Object Oriented Analysis and Design using Java

UE19CS353

Prof. Mahitha G and Prof. Sindhu R Pai

Department of Computer Science and Engineering

Acknowledgements: Significant portions of the information in the slide sets presented through the course in the class, are extracted from the prescribed text books, information from the Internet. Since these are only intended for presentation for teaching within PESU, there was no explicit permission solicited. We would like to sincerely thank and acknowledge that the credit/rights remain with the original authors/creators only

PES UNIVERSITY ONLINE

UE19CS353: Object Oriented Analysis and Design using Java

Interface and Implementation: Role of Constructors and Destructors, Garbage Collector

Prof. Mahitha G and Prof. Sindhu R Pai

Department of Computer Science and Engineering

Agenda

- Constructor
- Types of constructors
- Demo of constructor usage
- Access Modifiers on constructors
- Destructor
- Garbage Collector
- finalize() method
- Few questions

Constructor

- **Initializes an object during creation** and performs any other start-up procedures required to create a fully formed object.
- Has **the same name as it's class and** has **no explicit return type.**
- All classes have constructors, whether you define one or not, because Java automatically provides a default constructor that initializes all member variables to default values. However, once you define your own constructor, the default constructor is no longer called.
- Each time a object is created using new operator, constructor is invoked to **assign initial values to the data members of the same class.**

Types of Constructors

- **Default constructor**

  Has no parameters.

  If we don't define a constructor for a class, then compiler creates a default constructor

  Provides default values to the objects like 0,null etc depending on the type

- **Parameterized constructor**

  A constructor with parameters.

  To initialize the fields of a object with given values

- **Coding examples**

Note: **No return value statements in a constructor but constructors return the current class instance.**

Demo of Constructor usage

```
class Box{

    float width;    float height;   float depth;
    // constructor
    Box(){ }

    Box(float a, float b,float c)
    {   width = a;  height = b; depth = c;  }
    Box(float a, float b)
    {   width = a;  height = b; }
}
```

```
Box b1 = new Box();
Box b2 = new Box(23,6,8);
Box b3 = new Box(23,6);
```

Access Modifiers on constructors

- If private, cannot create the instance of that class from outside the class.

```
class A{
    private A()  { }            //private constructor
    void msg(){System.out.println("Welcome to OOAD with java class");}
    }
public class Sample
{
 public static void main(String args[]){
   A obj=new A();              //Compile Time Error
 }
}
```

- By default, it is default.

- Can make it as public.

But not a good idea

Destructor

- Special method called automatically during the destruction of an object
- Role of destructor
  - Recovering the heap space allocated during the lifetime of an object
  - Closing file or database connections
  - Releasing network resources
  - Releasing resource locks
  - Other housekeeping tasks

Note: In place of destructor, Java provides the **Garbage Collector**.

Garbage Collector

- A program that runs on the JVM.
- Program running in the background that looks into all the objects in the memory and find out objects that are not referenced by any part of the program.
- All these unreferenced objects are deleted and space is reclaimed for allocation to other objects.
- **Automatic garbage collection - Reference counting, mark and sweep, stop and copy**
- The programmer has no need to manage memory, manually.
- When the garbage collector destroys the object, the JRE calls the finalize() method to close the connections such as database and network connection, etc..

finalize ( ) method

```
protected void finalize()
{       //resources to be close  }
```

- It is **not a destructor** but it provides extra security.
- Ensures the use of external resources like closing the file, etc. before shutting down the program.
- Call it by using the method itself or invoking the method **System.runFinalizersOnExit(true)**.

Points to think!

- Can we have two classes in the same file?
- Can we have two public classes in the same file?
- If two classes have same name in the file and one of them contains the main method, what should be the name of the file?
- Which access modifier is best suited for attributes of the class to have maximum security?
- If the attributes are private, can functions in that class access those attributes?
- If the attributes are private, can the instance of a class modify the value which is assigned to that particular attribute?
- Can you have two constructors with the same number and type of parameters?

THANK YOU

Mahitha G and Sindhu R Pai
Department of Computer Science and Engineering

mahithag@pes.edu

sindhurpai@pes.edu
+918277606459