Singleton

Michal Moravik, SD20w2

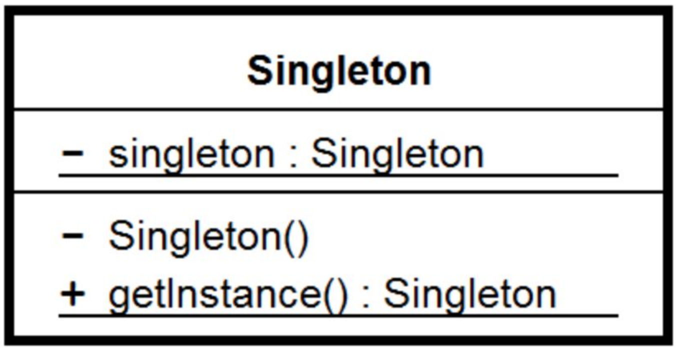
Name: Singleton Pattern

Category: Creational pattern

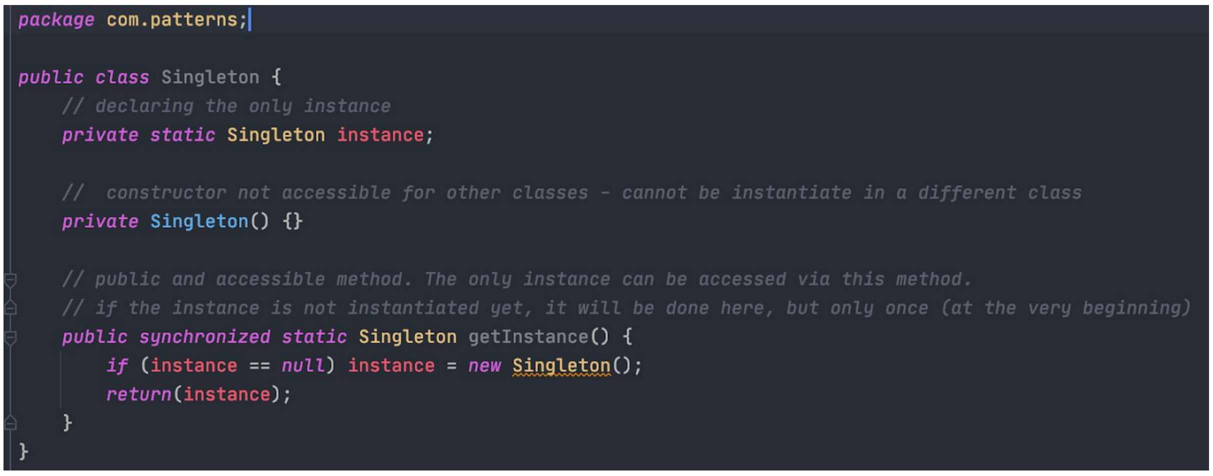
Intent: Keep only one instance of a class during the system's run time. It ensures that all requests go through that single instance. The action of creating multiple of them must be prohibited.

Motivation: Creating more than one instance of a class can sometimes trigger unwanted errors and/or other problems. This is especially true when one is creating a log file or establishing a database connection. The engineer needs to properly control that instance, synchronize it, close the connection, or otherwise, he/she would end up with multiple unwanted instances which could lead to problems like too many objects in the heap (overuse of resources), expensive connections opened but not closed, and more.

UML:



Implementation:



```
package com.patterns;

public class Singleton {
    // declaring the only instance
    private static Singleton instance;

    //  constructor not accessible for other classes - cannot be instantiate in a different class
    private Singleton() {}

    // public and accessible method. The only instance can be accessed via this method.
    // if the instance is not instantiated yet, it will be done here, but only once (at the very beginning)
    public synchronized static Singleton getInstance() {
        if (instance == null) instance = new Singleton();
        return(instance);
    }
}
```

Consequences:
Pros:
- Protection against overuse. There will be only one instance in the heap during the run time
- The instance is easily manageable - we can control and monitor the instance
- Can save resources, e.g. in database connections example mentioned above

Cons:
- We are making the instance global instead of passing it to functions which can be considered as code smells problem. The tracking of singletons is therefore hard.
- Functions, where singletons are called, are hard to test since the singletons are not passed via parameters
- The system is tightly coupled, we call the singleton instance everywhere we need to

Known uses:
- Database connection establishment and its manipulation
- Log files creation
- Other examples when we must have only one access point to that class

Related patterns:
- Null object Pattern
- Abstract Factory Pattern
- Factory Method Pattern