Java Inte

Monday, February 19, 2018 2:16 PM

Important Topics

1. **Class and Objects**
2. **Encapsulations and Abstraction**
3. **Abstract and Interface**
4. **Object Oriented Programing**
5. **Enums**
6. **Inheritance**
7. **File - read/write, serialization**
8. **Exception Handling**
9. **Collection and Generics**
10. **Inner Classes**
11. **Thread**
12. **New Concepts in Java 8**
13. **Design Patterns**
14. **Arrays**

Follow this style to write code

https://google.github.io/styleguide/javaguide.html

** **Class and Objects:-**

Class :
A template that describes the kinds of state and behavior that objects of its type support.

Object:
Object At runtime, when the Java Virtual Machine (JVM) encounters the new keyword, it will use the appropriate class to make an object which is an instance of that class. That object will have its own state, and access to all of
the behaviors defined by its class.

Singleton in java

- Singleton pattern restricts the instantiation of a class and ensures that only one instance of the class exists in the java virtual machine.
- The singleton class must provide a global access point to get the instance of the class.
- Singleton pattern is used for logging, drivers objects, caching and thread pool.
- Singleton design pattern is also used in other design patterns like Abstract Factory, Builder, Prototype, Facade etc.
- Singleton design pattern is used in core java classes also, for example `java.lang.Runtime`, `java.awt.Desktop`.

To implement Singleton pattern, we have different approaches but all of them have following common concepts.

- Private constructor to restrict instantiation of the class from other classes.
- Private static variable of the same class that is the only instance of the class.
- Public static method that returns the instance of the class, this is the global access point for outer world to get the instance of the singleton class.

Different approaches of Singleton pattern implementation and design concerns with the implementation.

1. Eager initialization
2. Static block initialization
3. Lazy Initialization
4. Thread Safe Singleton
5. Bill Pugh Singleton Implementation
6. Using Reflection to destroy Singleton Pattern
7. Enum Singleton
8. Serialization and Singleton

Eager initialization

In eager initialization, the instance of Singleton Class is created at the time of class loading, this is the easiest method to create a singleton class but it has a drawback that instance is created even though client application might not be using it.
If your singleton class is not using a lot of resources, this is the approach to use. But in most of the scenarios, Singleton classes are created for resources such as File System, Database connections etc and we should avoid the instantiation until unless client calls the `getInstance` method. Also this method doesn't provide any options for exception handling.

```
/**
 *
 * @author rkumar
 * Eager Initialization implementation of Singleton
 * Use if singleton class is not using lot of resource.
 *
 */
public class EarlyInitilizeSingltonDemo {

	// private variable
	private static final EarlyInitilizeSingltonDemo instance = new EarlyInitilizeSingltonDemo();

	// private constructor
	private EarlyInitilizeSingltonDemo() {
		// TODO Auto-generated constructor stub
	}

	// public method to get the instance of this class
	public static EarlyInitilizeSingltonDemo getInstance() {
		return instance;
	}
}
```

Static block initialization