**LAB ASSIGNMENT 11**

**FLYWEIGHT DESIGN PATTERN**

Flyweight DP is a structural design pattern that lets you fit more objects into the available amount of RAM by sharing common parts of state between multiple objects instead of keeping all of the data in each object.

CODE:

* Instrument.java

package flyweightDP;

public interface Instrument {

*void* play();

 }

* Harmonium.java

public class Harmonium implements *Instrument* {

    private *String* wood;

    public Harmonium(*String* *wood*){

       this.wood = wood;

    }

    @*Override*

    public *void* play() {

       System.out.println(wood + "-wood Harmonium");

    }

}

* HarmoniumFactory.java

package flyweightDP;

import java.util.HashMap;

public class HarmoniumFactory {

   private static final *HashMap* hm = new HashMap();

   public static *Instrument* getHarmonium(*String* *wood*) {

*Harmonium* p = (Harmonium)hm.get(wood);

      if(p == null) {

         p = new Harmonium(wood);

          hm.put(wood, p);

         System.out.println("Creating Harmonium of wood type :  " + wood);

       }

      return p;

   }

}

* FlyweightDP.java

public class FlyweightDP {

    private static final *String* choice[] = { "Mahogany", "Teak", "Maple" };

    public static *void* main(*String*[] *args*) {

       for(*int* i=0; i < 9; ++i) {

*Harmonium* p = (Harmonium) HarmoniumFactory.getHarmonium(getRandomChoice());

          p.play();

       }

    }

    private static *String* getRandomChoice() {

       return choice[(*int*)(Math.random()\*choice.length)];

    }

 }

OUTPUT:

