	Course Name: Design Patterns/Thinking LAB	**EXPERIMENT NO. 9**	
	Course Code: 20CP210P **Faculty: Dr. Ketan Sabale**	**Branch: CSE**	**Semester: IV**
(To be filled by Student) **Submitted by: Jangle Parth** **Roll no: 22BCP083**			

Objective: To familiarize students with standard Structural design patterns.

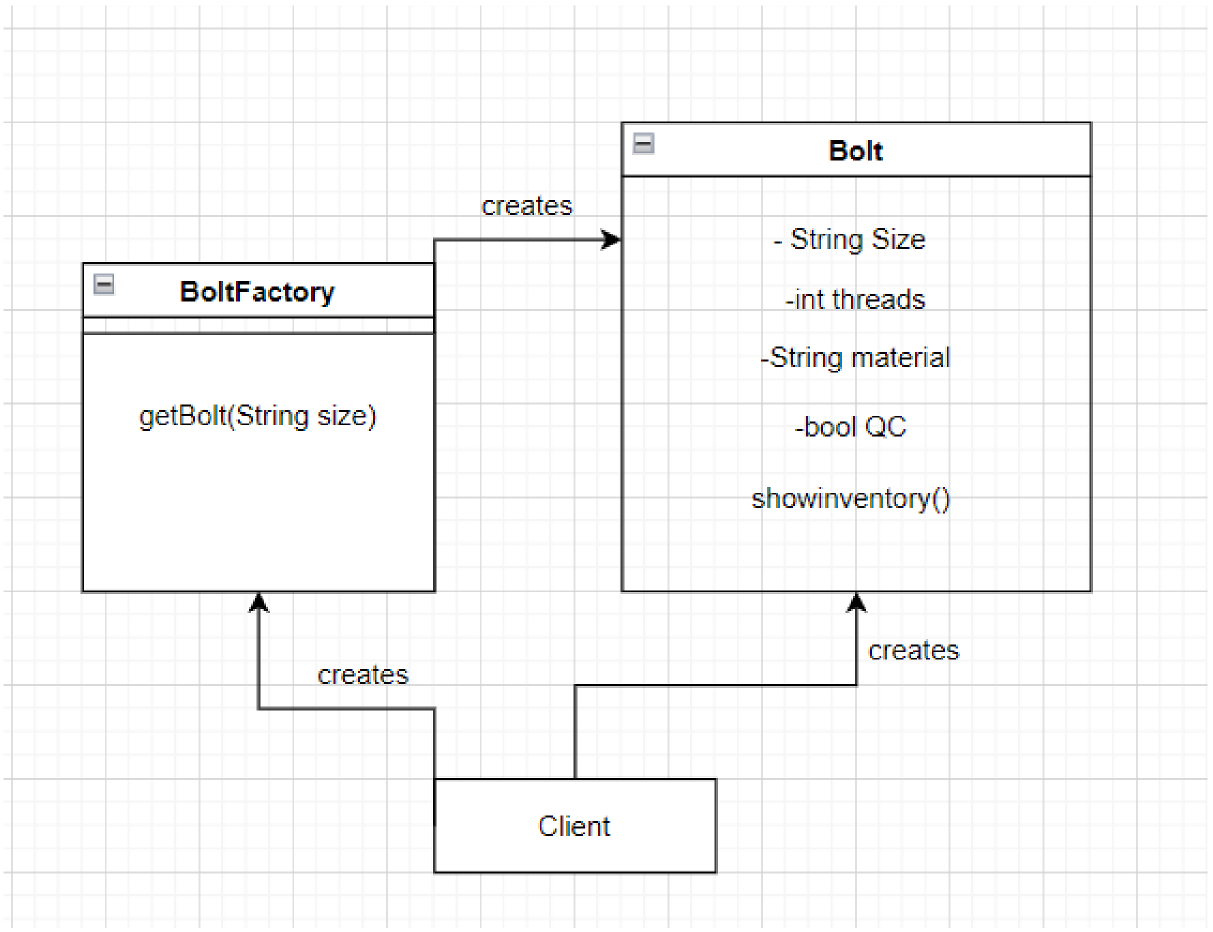
Experiment: Explain the Flyweight design pattern and write a program using any object-oriented programming language to demonstrate the working of Flyweight design pattern.

Theory: imagine a Scenario where in you are a factory owner and your factory is automated you can keep a track of everything that is being used. For a Big factory owner there is a Problem some items like screw which are used 1000 per day for them 1000 objects are created which can fill up the memory and make performance slow. So we use Flyweight Design Pattern wherein we create only one object of screw than other object are just references of te first obj with some changes if any.

Problem Statement Explanation:

We have a class bolt which has parameters like size, threads, material and QC. It has a method show inventory which show complete detail of screw. We have another class bolt factory which creates bolt there we have a HashMap bolt stock and a method get bolt to get a particular bolt. HashMap is used because it does not allow duplicate entries.

Flowchart Explanation:

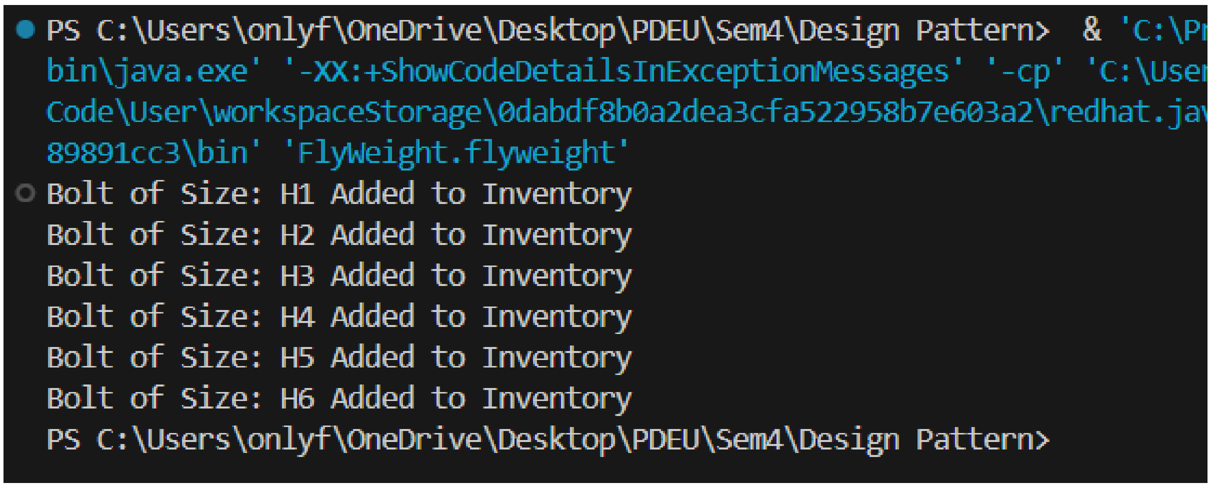


Code:

```java
package FlyWeight;

import java.util.HashMap;

class Bolt {
    String Size;
    int threads;
    String Material;
    boolean qc;

    Bolt(String size) {
        this.Size = size;
    }

    public void showinventory() {
        System.out.println("Size: " + Size + " Threads: " + threads + "
Material: " + Material + " QC OK: " + qc);
```

```
    }
}

class BoltFactory {
    static HashMap BoltStock = new HashMap();

    static Bolt getBolt(String Size) {
        Bolt bolt = (Bolt) BoltStock.get(Size);
        if (bolt == null) {
            bolt = new Bolt(Size);
            BoltStock.put(Size, bolt);
            System.out.println("Bolt of Size: " + Size + " Added to
Inventory");
        }
        return bolt;
    }
}

public class flyweight {
    public static void main(String[] args) {
        Bolt b1 = (Bolt) BoltFactory.getBolt("H1");
        b1.Material = "SS 304";
        b1.qc = true;
        b1.threads = 10;

        Bolt b2 = (Bolt) BoltFactory.getBolt("H2");
        b2.Material = "SS 304";
        b2.qc = true;
        b2.threads = 12;

        Bolt b3 = (Bolt) BoltFactory.getBolt("H3");
        b3.Material = "MS";
        b3.qc = false;
        b3.threads = 14;

        Bolt b4 = (Bolt) BoltFactory.getBolt("H4");
        b4.Material = "SS 316";
        b4.qc = true;
        b4.threads = 16;

        Bolt b5 = (Bolt) BoltFactory.getBolt("H5");
        b5.Material = "SS 304";
        b5.qc = true;
        b5.threads = 10;

        Bolt b6 = (Bolt) BoltFactory.getBolt("H6");
        b6.Material = "MS";
        b6.qc = false;
        b6.threads = 12;
    }
}
```

Output:



```
PS C:\Users\onlyf\OneDrive\Desktop\PDEU\Sem4\Design Pattern>  & 'C:\Pr
bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\User
Code\User\workspaceStorage\0dabdf8b0a2dea3cfa522958b7e603a2\redhat.jav
89891cc3\bin' 'FlyWeight.flyweight'
Bolt of Size: H1 Added to Inventory
Bolt of Size: H2 Added to Inventory
Bolt of Size: H3 Added to Inventory
Bolt of Size: H4 Added to Inventory
Bolt of Size: H5 Added to Inventory
Bolt of Size: H6 Added to Inventory
PS C:\Users\onlyf\OneDrive\Desktop\PDEU\Sem4\Design Pattern>
```