National University of Computer & Emerging Sciences,
Karachi
Computer Science Department
Spring 2023, Lab Manual – 04

Course Code: CL-1004	Course : Object Oriented Programming Lab
Instructor(s) :	Abeer Gauher, Hajra Ahmed, Shafique

LAB - 4

Constructors and Garbage Collection

Constructor

In Java, a constructor is a block of codes similar to the method. It is called when an instance of the class is created. At the time of calling constructor, memory for the object is allocated in the memory.

It is a special type of method which is used to initialize the object.

Every time an object is created using the *new* keyword, at least one constructor is called.

Example: Calling constructor for creating Box class object

```java
Box ob = new Box();
```

Types of Constructors

There are two types of constructors in Java: default (no parameter) constructor, and parameterized constructor.

Default Constructor

The default constructor is used to provide the default values to the object like 0, null, etc., depending on the type.

- Example of Default (no parameter) Constructor

```java
public class Box {
    int width, height;

    // Default constructor
    public Box()
    {
        width = 1;
        height = 1;
    }
}
```

Parameterized Constructor

A constructor which has a specific number of parameters is called a parameterized constructor.
The parameterized constructor is used to provide different values to distinct objects. However, you can provide the same values also.

- [Example of Parameterized Constructor](#)

```java
public class Box {
    int width, height;

    // Parameterized constructor
    public Box(int w, int h)
    {
        width = w;
        height = h;
    }
}
```

Rules for Constructors

1. Constructor name must be the same as its class name
2. A Constructor must have no explicit return type
3. If there is no constructor in a class, compiler automatically creates a default constructor.

[Complete Example](#)

```java
public class Box {
    int width, height;

    // Default constructor
    public Box()
    {
        width = 1;
        height = 1;
    }

    // Parameterized constructor
    public Box(int w, int h)
    {
        width = w;
        height = h;
    }

    void print()
    {
        System.out.print("Width of Box : " + width);
        System.out.println(" , Height of Box : " + height);
    }
}

// Using these constructors from main class:
```

```java
public class MainClass {
    public static void main(String[] args)
    {
        Box b1 = new Box();
        b1.print();

        Box b2 = new Box( w: 2,  h: 5);
        b2.print();
    }
}
```

Copy Constructor

We can use one object to initialize another object in Java by passing the object as parameter to a constructor.

Example:

```java
class Box {
    int width, height;
    // Parameterized constructor
    public Box(int w, int h) {
        width = w;
        height = h;
    }
    // Copy constructor
    public Box(Box ob) {
        width = ob.width;
        height = ob.height;
    }
    void print() {
        System.out.println("Width : " + width);
        System.out.println("Height : " + height);
        System.out.println("\n");
    }
}
```

```java
class MainClass {
    public static void main(String[] args) {
        Box b1 = new Box(2, 4);
        Box b2 = b1;

        System.out.println("Box 1 :- ");
        b1.print();
        System.out.println("Box 2:- ");
        b2.print();
    }
}
```

Output:

```
Box 1 :-
Width : 2
Height : 4


Box 2:-
Width : 2
Height : 4
```

Object Passing using Function

We can also pass objects as parameters to functions in Java just like other primitive type arguments.

Example:

```java
class Square {
    int dim;
    public Square(int d) {
        dim = d;
    }
    public void print() {
        System.out.println("Sides: " + dim * dim);
    }
}
```

```
class Test {
    static void printer(Square o)
    {
        o.print();
    }

    public static void main(String[] args) {
        Square sq = new Square(5);
        printer(sq);
    }
}
```

Output:

```
Sides: 25
```

Garbage Collection

In Java, when we create an object of the class it occupies some space in the memory (heap). If we do not delete these objects, it remains in the memory and occupies unnecessary space that is not upright from the aspect of programming. To resolve this problem, we use the **Destructor**.

Unlike C++, which has explicit destructor, Java uses **finalize( )** method for this purpose. Since it cannot be predicted when the Java garbage collection occurs.

The finalize( ) Method

It is difficult for the programmer to forcefully execute the garbage collector to destroy the object. But Java provides an alternative way to do the same. The Java Object class provides the finalize() method that works the same as the destructor.

- It is a protected method of Object class
- It can be called only once.
- We need to call the finalize() method explicitly if we want to override the method.
- The gc() is a method of JVM executed by the Garbage Collector. It invokes when the heap memory is full and requires more memory for new arriving objects.

Example

```
public class DestructorExample
{
    public static void main(String[] args)
    {
        DestructorExample de = new DestructorExample ();
        de.finalize();
        de = null;
        System.gc();
        System.out.println("Inside the main() method");
    }
    protected void finalize()
    {
        System.out.println("Object is destroyed");
    }
}
```

Output:

```
Object is destroyed
Inside the main() method
Object is destroyed
```

Lab Tasks

1. Modify the "**Box**" class given in example of this manual and implement the constructors as per the following requirements:
    a) A constructor that receives both height and width as parameter to create new **Box** object only if width and height are both positive values
    b) A constructor that receives only height as parameter and takes width as input from the user
    c) A constructor that receives no parameter and takes both width and height as user input

2. Create a class that contains a String variable named language.
    1) Create a default constructor that sets the language to "JAVA".
    2) Create a parameterized constructor that sets this variable.
    3) Create a function that contains a print statement that displays the languages.
    4)   In the main function create 2 objects. One with the default constructor and one with the parameterized constructor with the value "C".
    5) Display the result.

3. Write a class called Holiday. This class has three instance variables: name, Day & month.
    a) Write a constructor for the class Holiday, which takes a String representing the name, an int representing the day, and a String representing the month as its arguments, and sets the class variables to these values.
    b) Write a method inSameMonth, which compares two days of the class Holiday, and returns the Boolean value true if they have the same month, and false if they do not.
    c) Create a Holiday instance with the name "Independence Day", with the day "14", and with the month "August".

3. Implement a copy constructor for Holiday class and demonstrate its use in the main function.

4. Create a class called **Tank** that can be filled and emptied, and has a termination condition that it must be empty when the object is cleaned up. Write a **finalize**( ) that verifies this termination condition.
Hint: You can create an int variable capacity that represents the water level of **Tank**.

5. Create a class named as "GarbageCollection":
    *   Contains a constructor that prints "Object created".
    *   A finalize( ) method that prints "Object deleted".
        Perform object deletion using the following methods:
    1. Null a reference: Create an object then set the reference to null.
    2. By assigning a reference to another object using assignment operator.
    3. By creating an anonymous object.