

**Department of Computer Science**

**Spring 2018**

# **CS1421 – Object Oriented Programming Lab**

# **Lab 4 & 5**

**Instructors:**

Safiyah Batool

Zain Noreen

**Lab 4 & 5**

**Classes, Objects, Methods and Constructors**

**Objective**

After completing this lab, the students should be able to

* Create methods
* Define classes and create objects
* Access objects via object reference variables
* Understand the usage of constructors
* Define private data fields with appropriate getters and setters ( Encapsulation)

**Introduction**

The class is a logical construct that defines the shape and nature of an object .As it is the prime unit of execution for object oriented programming in java , so any concept in java program must be encapsulated within the class.

In java, class is defined as a new data type. This data type is used to create objects of its type. Each object created from the class contains its own copy of the attributes define in the class.The attributes are also referred to as fields and represent the state of an object . The initialization of objects is done using constructors and the behavior of the objects is defined using methods.

**Declaring a class**

As class declaration should begin with the keyword class followed by the name of the class that is being declared.

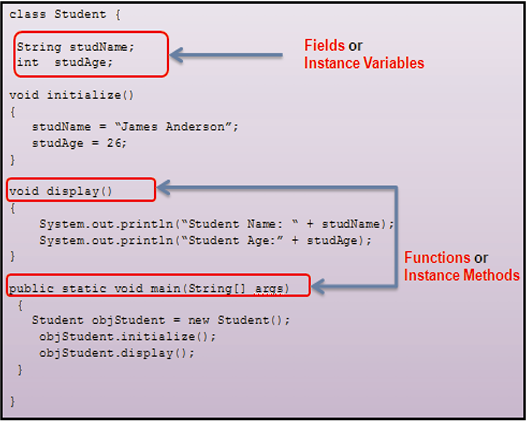
Besides this, following are some conventions to be followed while naming a class:

* Class name should be a noun.
* Class name can be in mixed case,with the first letter of each internal word capitalized.
* Class name should be simple, descriptive, and meaningful
* Class name cannot be java keywords
* Class name cannot begin with a digit . however they can begin with a dollar($) symbol or an underscore character:

Class <class\_name>{

//class body

}

The following figure shows the declaration of a sample class.

**Creating objects**

Objects are the actual instances of the class.

**Declaring and Creating an Object**:

An object is created using the new keyword. On encountering the new keyword JVM allocates memory for the object and returns a reference or memory address of the allocated object. The reference or memory address is then stored in a variable. This variable is also called as reference variable.

The syntax for creating an object is as follows



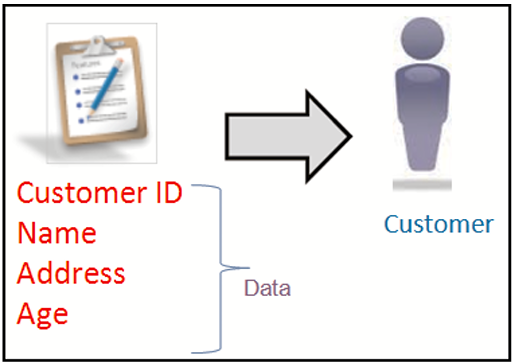
**Member of a class**

The member of a class are fields and methods. Fields define the state of an object created from the class and are referred to as instance variable the methods are used to implement the behavior of the objects and are referred as instance methods.

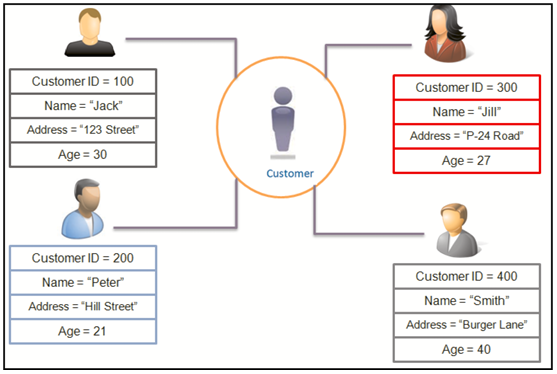
**Instance Variables**

The fields or variables defined within a class are called instance variables .Instance variables are used to stored data in them. They are called instance variables because each instance of the class that is objects of that class will have its own copy of the instance variable.This means each object of the class will contain instance variable during creation.

Consider a scenario where the customer class represent the details of customers holding accounts in a bank . In this scenario a typical question that can be asked is ‘What are the different data that are required to identify a customer in a banking domain and represent it as a single object?

The following shows a customer object with its data requirement.

As shown in figure the identified data requirements for a bank customer includes : CustomerID, Name, Address, and Age. To map these data requirements in a customer class, instance variables are declared. Each instances created from the customer class will have its own copy of the instances variables

Shows figure various instances of the class with its own copy of instance variables

As shown in above figure each instance of the class has its own instance variables initialized with unique data. Any changes made to the instance variables of one object will not affect the instance variables of the another object

The syntax to declare an instance variable within a class is as follows:

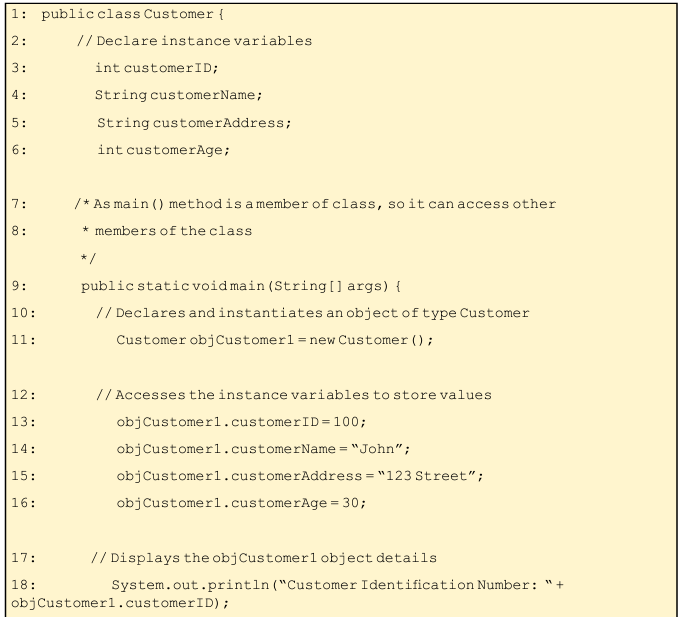
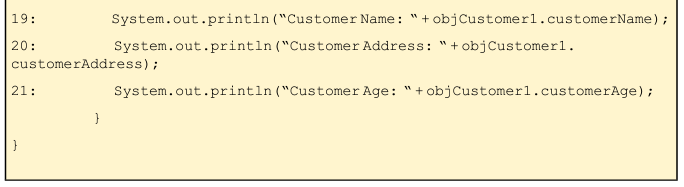


Where

access\_modifier is an optional keyword specifying the access level of an instance variable. It could be private, protected, and public.

data\_type specifies the data type of the variable. It can be of primitive types such as int, float, boolean, and so on. Also it can be of reference types, such as strings, arrays, or objects.

instanceVariableNamespecifies the name of the variable.



**Instance Methods**

They are functions declared in a class. They implement the behavior of an object. They are used to perform operations on the instance variables. They can be accessed by instantiating an object of the class in which it is defined and then, invoking the method. Instance method can access instance variables and instance methods directly.

For example, the class Car can have a method a Brake()that represents the ‘Apply Brake’ action. To perform the action, the method Brake()will have to be invoked by an object of class Car.

Following conventions have to be followed while naming a method:

• Cannot be a Java keyword.

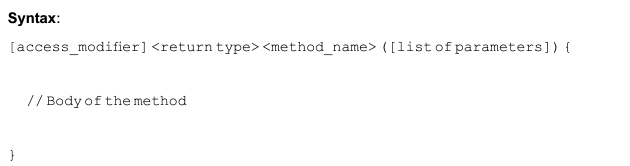
• Cannot contain spaces.

• Cannot begin with a digit.

• Can begin with a letter, underscore, or a ‘$’ symbol.

• Should be a verb in lowercase.

• Should be descriptive and meaningful.



where

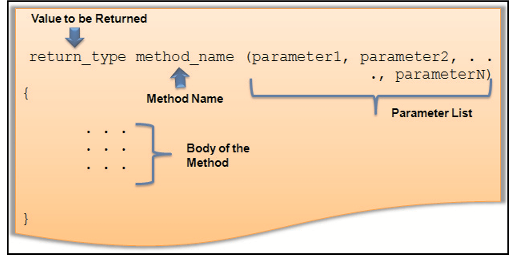
1. Modifiers — such as public, private, protected and default.

2. The return type — the data type of the value returned by the method, or void, if the method does not return a value.

3. The method name — the rules for field names apply to method names as well, but the convention is a little different.

4. The parameter list in parenthesis — a comma-delimited list of input parameters, preceded by their data types, enclosed by parentheses (). If there are no parameters, you must use empty parentheses.

5. The method body, enclosed between braces — the method's code, including the declaration of local variables.



Each instance of the class has its own instance variables, but the instance methods are shared by all the instances of the class during execution. As methods are part of class declaration, they can access the other class member, such as instance variables and method of the class.

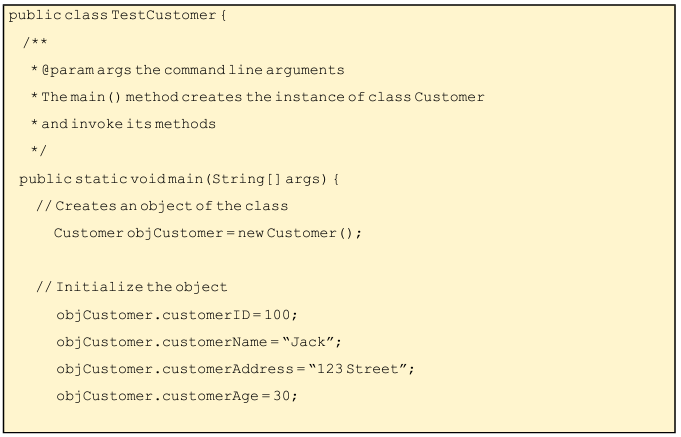
**Invoking Methods**

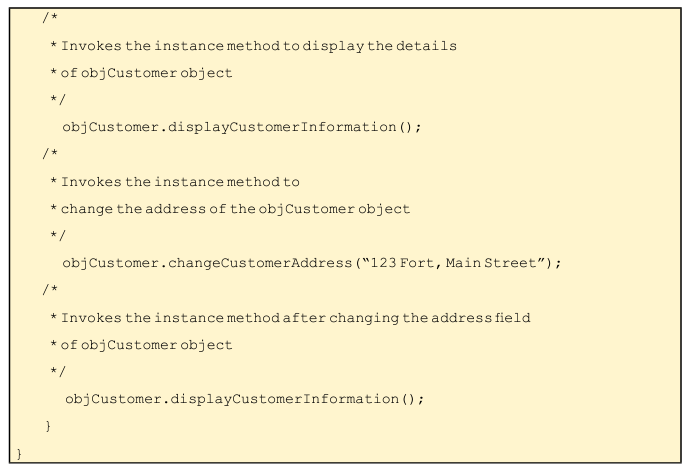
You can access a method of a class by creating an object of the class. To invoke a method, the object name is followed by the dot operator (.) and the method name. A method is always invoked from another method. The method which invokes a method is referred to as the calling method. The invoked method is referred to as the **called** **method**. After execution of all the statements within the code block of the invoked method, the control returns back to the **calling** **method**.

Most of the methods are invoked from the main() method of the class which is the entry point of the program execution.

When a program invokes a method, the program control gets transferred to the called methods. When the method is invoked, all the statements that are part of the method would be executed. When the JVM invokes a class method, it selects the method to invoke based on the type of object reference which is always known as compile time.

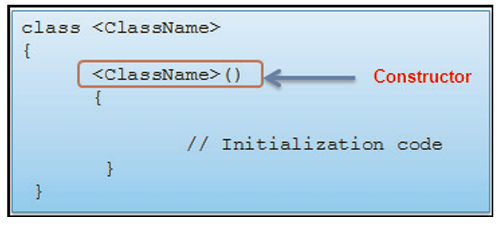
code snippet that demonstrates a class with main ()method which creates the instance of the class **Customer** and invokes the methods defined in the class



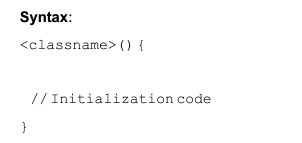


**Constructor**

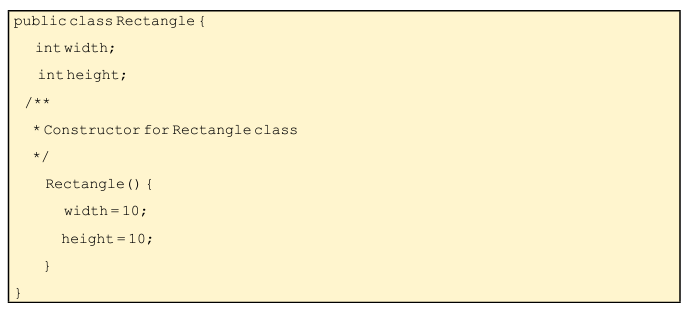
A class can contain multiple variables whose declarations and initialization becomes difficult to track if they are alone within different blocks. Likewise, they may be other startup operations that need to be performed in an application like opening a file and so forth. Java programming language allows objects to initialize themselves immediately upon their creation. The behavior is achieved by defining constructors in the class.

A constructor is a method having the same name as that of the class. Constructors initialize the variables of a class or perform startup operations only once when the object of the class is instantiate. They are automatically executed whenever an instance of a class is created before the new keyword completes.Also , constructor method donot have return types , but accepts parameters

Declaration of the constructor



Code snippet

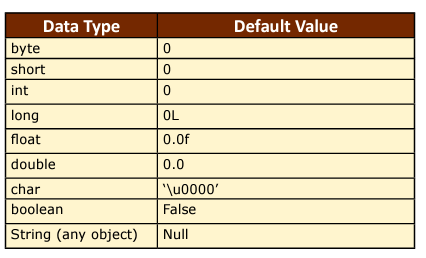


The code declares a method name Rectangle() which is a constructor. This method is invoked by JVM to initialize the two instance variables , width and height , when the object of type Rectangle is constructed. Also , the constructor does not have any parameters; hence it is called as no-argument constructor.

**Default Constructor:**

• Created for the classes where explicit constructors are not defined.

• Initializes the instance variables of the newly created object to their default values.

following table which lists the default values assigned to instance variables of the class depending on their data types

**Parameterized constructor**

The parameterized constructor contains a list of parameters that initializes instance variables of an object. The value for the parameters is passed during the object creation. This means each object will be initialized with different set of values.

**Encapsulation**

Encapsulation in Java is a mechanism of wrapping the data (variables) and code acting on the data (methods) together as a single unit. In encapsulation, the variables of a class will be hidden from other classes, and can be accessed only through the methods of their current class. Therefore, it is also known as data hiding.

To achieve encapsulation in Java

Declare the variables of a class as private.

Provide public setter and getter methods to modify and view the variables values.

Following is an example that demonstrates how to achieve Encapsulation in Java

public class EncapTest {

private String name;

private String idNum;

private int age;

public int getAge() {

return age;

}

public String getName() {

return name;

}

public String getIdNum() {

return idNum;

}

public void setAge( int newAge) {

age = newAge;

}

public void setName(String newName) {

name = newName;

}

public void setIdNum( String newId) {

idNum = newId;

}

}

The public setXyz() and getXyz() methods are the access points of the instance variables of the EncapTest class. Normally, these methods are referred as getters and setters. Therefore, any class that wants to access the variables should access them through these getters and setters.

The variables of the EncapTest class can be accessed using the following program

public class RunEncap {

public static void main(String args[]) {

EncapTest encap = new EncapTest();

encap.setName("James");

encap.setAge(20);

encap.setIdNum("12343ms");

System.out.print("Name : " + encap.getName() + " Age : " + encap.getAge());

}

}

**Practice Tasks**

**Note: Draw UML class diagram for all tasks.**

**Task 1**

A cell phone was created to provide basic functionality of Calling and Messaging. With the course of time, thousands of new features have been added and the count is still increasing but the basic features remain same. For any cell phone object, design a class that describes its basic characteristics and methods.

Remember, every cell phone has

**An IMEI code**

**A SIM card**

**A Processor**

**Internal Memory**

**isSingle SIM(true/false)**

and every cell phone can

**Dial a Number**

**Receive a Call**

**Send an SMS**

**Receive an SMS**

Create parameterized constructor and getters and setters for all the instance variables.

Create an application to test your class that has an object cellPhone1 whose IMEI code is: **IEDF34343435235,** accepts a nano SIM card, with SnapDragon as its processor, has 8 GB internal memory and is Single SIM. Call all the methods of cellPhone1.

**Task 2**

Create your version of **Date** class that includes three instance variables—a **month** (type int), a **day** (type int) and a **year** (type int). Provide a constructor that initializes the three instance variables and assumes that the values provided are correct. Provide a *set* and a *get* method for each instance variable. Provide a method displayDate that displays the month, day and year separated by forward slashes (/). Write a test app named DateTest that demonstrates class Date’s capabilities.

**Task 3**

While exercising, you can use a heart-rate monitor to see that your heart rate stays within a safe range suggested by your trainers and doctors. According to the American Heart Association (AHA), the formula for calculating your *maximum heart rate* in beats per minute is 220 minus your age in years.

Your *target heart rate* is a range that’s 50–85% of your maximum heart rate. [*Note:* These formulas are

estimates provided by the AHA. Maximum and target heart rates may vary based on the health, fitness

and gender of the individual. **Always consult a physician or qualified health-care professional before**

**beginning or modifying an exercise program.**] Create a class called **HeartRates**. The class attributes

should include the person’s first name, last name and date of birth (consisting of separate attributes for

the month, day and year of birth). Your class should have a constructor that receives this data as parameters.

For each attribute provide *set* and *get* methods. The class also should include a method that calculates and returns the person’s age (in years), a method that calculates and returns the person’s maximum heart rate and a method that calculates and returns the person’s target heart rate. Write a Java app that prompts for the person’s information, instantiates an object of class **HeartRates** and prints the information from that object—including the person’s first name, last name and date of birth—then calculates and prints the person’s age in (years), maximum heart rate and target-heart-rate range.

**Task 4**

A health-care issue that has been in the news lately is the computerization of health records. This possibility is being approached cautiously because of sensitive privacy and security concerns, among others.

Computerizing health records could make it easier for patients to share their health profiles and histories

among their various health-care professionals. This could improve the quality of health care, help avoid drug conflicts and erroneous drug prescriptions, reduce costs and, in emergencies, could save lives. In this exercise, you’ll design a “starter” **HealthProfile** class for a person. The class attributes should include the person’s first name, last name, gender, date of birth (consisting of separate attributes for the month, day and year of birth), height (in inches) and weight (in pounds). Your class should have a constructor that receives this data. For each attribute, provide *set* and *get* methods.

The class also should include methods that calculate and return the user’s age in years, maximum heart rate and target-heart-rate range (see Task 3), and body mass index (BMI; see Lab 2 Practice Task 1). Write a Java app that prompts for the person’s information, instantiates an object of class **HealthProfile** for that person and prints the information from that object—including the person’s first name, last name, gender, date of birth, height and weight—then calculates and prints the person’s age in years, BMI, maximum heart rate and target-heart-rate range. It should also display the BMI values chart

**Task 5**

Hitachi ltd. has hired you to program the television sets they manufacture. Each TV has a **channel, volumeLevel**, and a **state**(on/off. The state should be true if television is on and false if it is off). In the same way, every TV set can perform a few functions that are **turnOn()** to turn the television on, **turnoff()** to switch it off, **setChannel(newChannel)** that provides a specific channel to set, **setVolume(newVolumeLevel)** to set a specified volume, **channelUp()** to go up by one channel, **channelDown()** to go one channel down, **volumeUp()** in order to increase volume level by one and **volumeDown()** to decrease volume level by one. Design a class **TV** with appropriate instance variables, methods, constructors, getters and setters according to the information given above. Write a Java app to test your TV class.

**Task 6**

Pakistan Stock Exchange provides a reliable, orderly, liquid and efficient digitized market place where investors can buy and sell listed companies’ common stocks and other securities. They have asked you to design a class for their Stocks. Create a class **Stock**

* A **string** data field named **symbol** for the stock’s symbol.
* A **string** data field named **name** for the stock’s name.
* A **double** data field named **previousClosingPrice** that stores the stock price for the previous day.
* A **double** data field named **currentPrice** that stores the stock price for the current time.
* A constructor that creates a stock with the specified symbol and name.
* A method named **getChangePercent()** that returns the percentage changed from **previousClosingPrice** to **currentPrice**.

Draw the UML diagram for the class and then implement the class. Write a test program that creates a **Stock** object with the stock symbol **ORCL**, the name **Oracle Corporation**, and the previous closing price of **34.5**. Set a new current price to **34.35** and display the price-change percentage.