**LAB ASSIGNMENT 13**

**ITERATOR DESIGN PATTERN**

ITERATOR DP is a structural design pattern in which client does not know the internal data structure of concrete containers instead it uses iterator.

CODE:

* iIterator.java

package iteratordp;

public interface iIterator {

    public *boolean* hasnext();

    public *Object* next();

}

* iContainer.java

package iteratordp;

public interface iContainer {

    public iIterator getIterator();

}

* MusicContainer.java

package iteratordp;

public class MusicContainer implements iContainer{

    private *String* instruments[] = {"Piano","Guitar","Violin","Drum"};

    @*Override*

    public iIterator getIterator() {

        return new MusicIterator();

    }

    private class MusicIterator implements iIterator{

        private *int* index;

        @*Override*

        public *boolean* hasnext() {

            if (index < instruments.length){

                return true;

            }

else {return false;}

        }

        @*Override*

        public *Object* next() {

            if (this.hasnext()){

                return instruments[index++];

            }else {

                return null;

            }

        }

    }

}

* Client.java

package iteratordp;

public class Client {

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

        iContainer container = new MusicContainer();

        iIterator iterator = container.getIterator();

        while (iterator.hasnext()){

*Object* obj = iterator.next();

            System.out.println(obj);

        }

    }

}

OUTPUT:

