**Iterator Design Pattern**

**Assignment - 1**

Name - Heet Dobariya Roll No. - 22BCP177 Group - G5

* **Iterator Design Pattern :**

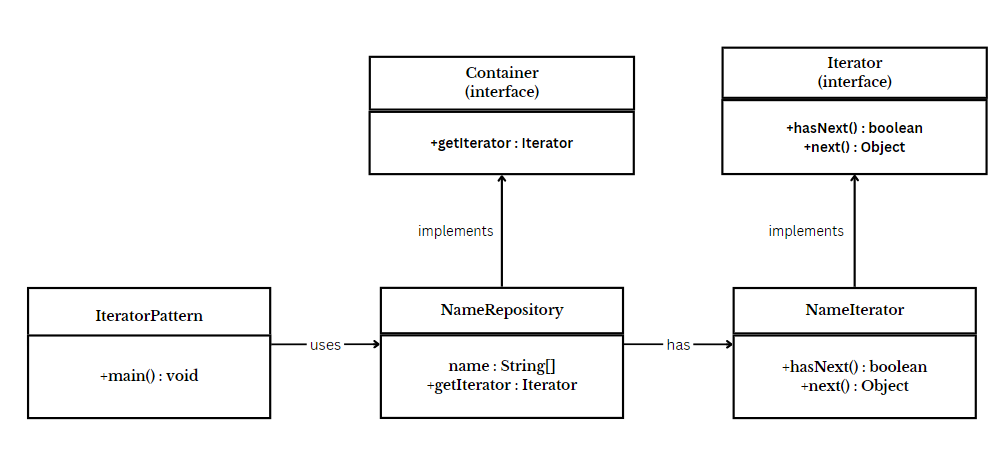
The Iterator Design Pattern is a behavioral design pattern that provides a way to access elements of an aggregate object sequentially without exposing its underlying representation. It is commonly used to traverse collections of objects in a uniform manner without needing to know the internal structure of the collection.

The key components of the Iterator pattern are:

1. Iterator Interface: This interface defines methods for traversing the collection of objects. It typically includes methods like next(), hasNext(), first(), last(), etc., depending on the requirements.
2. Concrete Iterator: This class implements the Iterator interface and maintains the current position in the traversal of the aggregate object. It is responsible for managing the iteration over the collection.
3. Aggregate Interface: This interface defines methods for creating iterators. It could be an abstract class or an interface depending on the design.
4. Concrete Aggregate: This class implements the Aggregate interface and provides methods for creating iterators. It represents a collection of objects that the iterator can traverse.

By using the Iterator pattern, the client code can traverse through elements of a collection without knowing its internal structure. This promotes a clean separation of concerns between the client and the collection, making the code more modular and easier to maintain.

* **Program :** Implement iterator design pattern for name example.
* **UML Diagram :**



* **Code :**

interface Iterator

{

public boolean hasNext();

public Object next();

}

interface Container

{

public Iterator getIterator();

}

class NameRepository implements Container

{

public String names[] = {"Heet", "Devanshi", "Meet", "Maitri"};

public Iterator getIterator()

{

return new NameIterator();

}

class NameIterator implements Iterator

{

int index;

public boolean hasNext()

{

return index < names.length;

}

public Object next()

{

if (this.hasNext())

{

return names[index++];

}

return null;

}

}

}

public class IteratorDesign

{

public static void main(String[] args)

{

NameRepository nameRepository = new NameRepository();

for (Iterator iter = nameRepository.getIterator(); iter.hasNext(); )

{

String name = (String) iter.next();

System.out.println("Name : " + name);

}

}

}

* **Output :**

