Lecture 4
Java Basics cont..

Spring, 2020

Content

- Methods
- Class
- Object

Methods

- Method is defined as the set of code which performs a specific task and returns any data if specified.
- It is executed, whenever it is invoked. It is like function in C.
- Method Declaration
- Syntax:
- Modifier ReturnType MethodName([Parameter list])
- {
- //body of the method
- [return value;]
- }

Methods

- Example
- **public int** add(**int** a, **int** b) {
-
- **int** c = a+b;
-
- **return** c;
-
- }

Methods

- Modifier:
- They are used to describe the level of access for the method. There are also static and final modifiers which would be explained later. In the above method **public** is the modifier. Two other modifiers commonly used are **private** and **protected**.

- Return Type:
- They are used to specify the type of the data to be returned if required. If no data is to be returned, then **void** keyword can be used.

Methods

- Method Name:
- Represents the name of the method, useful for invoking the method. In above method, **add** is the method's name.
-

  Argument List:
- Represents the list of parameters (along with the type) which is passed to the method. Parameters are used for the processing, inside the method. Parameters are optional. Two integer parameters **a and b** are passed in the above method for adding two numbers.

Methods

- Method Body:
- Method body is enclosed within a pair of curly braces. They tell the method what to perform. In the add() method, two integer values are added, the result is of addition is returned.
- Invoking a Method:
- A method can be invoked any number of times, by using the object of the class in which the method is declared followed by the dot operator.
- If a invoking method is present in the same class as the method to be invoked, object reference or dot operator is not required. Mostly, internal calculations or something that is needed only within that particular class will be defined and invoked without object reference like this.

Method

```
class Addition {
   public int add(int a, int b) {
   return (a+b);
   }
}

class MainClass {
     public static void main(String args[]) {
       Addition obj = new Addition();
       int r = obj.add(4,8);       //add method invoked
       System.out.println("Result of (4+8):"+r);
       System.out.println("Result of (5+6):"+obj.add(5,6));   //add method invoked
     }
}
```

Class & Object

- In object oriented programming languages, classes and objects are the basic elements of a program. Messages are passed only through objects.
- Class:
- A class is a blueprint which defines the properties and behaviour of the object which it supports
- Object: (Instance of a class)
- An object is an entity, which possess some properties and behavior. Example – Person is an object with properties – Name, Gender, etc and with behavior – sleeping, eating etc.

Class & Object

- Class:
- Properties and behavior of class are referred to as data members (fields/variables) and methods respectively.
- Syntax:
- class class_name
  {
  //fields//methods
  }

Class & Object

```
class Employee{

int id;

String name, department;

float salary;

void display(){

System.out.println(“Employee ID:”+id);

System.out.println(“Employee Name:”+name);

System.out.println(“Employee
Department:”+department);

}
```

Class & Object

- Object is an instance of class. In other words, they are variables of the type class.
- Every Object has:
  - State – State represents the property of the object.
  - Behavior – Behavior represents the functionality of the object.
  - Identity – It is an unique id defined by the JVM (internally) to refer the object.

- Creation of Object:
- Once a class is created, objects of that class are created by using it’s class name.
- Syntax:
  - class_name object_name = new constructor();

Class & Object

- Example:
- Employee e1 = new Employee();
  - The object ‘e1’ is created and instantiated using the new keyword.
  - The new keyword is followed by a call to the constructor which initializes the object.
  - The name of the constructor is same as that of the class name. Constructors would be discussed later.
- Accessing fields and methods of a class:
- The fields and methods of a class can be accessed by using the ‘dot’ operator.
- Syntax:
  - object_name.field;
  - object_name.method();

Class & Object

```
class Employee{

int id;

String name, department;

float salary;

void display(){

System.out.println("Employee ID:"+id);

System.out.println("Employee Name:"+name);

System.out.println("Employee Department:"+department);

}

public static void main(String args[]) {

Employee e1 = new Employee();                    //object
creation

e1.id = 101;                                     //accessing
fields
e1.name = "Anu";
e1.department = "Finance";
e1.display();                                    //accessing
methods
}
}
```

Class & Object

```
public class Person{
String name;
int age;
public void displayMsg(){
  System.out.println("Im " + name );
  System.out.println("Im " + age + " years old");
  }
}
```

```
public class UsePerson{
public static void main(String[] args) {
  // Create an object of type Person and referred by a variable personOne.
  Person personOne= new Person();
  personOne.name = "Dara";
  personOne.age = 20;
  personOne.displayMsg();
  }
}
```

Output:

Im Dara
Im 20 years old

Class & Object

- We have two classes in a same source file. Then, which name should we use as a file name?
- You have to use the name of the class containing main() method. so we save the above source file as UsePerson.java.
- The main() method:
- A Java application can have any number of classes but atleast one class should contain a main() method. During runtime, the JVM will search for the main() method and start executing the statements in main() method.
  - Syntax:
  - public static void main(String [] args)
    {
    // statements
    }

Class & Object

- The new operator:
- Using Person class, you can create any number of objects of type Person using new() operator.
- Syntax:
  - classname objectname = new classname();
  - To create an object for Person class, we use
  - Person personOne= new Person();
  - new Person() will create a object of type Person.
    personOne is a variable name referring to the Person object.

Class & Object

- The dot operator
- The variables and methods of Person class is available to all the objects of Person class. Those variables and method are accessed by using dot operator.
- The name variable of personOne object is specified by personOne.name
  - personOne.name = “Dara”;
- similarly, the age variable of personOne object is specified by personOne.age
  - personOne.age= 20;

Class & Object

- Constructor:
- A constructor is a special method in a class, used to initialize the object’s fields.
- The name of the constructor is same as that of the class name.
- Whenever a new object of its class is created, a constructor is invoked.
- Every class has a constructor. Even if it is not defined, compiler would create a one implicitly.
- Constructors do not have return type.

Constructor

```
class Box {
   double height, width;
   Box() {                          //constructor
    height = 5;
    width = 5;
   }
}

class BoxExample {
  public static void main(String args[])
   Box b = new Box();               //invokes the constructor

  }
}
```

Constructor

- If no constructor is defined, the default constructor would initialize the instance variables to their default values (0 for numeric values, false for boolean type and null for reference types).
- Parameterized Constructor:
- We can also pass parameters to the constructors as we pass in methods.

Constructor

```
class Box {
   double height, width;
   Box(double h, double w) {                    //parameterized constructor
    height = h;
    width = w;
   }
}

class BoxExample {
  public static void main(String args[])
   Box b = new Box(3.0,5.0);                    //invokes the constructor
  }
}
```