The Singleton pattern provides at most one instance of a given class and global access to its references. Very often is presented as an anti-pattern as it breaks two SOLID rules (single responsibility principle and open/closed principle), sometimes it is overused and very often is treated as object substitute of a global variable. It also happens to be problematic during implementation in multi-threaded environment. Use this pattern if your application requires global access to the same resource.

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, driver’s 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.

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.