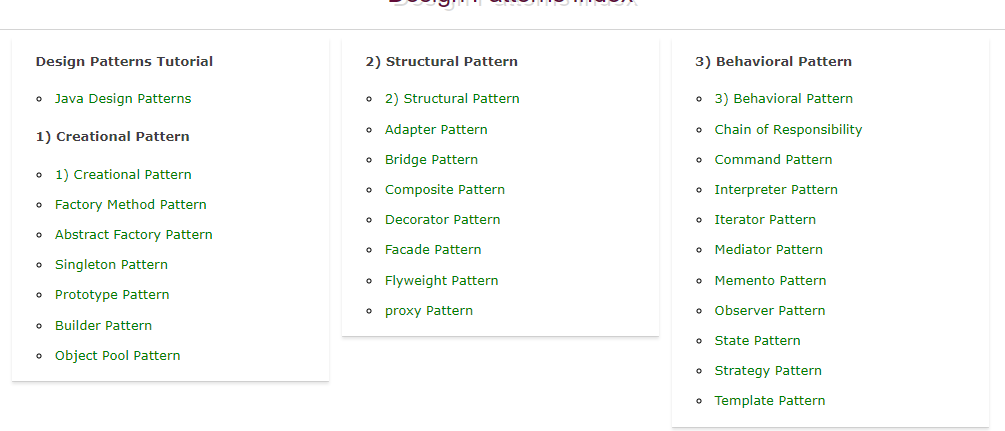
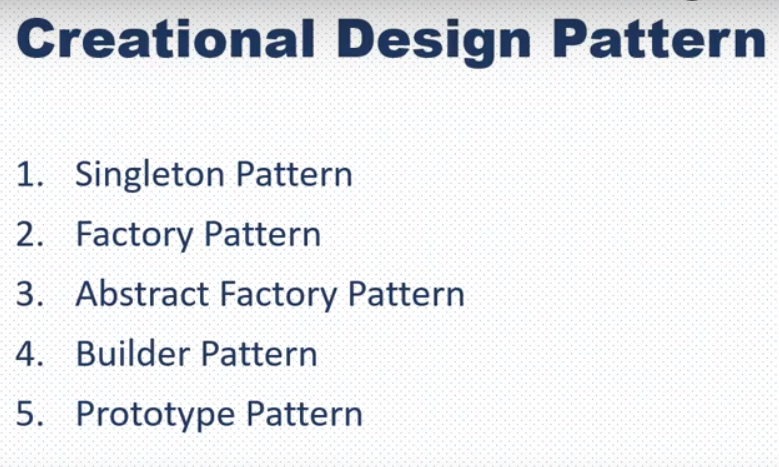
# Design patterns





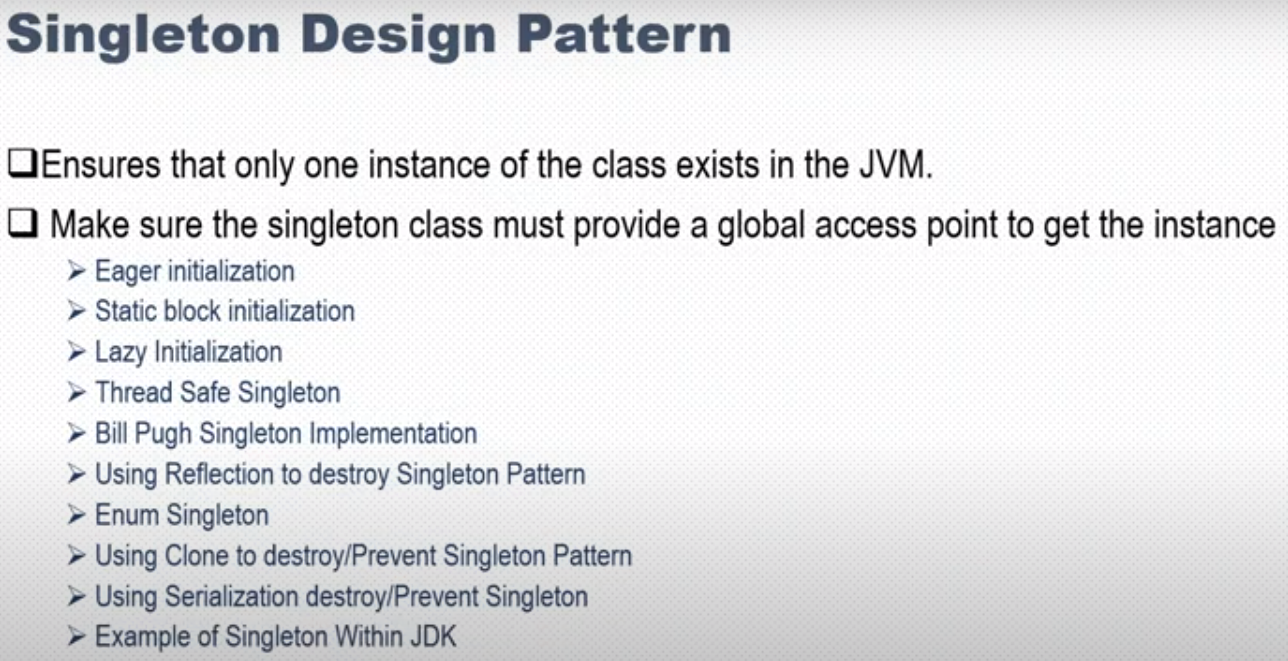
Creational Design Patterns: Creational design patterns are for creating an object in way different ways or loose coupled manner. Following are the 5 main Design patterns for creating the Object



<https://www.javatpoint.com/creational-design-patterns>

Singleton:

SIngleton design pattern can be achieved using below ways in screenshot



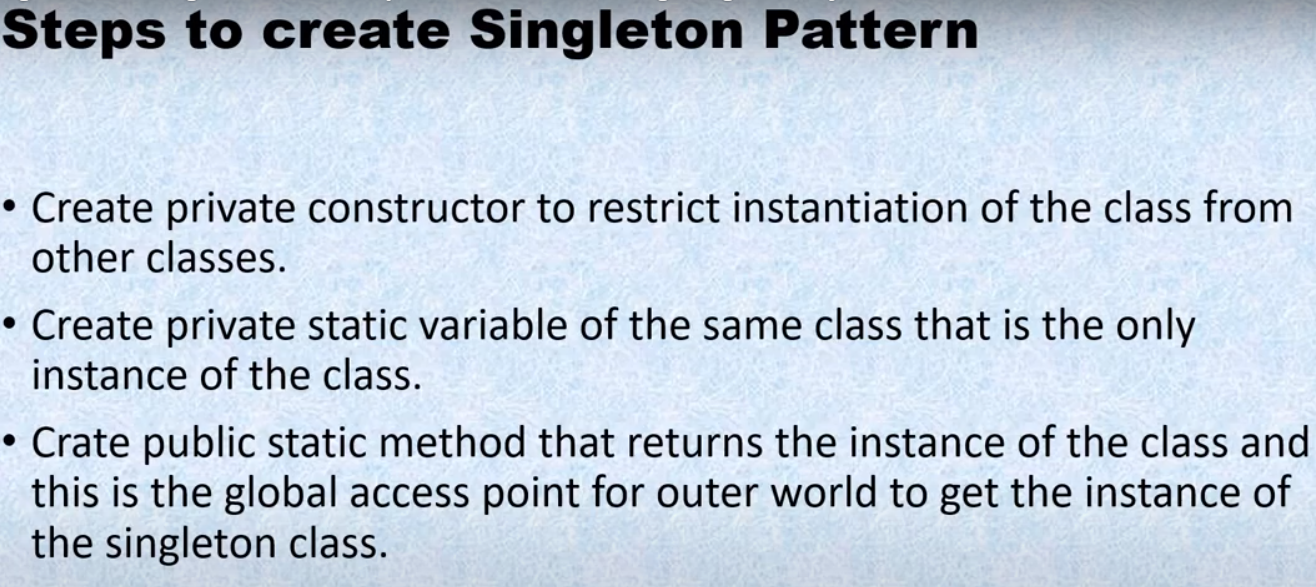
In Eager Initialization and static block initialization, the instances are created even before the request of creating the instance. Intances will be created in advance and can be used on every request.

Lazy Initialization, instance is created on request and send to user.

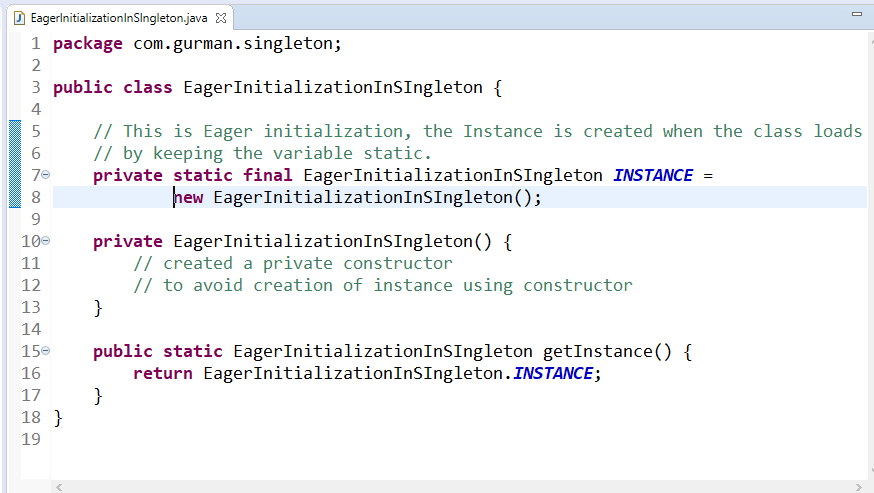
All three approaches are not thread safe.

Below we will start on the approach of SIngleton design pattern

Traditional way of creating it

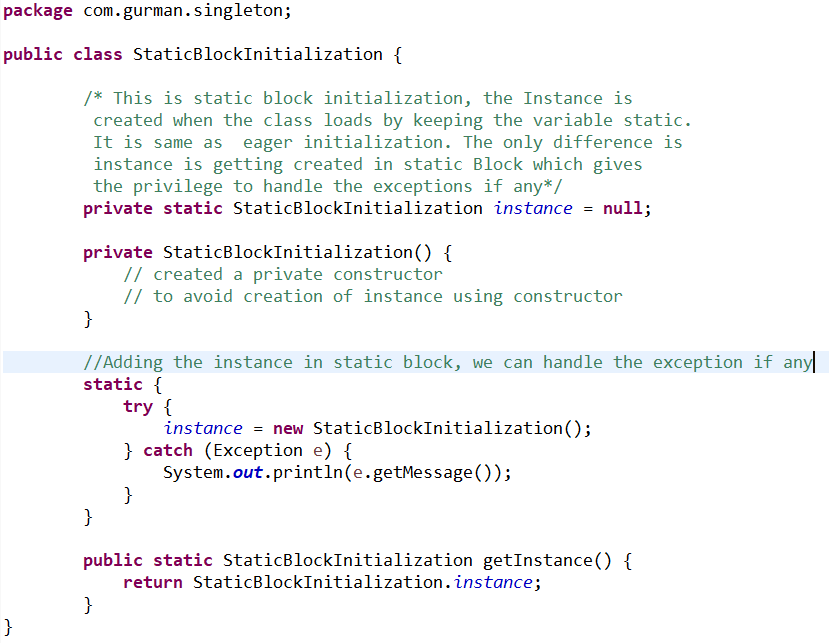


### Eager initialization



Drawback: the intance will be created even if no one request for the instance. Also using this we can't add exception handling.

### StaticBlock initialization:

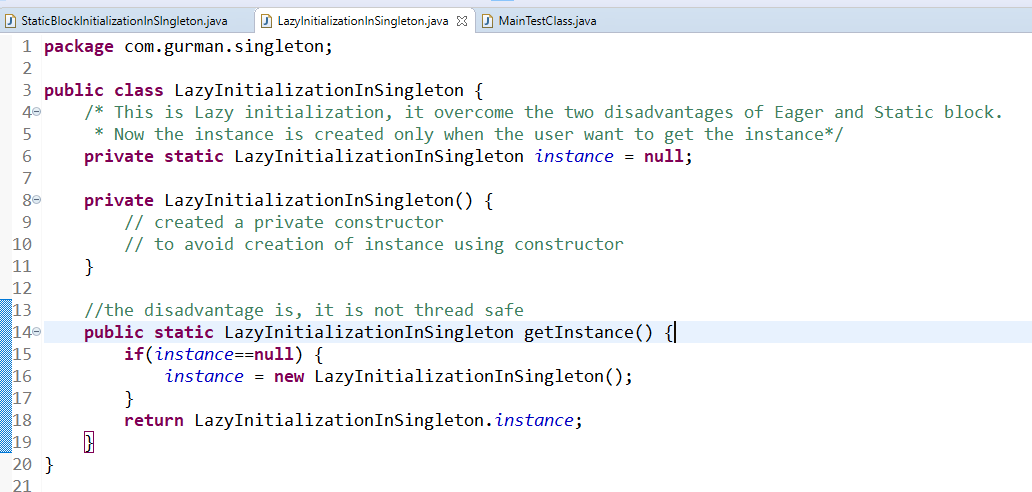


Disadvantage: the intance will be created even if no one request for the instance.

Advantage over Eager initialization: Exceptions can be handled

Lazy Initialization: instance is created when the user requests. It is created in the getinatance method after checking there is not other instance created yet. But suppose in multithreaded environment, two threads enters the if condition at same time. It will result in creating two intances. This is disadvantage of Lazy Initialization.

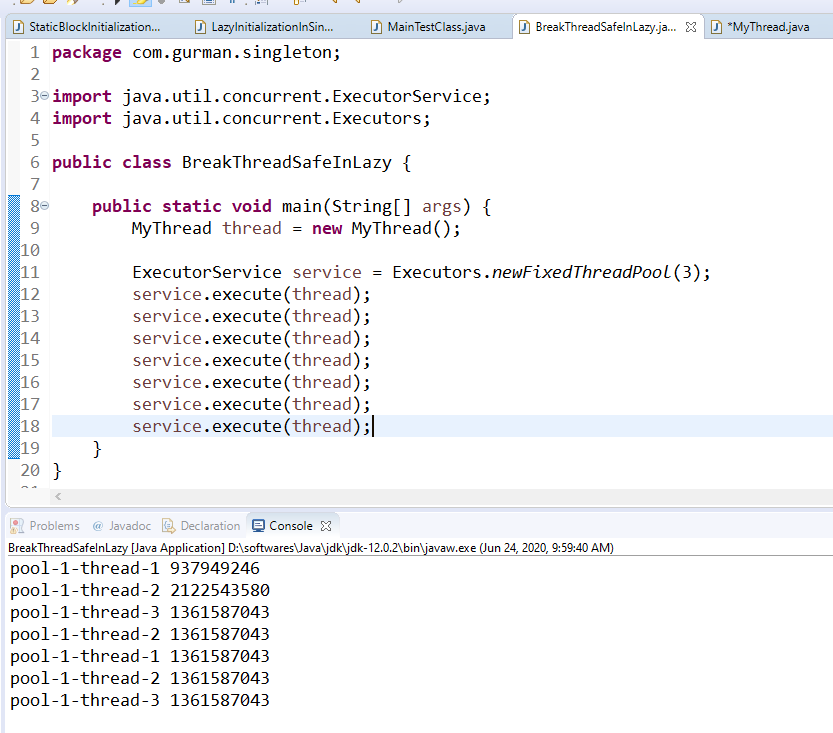
DIsadvantage: it is not thread safe.



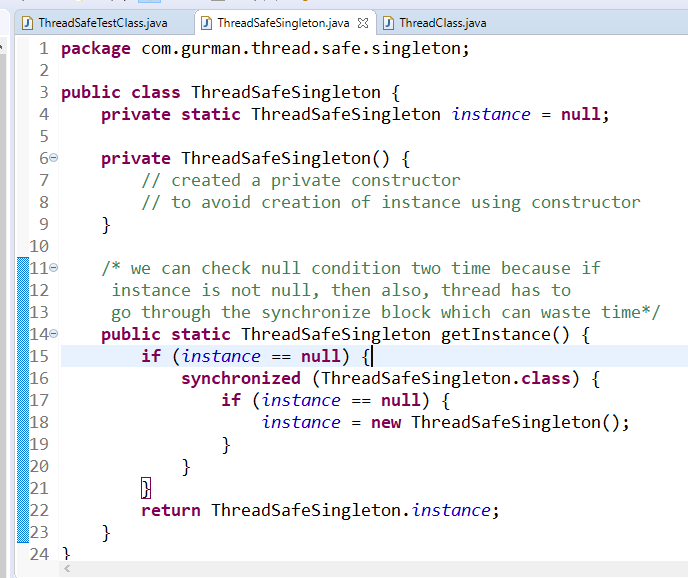
Check Thread safety in Lazy initialization

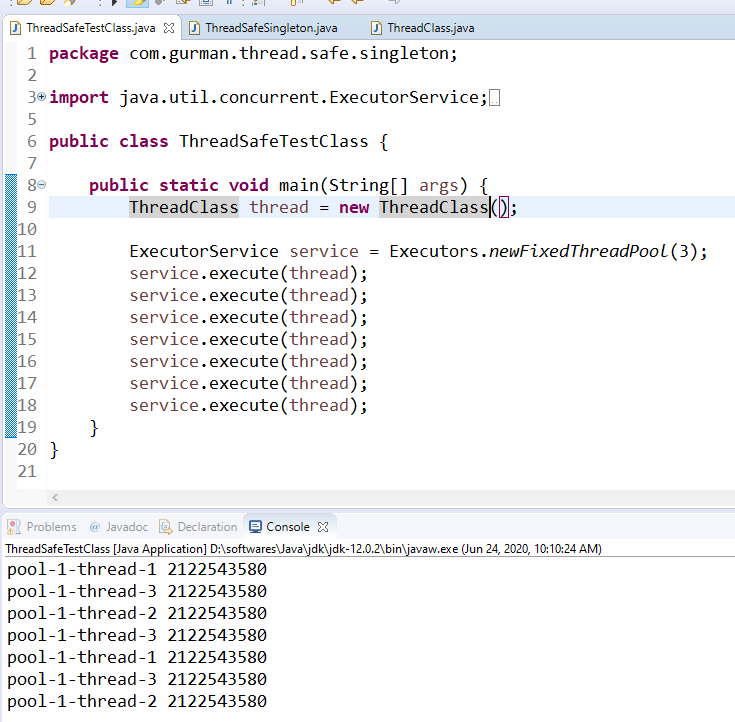


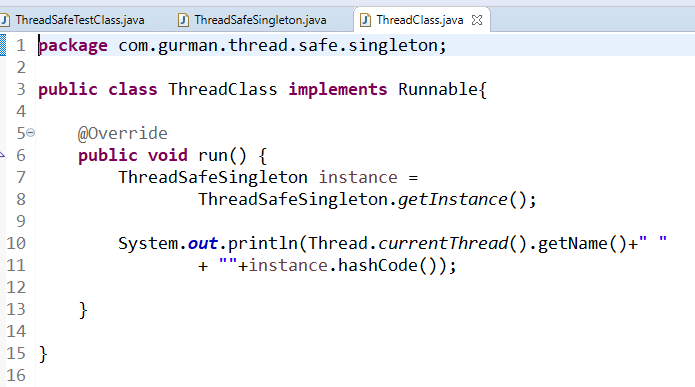
It has created mutiple instances for thread class



ThreadSafeSingleton: the code for creating the instance and checking the null condition can be added in synchronized block.

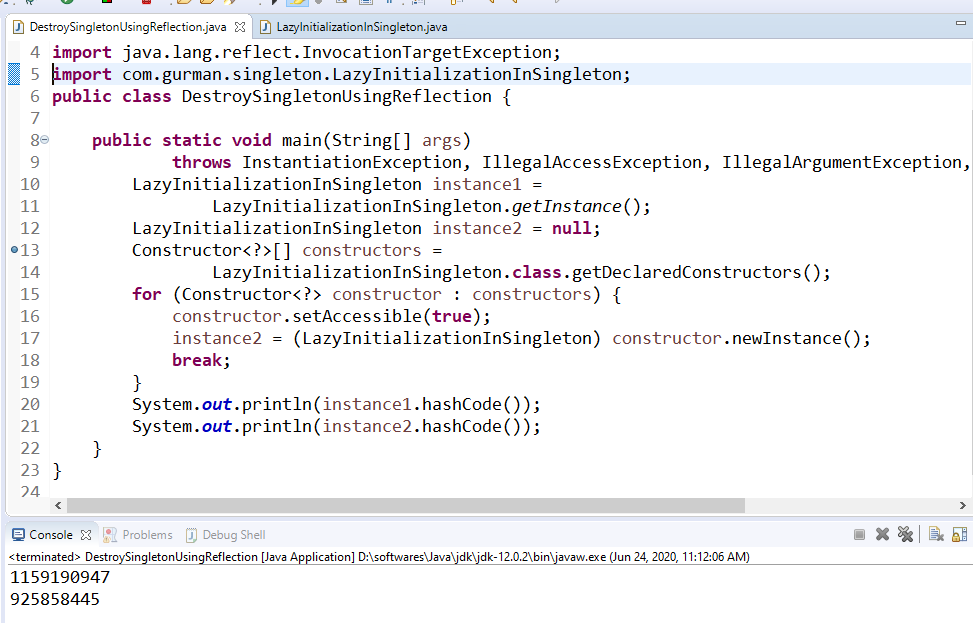






Destroy singleton using reflection

Using reflection we can access even the private methods of a class. Then we can create the instance using contructor result in creating more than 1 instance even in singleton approach. This can be saved using Enum approach.



### Creating Singleton using Enum

It is just like craeting an enum in place of class as we can't access Enum using reflection. But it is not feasible to use enum everywhere.

Destroy Singleton using serialization

As whenever we serailize an object and after deserialization, it creates a new instance instead of giving the same instance which can easily break singleton approach.