for initializing objects with default values unless different values are supplied.
* Constructorhas the same name as the class name.
* Constructor cannot return values.
* A class can have more than one constructor as long as they have different signature (i.e., different input arguments syntax).