JAC444 - Lecture 3

Object-Oriented Concepts

Segment 1 - Object Class

Objectives

Upon completion of this lecture, you should be able to:

- Explore the Object Class
- Apply Inheritance Concept in Java
- Utilize Polymorphism Concept in Java
- Experiment with Namespace in Java - Package

Object Class

In this segment you will be learning about:

- java.lang.Object Class
- Methods of the Object Class
- Develop Classes using Object Class
- Construct a Pattern for Designing Classes in Java

Object Class

- The class **`Object`** is the root of the Java class hierarchy
- It is defined in the package **`java.lang`**
- Every class has **`Object`** as a superclass

  One MUST understand all methods defined in the Object class, since every class developed in Java inherits all its methods.

Methods in Object Class

- There are 11 methods in Object class.

```
Object      clone()
boolean     equals(Object obj)
void        finalize()
Class<?>    getClass()
int         hashCode()
void        notify()
void        notifyAll()
String      toString()
void        wait()
void        wait(long timeout)
void        wait(long timeout, int nanos)
```

toString() Method

- The **`toString()`** method does not take any parameters and returns a string representation of the object.

- The string must contain the state (field values) of the object.

- Every class that you develop must override this method

equals(Object obj) Method

- The **`equals`** method takes an object of type `Object` as param and returns a `boolean` value.

```
boolean equals(Object obj)
```

- The method compares the current object with the object given as param

- It is generally necessary to override the **`hashCode`** method whenever **`equals`** method is overridden

hashCode() Method

- The **`hashCode`** method does not take any param and returns a `integer` value as *hash code value*.

  (a *hash code value* is a 32-bit signed integer which represents the data stored in the object - see implementation example in the sample provided)

```
int hashCode()
```

- If two objects are equal according to the **`equals(Object)`** method, then calling the **`hashCode()`** method on each of the two objects must produce the same integer result.

Class<?> getClass()

- The **`getClass()`** method does not take any param and returns the runtime class of this Object

```
public final Class<?> getClass()
```

Example of using **`getClass()`** method

```
byte[] bytes = new byte[10];
Class c = bytes.getClass();
String className = c.getName();
```

Instances of the class **`Class`** represent classes and interfaces in a running Java application

wait() and notify() Methods

- The methods **`wait()`** and all its overloaded methods with **`notify()`** and **`notifyAll()`** are used in multithreading

- More about them in the lecture about **`Thread`** in Java

- Since these methods are inherited in all classes any object could invoke them

Question

Read the following class:

```
public class Question {
    public static void main(String[] args) {
        Question obj = new Question();
        String s = obj.toString();
        System.out.println(s);
    }
}
```

Can we invoke method `toString` on an object of type `Question` ?

YES. Since Question class inherits from Object class all its methods.

What is it printed?

The string returned from the invocation of `toString` method in the `Object` class:

`Question@3cd1a2f1` (name of the class @ hashcode of the obj)