



LEARN JAVA DESIGN PATTERNS

problem solving approaches

Design Patterns Tutorial

- Design Patterns - Home
- Design Patterns - Overview
- Design Patterns - Factory Pattern
- Abstract Factory Pattern
- Design Patterns - Singleton Pattern
- Design Patterns - Builder Pattern
- Design Patterns - Prototype Pattern
- Design Patterns - Adapter Pattern
- Design Patterns - Bridge Pattern
- Design Patterns - Filter Pattern
- Design Patterns - Composite Pattern
- Design Patterns - Decorator Pattern
- Design Patterns - Facade Pattern
- Design Patterns - Flyweight Pattern
- Design Patterns - Proxy Pattern
- Chain of Responsibility Pattern
- Design Patterns - Command Pattern
- Design Patterns - Interpreter Pattern
- Design Patterns - Iterator Pattern
- Design Patterns - Mediator Pattern
- Design Patterns - Memento Pattern
- Design Patterns - Observer Pattern
- Design Patterns - State Pattern
- Design Patterns - Null Object Pattern
- Design Patterns - Strategy Pattern
- Design Patterns - Template Pattern
- Design Patterns - Visitor Pattern
- Design Patterns - MVC Pattern
- Business Delegate Pattern
- Composite Entity Pattern
- Data Access Object Pattern
- Front Controller Pattern
- Intercepting Filter Pattern
- Service Locator Pattern
- Transfer Object Pattern

Design Patterns Resources

- Design Patterns - Questions/Answers
- Design Patterns - Quick Guide

Design Pattern - Singleton Pattern

[Previous Page](#)
[Next Page](#)

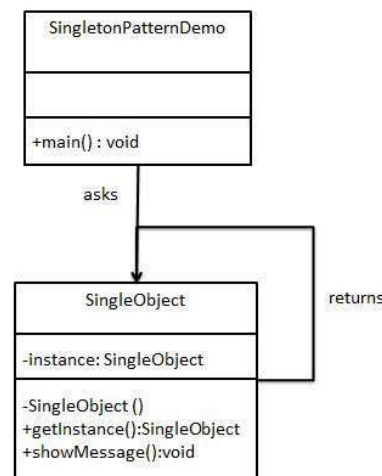
Singleton pattern is one of the simplest design patterns in Java. This type of design pattern comes under creational pattern as this pattern provides one of the best ways to create an object.

This pattern involves a single class which is responsible to create an object while making sure that only single object gets created. This class provides a way to access its only object which can be accessed directly without need to instantiate the object of the class.

Implementation

We're going to create a *SingletonObject* class. *SingletonObject* class have its constructor as private and have a static instance of itself.

SingletonObject class provides a static method to get its static instance to outside world. *SingletonPatternDemo*, our demo class will use *SingletonObject* class to get a *SingletonObject* object.



Step 1

Create a Singleton Class.

SingletonObject.java

```

public class SingletonObject {

    //create an object of SingletonObject
    private static SingletonObject instance = new SingletonObject();

    //make the constructor private so that this class cannot be
    //instantiated
    private SingletonObject(){}

    //Get the only object available
    public static SingletonObject getInstance(){
        return instance;
    }

    public void showMessage(){
        System.out.println("Hello World!");
    }
}
    
```

Step 2

Get the only object from the singleton class.

SingletonPatternDemo.java

```

public class SingletonPatternDemo {
    public static void main(String[] args) {
    }
}
    
```

Design Patterns - Useful Resources

Design Patterns - Discussion

Selected Reading

UPSC IAS Exams Notes

Developer's Best Practices

Questions and Answers

Effective Resume Writing

HR Interview Questions

Computer Glossary

Who is Who

```
//illegal construct
//Compile Time Error: The constructor SingleObject() is not visible
//SingleObject object = new SingleObject();

//Get the only object available
SingleObject object = SingleObject.getInstance();

//show the message
object.showMessage();
}
```

Step 3

Verify the output.

Hello World!

[Previous Page](#) [Print Page](#)

[Next Page](#)



[About us](#)

[Terms of use](#)

[Privacy Policy](#)

[FAQ's](#)

[Helping](#)

[Contact](#)

© Copyright 2020. All Rights Reserved.