

Singleton design pattern in Java

Singleton Pattern says that just "**define a class that has only one instance and provides a global point of access to it**".

In other words, a class must ensure that only single instance should be created and single object can be used by all other classes.

There are two forms of singleton design pattern

- **Early Instantiation:** creation of instance at load time.
- **Lazy Instantiation:** creation of instance when required.

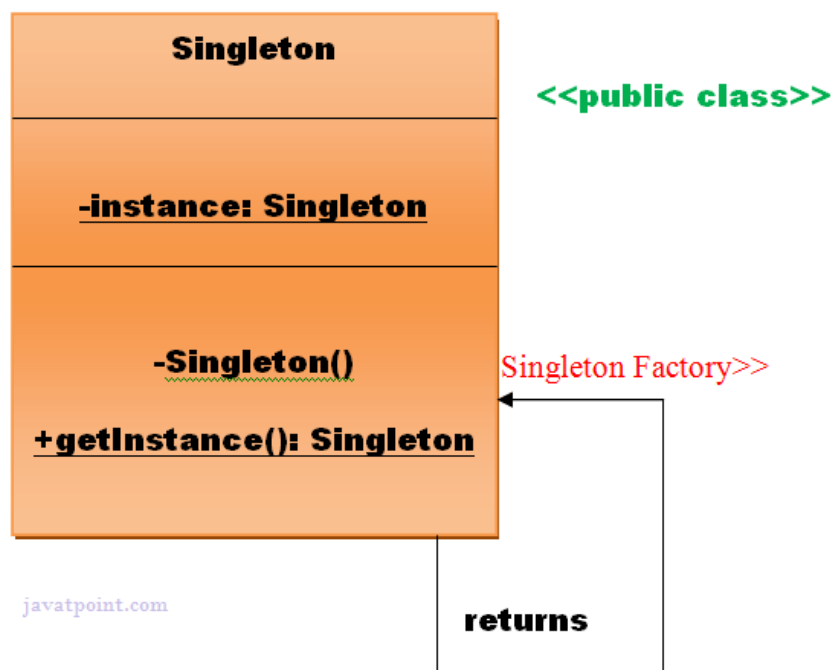
Advantage of Singleton design pattern

- Saves memory because object is not created at each request. Only single instance is reused again and again.

Usage of Singleton design pattern

- Singleton pattern is mostly used in multi-threaded and database applications. It is used in logging, caching, thread pools, configuration settings etc.

Uml of Singleton design pattern





100% Placement Assistance

Want to join a bigger, better company? St
Big Data from NIT Rourkela & Edureka

www.edureka.co

How to create Singleton design pattern?

To create the singleton class, we need to have static member of class, private constructor and static factory method.

- **Static member:** It gets memory only once because of static, it contains the instance of the Singleton class.
- **Private constructor:** It will prevent to instantiate the Singleton class from outside the class.
- **Static factory method:** This provides the global point of access to the Singleton object and returns the instance to the caller.

Understanding early Instantiation of Singleton Pattern

In such case, we create the instance of the class at the time of declaring the static data member, so instance of the class is created at the time of classloading.

Let's see the example of singleton design pattern using early instantiation.

File: A.java

```
class A{  
    private static A obj=new A();//Early, instance will be created at load time  
    private A(){}  
  
    public static A getA(){  
        return obj;  
    }  
  
    public void doSomething(){  
        //write your code  
    }  
}
```

Understanding lazy Instantiation of Singleton Pattern

In such case, we create the instance of the class in synchronized method or synchronized block, so instance of the class is created when required.

Let's see the simple example of singleton design pattern using lazy instantiation.

File: A.java

```
class A{  
    private static A obj;  
    private A(){}
```



```
public static A getA(){
    if (obj == null){
        synchronized(Singleton.class){
            if (obj == null){
                obj = new Singleton();//instance will be created at request time
            }
        }
    }
    return obj;
}

public void doSomething(){
    //write your code
}
}
```

Significance of Classloader in Singleton Pattern

If singleton class is loaded by two classloaders, two instance of singleton class will be created, one for each classloader.

Significance of Serialization in Singleton Pattern

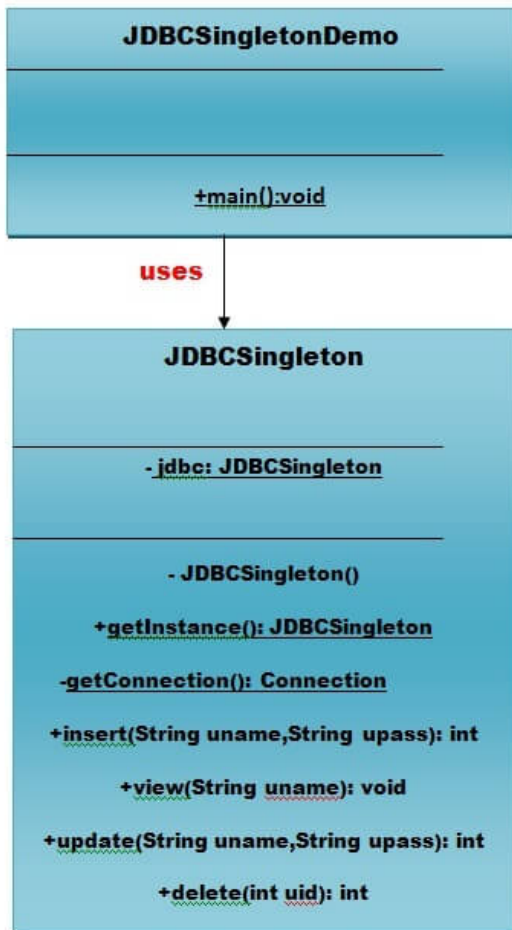
If singleton class is Serializable, you can serialize the singleton instance. Once it is serialized, you can deserialize it but it will not return the singleton object.

To resolve this issue, you need to override the **readResolve()** method that enforces the singleton. It is called just after the object is deserialized. It returns the singleton object.

```
public class A implements Serializable {
    //your code of singleton
    protected Object readResolve() {
        return getA();
    }
}
```

Understanding Real Example of Singleton Pattern

- We are going to create a JDBCSingleton class. This JDBCSingleton class contains its constructor as private and a private static instance jdbc of itself.
- JDBCSingleton class provides a static method to get its static instance to the outside world. Now, JDBCSingletonDemo class will use JDBCSingleton class to get the JDBCSingleton object.



Assumption: you have created a table `userdata` that has three fields `uid`, `uname` and `upassword` in `mysql` database. Database name is `ashwinirajput`, username is `root`, password is `ashwini`.

File: `JDBCSingleton.java`

```

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

class JDBCSingleton {
    //Step 1
    // create a JDBCSingleton class.
    //static member holds only one instance of the JDBCSingleton class.

    private static JDBCSingleton jdbc;

    //JDBCSingleton prevents the instantiation from any other class.
    private JDBCSingleton() { }

    //Now we are providing global point of access.
    public static JDBCSingleton getInstance() {

```



```

        if (jdbc==null)
        {
            jdbc=new JDBCSingleton();
        }
        return jdbc;
    }

```

// to get the connection from methods like insert, view etc.

```

private static Connection getConnection()throws ClassNotFoundException, SQLException
{

    Connection con=null;
    Class.forName("com.mysql.jdbc.Driver");
    con= DriverManager.getConnection("jdbc:mysql://localhost:3306/ashwanirajput", "root", "ashwani");
    return con;

}

```

//to insert the record into the database

```

public int insert(String name, String pass) throws SQLException
{
    Connection c=null;

    PreparedStatement ps=null;

    int recordCounter=0;

    try {

        c=this.getConnection();
        ps=c.prepareStatement("insert into userdata(uname,upassword)values(?,?)");
        ps.setString(1, name);
        ps.setString(2, pass);
        recordCounter=ps.executeUpdate();

    } catch (Exception e) { e.printStackTrace(); } finally{
        if (ps!=null){
            ps.close();
        }if(c!=null){
            c.close();
        }
    }
    return recordCounter;
}

```

//to view the data from the database

```

public void view(String name) throws SQLException
{
    Connection con = null;
    PreparedStatement ps = null;
    ResultSet rs = null;

```



```

    try {

        con=this.getConnection();
        ps=con.prepareStatement("select * from userdata where uname=?");
        ps.setString(1, name);
        rs=ps.executeQuery();
        while (rs.next()) {
            System.out.println("Name= "+rs.getString(2)+"\t"+"Paasword= "+rs.getString(3));
        }

    } catch (Exception e) { System.out.println(e);}
    finally{
        if(rs!=null){
            rs.close();
        }if (ps!=null){
            ps.close();
        }if(con!=null){
            con.close();
        }
    }
}

```

// to update the password for the given username

```

public int update(String name, String password) throws SQLException {
    Connection c=null;
    PreparedStatement ps=null;

    int recordCounter=0;
    try {
        c=this.getConnection();
        ps=c.prepareStatement(" update userdata set upassword=? where uname='"+name+"' ");
        ps.setString(1, password);
        recordCounter=ps.executeUpdate();
    } catch (Exception e) { e.printStackTrace(); } finally{

        if (ps!=null){
            ps.close();
        }if(c!=null){
            c.close();
        }
    }
    return recordCounter;
}

```

// to delete the data from the database

```

public int delete(int userid) throws SQLException{
    Connection c=null;
    PreparedStatement ps=null;
    int recordCounter=0;

```

```

    try {
        c=this.getConnection();
        ps=c.prepareStatement(" delete from userdata where uid='"+userid+"' ");
        recordCounter=ps.executeUpdate();
    } catch (Exception e) { e.printStackTrace(); }
    finally{
        if (ps!=null){
            ps.close();
        }if(c!=null){
            c.close();
        }
    }
    return recordCounter;
}

} // End of JDBCSingleton class

```

File: JDBCSingletonDemo.java

```

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

class JDBCSingletonDemo{
    static int count=1;
    static int choice;
    public static void main(String[] args) throws IOException {

        JDBCSingleton jdbc= JDBCSingleton.getInstance();

        BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
        do{
            System.out.println("DATABASE OPERATIONS");
            System.out.println(" ----- ");
            System.out.println(" 1. Insertion ");
            System.out.println(" 2. View    ");
            System.out.println(" 3. Delete  ");
            System.out.println(" 4. Update  ");
            System.out.println(" 5. Exit    ");

            System.out.print("\n");
            System.out.print("Please enter the choice what you want to perform in the database: ");

            choice=Integer.parseInt(br.readLine());
            switch(choice) {

                case 1:{
                    System.out.print("Enter the username you want to insert data into the database: ");

```



```
String username=br.readLine();
System.out.print("Enter the password you want to insert data into the database: ");
String password=br.readLine();

try {
    int i= jdbc.insert(username, password);
    if (i>0) {
        System.out.println((count++) + " Data has been inserted successfully");
    }else{
        System.out.println("Data has not been inserted ");
    }

} catch (Exception e) {
    System.out.println(e);
}

System.out.println("Press Enter key to continue...");
System.in.read();

} //End of case 1
break;
case 2:{
    System.out.print("Enter the username : ");
    String username=br.readLine();

    try {
        jdbc.view(username);
    } catch (Exception e) {
        System.out.println(e);
    }

    System.out.println("Press Enter key to continue...");
    System.in.read();

} //End of case 2
break;
case 3:{
    System.out.print("Enter the userid, you want to delete: ");
    int userid=Integer.parseInt(br.readLine());

    try {
        int i= jdbc.delete(userid);
        if (i>0) {
            System.out.println((count++) + " Data has been deleted successfully");
        }else{
            System.out.println("Data has not been deleted");
        }

    } catch (Exception e) {
        System.out.println(e);
    }

    System.out.println("Press Enter key to continue...");
```




```
        System.in.read();

    } // End of case 3
    break;
case 4: {
    System.out.print("Enter the username, you want to update: ");
    String username=br.readLine();
    System.out.print("Enter the new password ");
    String password=br.readLine();

    try {
        int i= jdbc.update(username, password);
        if (i>0) {
            System.out.println((count++) + " Data has been updated successfully");
        }

    } catch (Exception e) {
        System.out.println(e);
    }

    System.out.println("Press Enter key to continue...");
    System.in.read();

    } // end of case 4
    break;

default:
    return;
}

} while (choice!=4);
}
}
```

download this Singleton Pattern Example

Output

```
Command Prompt - java JDBCSingletonDemo
E:\All design patterns\Design patterns and their codes\3-Singleton Pattern>java
JDBCSingletonDemo
DATABASE OPERATIONS
-----
1. Insertion
2. View
3. Delete
4. Update
5. Exit

Please enter the choice what you want to perform in the database: 1
Enter the username you want to insert data into the database: himanshu
Enter the password you want to insert data into the database: 123@himanshu1987
1 Data has been inserted successfully
Press Enter key to continue...
```

```
Command Prompt - java JDBCSingletonDemo
E:\All design patterns\Design patterns and their codes\3-Singleton Pattern>javac
JDBCSingletonDemo.java
E:\All design patterns\Design patterns and their codes\3-Singleton Pattern>java
JDBCSingletonDemo
DATABASE OPERATIONS
-----
1. Insertion
2. View
3. Delete
4. Update
5. Exit

Please enter the choice what you want to perform in the database: 4
Enter the username for which you want to update the data into the database: ash
wani
Enter the new password 1987@ashwanirajput
1 Data has been updated successfully
Press Enter key to continue...
```

```
Command Prompt - java JDBCSingletonDemo
E:\All design patterns\Design patterns and their codes\3-Singleton Pattern>java
JDBCSingletonDemo
DATABASE OPERATIONS
-----
1. Insertion
2. View
3. Delete
4. Update
5. Exit

Please enter the choice what you want to perform in the database: 4
Enter the username for which you want to update the data into the database: ash
wani
Enter the new password 1987@ashwanirajput
1 Data has been updated successfully
Press Enter key to continue...

E:\All design patterns\Design patterns and their codes\3-Singleton Pattern>java
JDBCSingletonDemo
DATABASE OPERATIONS
-----
1. Insertion
2. View
3. Delete
4. Update
5. Exit

Please enter the choice what you want to perform in the database: 2
Enter the username you want to view the entire data from the database: ashwani
Name= ashwani Password= 1987@ashwanirajput
Press Enter key to continue...
```

[< prev](#)[next >](#)

Help Others, Please Share



Join Javatpoint Test Series

Placement Papers	AMCAT	Bank PO/Clerk	GATE
TCS	eLitmas	UPSSSC	NEET
HCL	Java	Government Exams	CAT
Infosys	Python	SSC	Railway
IBM	C Programming	Civil Services	CTET
Accenture	Networking	SBI	IIT JEE


Learn Latest Tutorials

Ansible Tutorial Ansible	Mockito Tutorial Mockito	Talend Tutorial Talend	Microsoft Azure Tutorial Azure
Sharepoint Tutorial SharePoint	Powershell Tutorial Powershell	Kali Linux Tutorial Kali Linux	OpenCV Tutorial OpenCV
Kafka Tutorial Kafka	Pandas Tutorial Pandas	Joomla Tutorial Joomla	Reinforcement Learning Tutorial Reinforcement













Preparation

Aptitude	Logical Reasoning	Verbal Ability	Interview Questions
----------	-------------------	----------------	---------------------



















Aptitude	Reasoning	Verbal A.	Interview
 Company Interview Questions Company			


Trending Technologies

 Artificial Intelligence Tutorial AI	 AWS Tutorial AWS	 Selenium tutorial Selenium	 Cloud tutorial Cloud
 Hadoop tutorial Hadoop	 ReactJS Tutorial ReactJS	 Data Science Tutorial D. Science	 Angular 7 Tutorial Angular 7
 Blockchain Tutorial Blockchain	 Git Tutorial Git	 Machine Learning Tutorial ML	 DevOps Tutorial DevOps


B.Tech / MCA

 DBMS tutorial DBMS	 Data Structures tutorial DS	 DAA tutorial DAA	 Operating System tutorial OS
 Computer Network tutorial C. Network	 Compiler Design tutorial Compiler D.	 Computer Organization and Architecture COA	 Discrete Mathematics Tutorial D. Math.
 Ethical Hacking Tutorial E. Hacking	 Computer Graphics Tutorial C. Graphics	 Software Engineering Tutorial Software E.	 html tutorial Web Tech.
 Cyber Security tutorial Cyber Sec.	 Automata Tutorial Automata	 C Language tutorial C	 C++ tutorial C++







Java tutorial
Java




.Net
Framework
tutorial
.Net



Python
tutorial
Python



List of
Programs
Programs



Control
Systems
tutorial
Control S.



Data Mining
Tutorial
Data Mining