Exercise No.2
(Inheritance/Encapsulate/Polymorphism)

Q1

Implement MizuhoBank and UFJBank classes that extend AbstractBank shown as follows. In addition, implement Q2Main that include main method and the result of the execution should be the same as following.

AbstractBank.java

```
public abstract class AbstractBank {
    public abstract String getName();
    public abstract int getMaxDeposit();
    public abstract int getFee();
}
```

```
1
2  public class Q21Main {
3
4    public static void main(String[] args) {
5      AbstractBank bank = null;
6
7      bank = new MizuhoBank();
8      System.out.println(bank.getName());
9      System.out.println(bank.getMaxDeposit());
10     System.out.println(bank.getFee());
11
12     System.out.println();
13
14     bank = new UFJBank();
15     System.out.println(bank.getName());
16     System.out.println(bank.getMaxDeposit());
17     System.out.println(bank.getFee());
18   }
19 }
```

Listing 1: Q21Main

Execution Example

```
%java Q21Main
This is Mizuho Bank
100000
210

UFJ
2000000
105
```

Q2

Implement ReverseList class that has two methods: add(String str) and get(). add(String str) inserts the argument into the list, also get() bring a string out by reverse order. You also have to implement Q22Main that uses ReverseList. Note that, List and/or ArrayList Java prepares can be used to implement ReverseList.

Examples of main mehtod in Q22Main and an execution example

```
public static void main(String[] args) {
    ReverseList rlist = new ReverseList();

    rlist.add("test1");
    rlist.add("test2");
    rlist.add("test3");
    rlist.add("test4");
    rlist.add("test5");

    String str = null;
    while((str = rlist.get()) != null) {
        System.out.println(str);
    }
}

%java Q21Main
test5
test4
test3
test2
test1
```

Q3

Implement MultiCollection class that includes List and Map interface Java prepares. This class has two method:addList(Object element inserts the argument into List and addMap(Object key, Object value stores value on Map by using key. Also, implement showAll method that displays the result when Q23Main is executed. Note that, you can implement the collection framework of Java such as List and/or Map.

An example of MultiCollection

```
public class MultiCollection {
    public void addList(Object str) {
        .....
    }
    public void addMap(Object key, Object value) {
        .....
    }
    public void showAll() {
        .....
    }
}
```

```
1
2  public class Q23Main {
3
4    public static void main(String[] args) {
5      MultiCollection multiCol = new MultiCollection();
6
7      multiCol.addList("AAA");
8      multiCol.addList("BBB");
9      multiCol.addList("CCC");
10
11     multiCol.addMap("X", 100);
12     multiCol.addMap("Y", 200);
13     multiCol.addMap("Z", 300);
14
15     multiCol.showAll();
16
17
18   }
19
20 }
```

Listing 2: Q23Main

Execution example

```
%java Q23Main
## List ##
AAA
BBB
CCC
## Map ##
Y : 200
X : 100
Z : 300
```

Q4

Map interface that Java prepares does not guarantee the order when a set of key and value is stored. Therefore, implement OrderdHashTable that guarantees the order. This class prepares methods as follows.

- void put(Object key, Object value)
  stores value on Map by using key.
- Object get(Object key)
  returns value associated with key.
- List getKeyList()
  returns a list of key as List.

Implement these method that display the result when Q24Main is executed as follows. Nota that, when the key is duplicated because of multiple insertions with the same key, the order of keys should not be changed. You can use Map and/or List Java prepares in order to implement.

```
1  import java.util.List;
2
3
4  public class Q24Main {
5
6    public static void main(String[] args) {
7      OrderedHashMap oMap = new OrderedHashMap();
8
9      oMap.put("AAA", 100);
10     oMap.put("BBB", 200);
11     oMap.put("CCC", 300);
12     oMap.put("DDD", 400);
13     oMap.put("EEE", 500);
14     oMap.put("DDD", 1000);
15     oMap.put("DDD", 2000);
16     oMap.put("XYX", 10000);
17
18     List keyList = oMap.getKeyList();
19     for(int i = 0; i < keyList.size(); i++) {
20       Object key = keyList.get(i);
21       System.out.println(key + " = " + oMap.get(key));
22     }
23
24   }
25
26 }
```

Listing 3: Q24Main

Execution example

```
AAA = 100
BBB = 200
CCC = 300
DDD = 2000
EEE = 500
XYX = 10000
```

Q5

Implement MyIterator that implements java.util.Iterator interface. java.util.Iterator interface prepares the following methods. Details are shown in JavaDoc.

- boolean hasNext()
  return true if the next element exists.
- Object next()
  return a next element.
- void remove()
  remove the last element.
- Iterator iterator()
  return an object that implements java.util.Iterator.

Implement this class that displays the result when Q25Main is executed as follows. Note that, MyIterator can only stores ten objects as an uppter limitation.

```
1  import java.util.Iterator;
2
3
4  public class Q25main {
5
6    public static void main(String[] args) {
7      MyIterator myIterator = new MyIterator();
8
9      for(int i = 0; i < 20; i++) {
10       myIterator.add(i);
11     }
12
13     myIterator.remove();
14     myIterator.add(30);
15
16     Iterator ite = myIterator.iterator();
17     while(ite.hasNext()) {
18       System.out.println(ite.next());
19     }
20
21   }
22
23 }
```

Listing 4: Q25Main

Execution Example

```
0
1
2
3
4
5
6
7
8
30
```