**Experiment – 12**

**Aim** –Iterator design pattern (Behavioral pattern)

**Concept** -**client does not know the internal data structure of concrete containers instead it uses iterator.**

iIterator.java

|  |
| --- |
| public interface iIterator {      public boolean hasnext();      public Object next();  } |

iContainer.java

|  |
| --- |
| public interface iContainer {      public iIterator getIterator();  } |

MusicContainer.java

|  |
| --- |
| 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

|  |
| --- |
| 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 :**

Piano

Guitar

Violin

Drum