	Course Name: Design Patterns/Thinking LAB	**EXPERIMENT NO. 11**	
	Course Code: 20CP210P **Faculty: Dr. Ketan Sabale**	**Branch: CSE**	**Semester: IV**
Submitted by: Rhythm Shah **Roll no:** 22BCP071			

Objective: To familiarize students with standard Behavioral design patterns.
Experiment: Explain the Iterator design pattern and write a program using any object-oriented programming language to demonstrate the working of Iterator design pattern.

Theory: -

Iterator is a behavioural design pattern that lets you traverse elements of a collection without exposing its underlying representation (list, stack, tree, etc.).

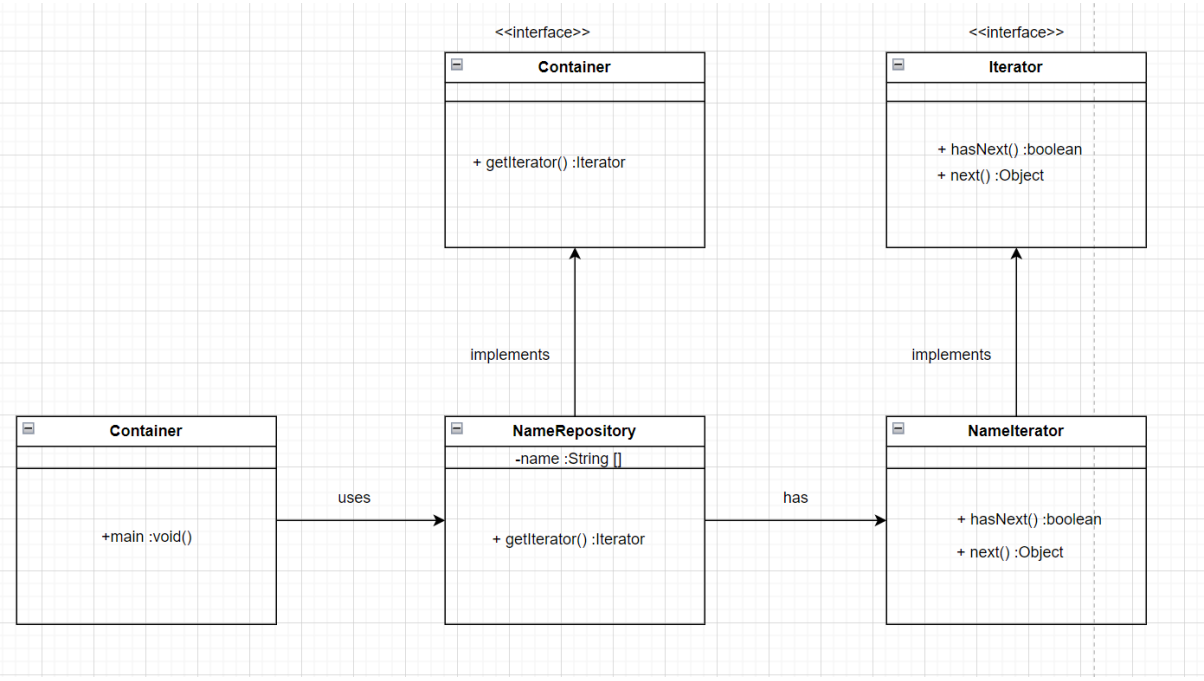
A behavioral design pattern called the Iterator design pattern makes it possible to access an aggregate object's (such as a list's) components in a sequential manner without disclosing the representation that underlies it. The aggregate can alter its internal structure without changing how its elements are accessed since it creates a distinct object called an iterator that contains the details of traversing the aggregate's elements.

A common and rather easy design pattern is the iterator pattern. Every language has a large number of data structures and collections available.
An iterator that allows it to loop through its objects must be provided by each collection. But it must take care to avoid revealing its implementation while doing so.

Implementation: -

In this this pattern the implementation is done of Iterator design pattern through takings name of laptop shops.
Two interfaces are created
One is Iterator and the other is Container where nameRepository implements Conatiner and nameIterator implements Iterator. Two methods are created in Iterator interface which are hasNext() and next().
And one main class to created objects and printing outputs.

UML Diagram: -



Code:-

```
interface Iterator {
    public boolean hasNext();
    public Object next();
 }

interface Container {
    public Iterator getIterator();
 }

class NameRepository implements Container {
    public String Shopnames[] = {"Lenovo" , "DELL" ,"HP" , "ASUS"};
```

```
   @Override
   public Iterator getIterator() {
      return new NameIterator();
   }

   private class NameIterator implements Iterator {

      int index;

      @Override
      public boolean hasNext() {

         if(index < Shopnames.length){
            return true;
         }
         return false;
      }

      @Override
      public Object next() {

         if(this.hasNext()){
            return Shopnames[index++];
         }
         return null;
      }
   }
}

public class iterator_pattern {

   public static void main(String[] args) {
      NameRepository namesRepository = new NameRepository();

      for(Iterator iterator = namesRepository.getIterator();
iterator.hasNext();){
         String name = (String)iterator.next();
         System.out.println("ShopName : " + name);
      }
   }
}
```

Output: -

```
PS E:\Fourth sem\Design pattern lab> cd "e:\Fourth sem\De
a } ; if ($?) { java iterator_pattern }
ShopName : Lenovo
ShopName : DELL
ShopName : HP
ShopName : ASUS
PS E:\Fourth sem\Design pattern lab>
```