NSBT

CORE JAVA - Day 10&11 Assignment

1. What are the three different ways of creating Singleton Objects? Show a demo of all three.

2. Find out which of the SOLID principles are used in Java API and which part of API. Explain the same citing the references in a pdf file.

3. Re-create the addLoanProduct() method of Bank Class created in Question-1 of Day-6 assignment with the help of Builder Pattern.

4. Implement Factory Design pattern to create the object of Bank Class using the interfaces – Maker and Operator as mentioned in the Question – 3 of Day-6 assignment. The object of the bank must be accessed using the command mentioned below:

```
Maker maker = BankFactory.getMakerInstance();

or

Operator operator = BankFactory.getOperatorInstance();
```

5. Change the method – printAllCustomers() of the same class as above Question. While we are printing the array of Customers, we need to sort the same based on the requirement of the User. If user wants the Customers to be printed sorted based on Customer Name, it must print the result in the same manner. If user wants Customers to be printed sorted based on totalMonthlyExpense, it must be able to do the same. Use Strategy Pattern to implement the same as mentioned below:

```
interface Comparator{
        boolean compare(Customer c1, Customer c2);
}

class NameCompare implements Comparator{
   public boolean compare(Customer c1, Customer c2){
        return c1.getName().compareTo(c2.getName());
   }
}
--> In the same manner create multiple classes based on different requirements
//In the Bank Class, change the following:
class  Bank{

void printAllCustomers(Comparator comparator){

/* Use bubble sort to sort the array of objects and use comparator.compare()
to compare two objects */
}

}
```

6. Assume we already have a NumberSorter available which sorts a list of Integers as displayed below:

```
import java.util.*;
public class NumberSorter{
  public List<Integer> sort(List<Integer> numbers)   {
       Collections.sort(numbers);
       return numbers;
  }
}
```

The requirement is to sort an array of integers. Use Adapter pattern to sort the integer array which inturn will use the NumberSorter class provided to us as mentioned below:

```
public class SortListAdapter implements Sorter{
    public int[] sort(int[] numbers)   {
           //convert the array to a List
          //call the adapter
         NumberSorter sorter = new NumberSorter();
         numberList = sorter.sort(numberList);
         //convert the list back to an array and return
         return sortedNumbers;
    }
}
```

7. Findout which part of Java API is based on Adapter pattern. Document your findings.