CS 542 Design Pattern and Object-Oriented Analysis

Lab 4

Student Name		Student CSUSM ID	Contribution percentage
1	Niki Patel	200360048	33.33%
2	Dhruval Shah	200361439	33.33%
3	Rishi Ramrakhyani	200024661	33.33%

Grading Rubrics (for instructor only):

Criteria	1. Beginning	2. Developing	3. Proficient	4. Exemplary
Modeling	0-14	15-19	20-24	25-30
Program: functionality ***correctness***	0-9	10-14	15-19	20
Program: functionality ***Behavior Testing***	0-9	10-14	15-19	20
Program: quality -> ***Readability***	0-2	3-5	6-9	10
Program: quality -> ***Modularity***	0-2	3-5	6-9	10
Program: quality -> ***Simplicity***	0-2	3-5	6-9	10
Total Grade (100)				

CS 542 Design Pattern and Object-Oriented Analysis

Problems:

The ABC Company typically uses an object of the SortingUtility class to sort products. A product has at least three attributes: ID, name and price. All are accessible through their corresponding get() method but the ID is fixed once set.

The SortingUtility class implements two private sorting algorithms, bubbleSort and quickSort, each of which takes the list of products and returns an ordered list of products. The SortingUtility class also has a public method List<Product> sort(List<Product> items, int sortingApproach), which simply calls the specified sorting approach (i.e., bubbleSort or quickSort) to return a list of sorted products to its client.

The SortingUtility currently does not log the list of sorted products before returning it to the client. Now the ABC Company would like to have an improved sorting service that can log (for this lab, simply printing to the display console) the list of sorted products before returning it to the client. To implement this improved service you cannot change **the existing** SortingUtility **class for compatible reason**. Moreover, the returned products from bubbleSort should be logged (printed) with ID followed by name and price, whereas the returned products from the quicksort should be logged (printed) with name first followed by ID and price.

(30 pts) What design pattern can be used? Document your pattern-based design in UML class diagram, ensure attributes, methods, visibility, arguments and relationships are correctly included.

(70 pts) Implement code in Java. Implement two test scenarios: one using quicksort to sort a list of products such as books, bags, and buttons, another using bubblesort to sort the same list of products.

Solution:

- First, remember to zip the src folder of your project and submit the zip file to the ungraded assignment named "Lab4CodeSubmission". One submission from each team.
- Paste a screenshot of a run of your program here.

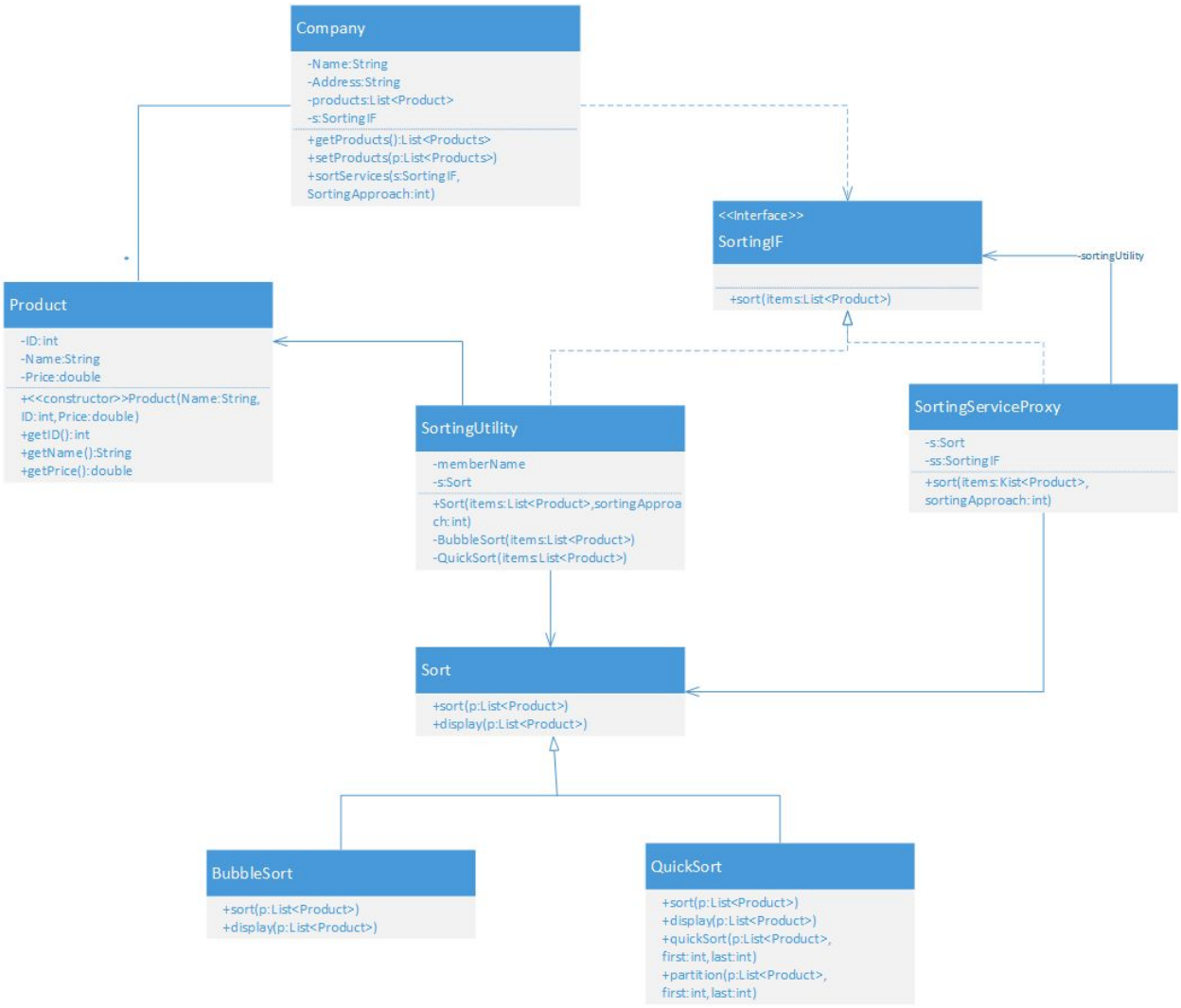
CS 542 Design Pattern and Object-Oriented Analysis

- Also paste all you source code here.
- Save this report in PDF, then **each student** needs to submit the pdf report to the graded assignment named "Lab4ReportSubmission".

Solution:

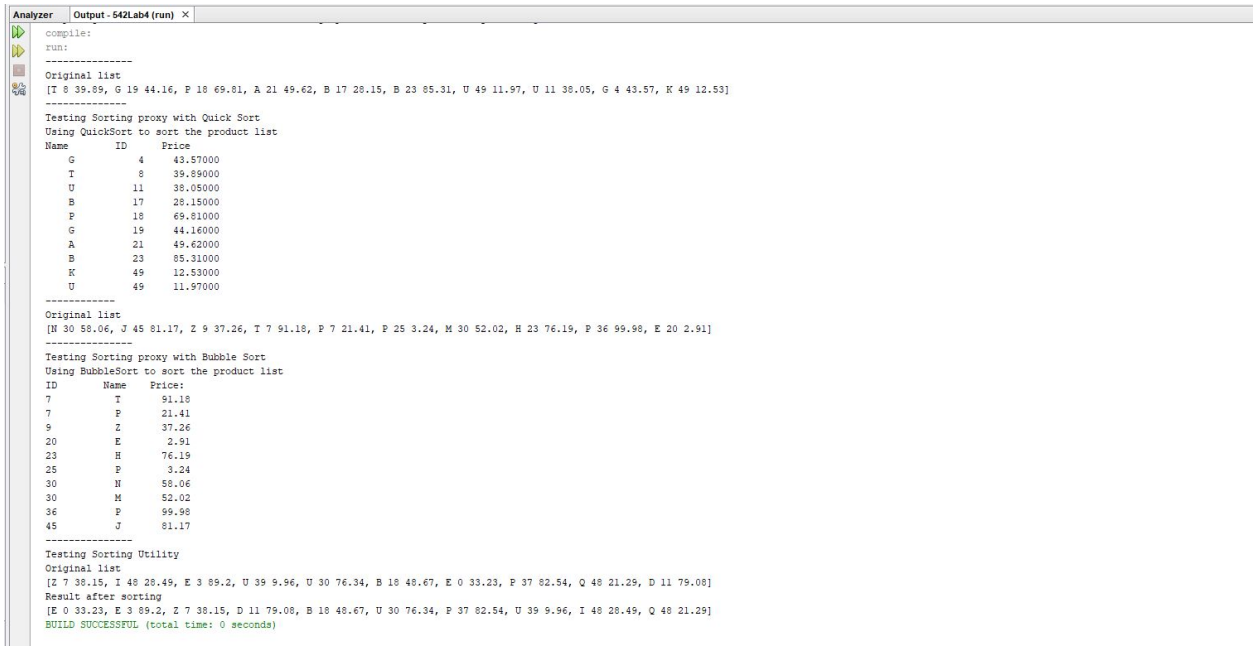
Immutable Interface and Proxy are the design patterns that are been used.

UML Diagram



CS 542 Design Pattern and Object-Oriented Analysis

Output:



Code:

Class Company.java

```
package pkg542lab4;
import java.util.List;

public class Company {
    private String Name;
    private String Address;
    private SortingIF s;
    private List<Product> p;
```

```
    public Company(String name, String address){
        this.Name=name;
        this.Address=address;
        this.p=null;
    }
   public Company(String name, String address, List<Product>products){
     this.Name=name;
     this.Address=address;
     this.p=products;
   }
   public void setProducts(List<Product> p){
      this.p=p;
   }
   public List<Product>getProducts(){
      return this.p;
   }

    public void sortService(SortingIF s, int sortingApproach) {
        this.s = s;
        s.sort(p, sortingApproach);
    }
}
```

Class Product.java

```
package pkg542lab4;

import java.util.List;

public class Product {
    private final String Name;
    private final int ID;
    private final double Price;

    public Product(String Name,int ID, double Price){
        this.Name=Name;
        this.ID=ID;
        this.Price=Price;
```

```
    }
    public String getName(){
        return this.Name;
    }
    public int getID(){
        return this.ID;
    }
    public double getPrice(){
        return this.Price;
    }
    @Override
    public String toString(){
        return getName()+" "+getID()+" "+getPrice();
    }
}
```

Class SortingUtility.java

```
package pkg542lab4;
import java.util.List;

public class SortingUtility implements SortingIF {
    private Sort s;
    private List<Product>p;

    public SortingUtility(){

    }
    @Override
   public List<Product>sort(List<Product>items,int sortingApproach){
        this.p=items;
        if(sortingApproach==1){
            bubbleSort(items);
        }else if(sortingApproach==2){
            quickSort(items);
        }
        return items;
    }
```

```
  private void bubbleSort(List<Product>items){
     s=new BubbleSort();
     s.sort(items);
  }
  private void quickSort(List<Product>items){
     s=new QuickSort();
     s.sort(items);
  }
}
```

Interface SortingIF.java

```
package pkg542lab4;
import java.util.List;
public interface SortingIF{

 public List<Product>sort(List<Product>items,int sortingApproach );

}
```

Class SortingServiceProxy.java

```
package pkg542lab4;

import java.util.List;

public class SortingServiceProxy implements SortingIF {
    private Sort s;
    private SortingIF ss;

    public SortingServiceProxy(){
        this.ss=new SortingUtility();
    }
    @Override
   public List<Product>sort(List<Product>items, int sortingApproach)
{
```

```
 ss.sort(items,sortingApproach);

        if(sortingApproach==1){
          s=new BubbleSort() {};
       }
       else if(sortingApproach==2){
          s=new QuickSort();
       }
       s.display(items);
       return items;
   }


}
```

Class Sort.java

```
package pkg542lab4;

import java.util.List;
public abstract class Sort {
    public abstract void sort(List<Product> p);

    public abstract void display(List<Product> p);
}
```

Class BubbleSort.java

```
package pkg542lab4;

import java.util.List;

public class BubbleSort extends Sort {
```

```
    private int ID;
    private String Name;
    private double Price;

    public int getID(){
        return this.ID;
    }
    public String getName(){
        return this.Name;
    }
    public double getPrice(){
        return this.Price;
    }

    @Override
    public void sort(List<Product>p){
        int n=p.size();
        for(int i=0;i<n;i++){
            for(int j=0;j<n-i-1;j++){
                if(p.get(j).getID()>p.get(j+1).getID()){
                    Product temp=p.get(j);
                    p.set(j,p.get(j+1));
                    p.set(j+1,temp);
                }
            }
        }
    }
    @Override
   public  void display(List<Product> p){
        System.out.println("Using BubbleSort to sort the product list");
        System.out.printf("%s%12s%11s","ID","Name","Price:\n");
        for(Product item: p)
        {
            System.out.printf("%-2s %10s %12s", item.getID(), item.getName(), item.getPrice() +
"\n");
        }
    }

}
```

Class QuickSort.java

```
package pkg542lab4;

import java.util.List;
        public class QuickSort extends Sort {


    @Override
   public void sort(List<Product>p){
      quickSort(p,0,p.size()-1);
   }
   private void quickSort(List<Product>p,int first,int last){
      if(first<last){
         int pivIndex=partition(p,first,last);
         quickSort(p,first,pivIndex-1);
         quickSort(p,pivIndex+1,last);

      }
   }
   private int partition(List<Product>p,int first,int last){
      sortMid(p,first,last);
      swap(p,first,(first+last)/2);
      Product pivot=p.get(first);
      int head=first;
      int tail=last;
      while(head<tail){
         while(head<last&&pivot.getID()>=p.get(head).getID()){
            head++;
         }
         while(pivot.getID()<p.get(tail).getID()){
            tail--;
         }
         if(head<tail){
            swap(p,head,tail);
         }
      }
      swap(p,first,tail);
      return tail;
   }
```

```
  private void sortMid(List<Product>p,int first,int last){
     int mid=(first+last)/2;
     if(p.get(mid).getID()<p.get(first).getID()){
        swap(p,first,mid);
     }
     if (p.get(last).getID()<p.get(mid).getID()){
        swap(p,mid,last);
     }
     if(p.get(mid).getID()<p.get(first).getID()){
        swap(p,first,mid);
     }
  }
  private void swap(List<Product>p, int index1,int index2){
     Product temp=p.get(index1);
     p.set(index1,p.get(index2));
     p.set(index2,temp);
  }


   @Override
  public void display(List<Product>p){
     System.out.println("Using QuickSort to sort the product list");
     System.out.printf("%s%10s%11s\n","Name","ID","Price");
     for(Product item:p){
        System.out.printf("%5s%13.5f%12s",item.getName(),item.getID(),item.getPrice());
        System.out.println ();       }
  }


}
```

Main Class

```
package pkg542lab4;

import java.util.ArrayList;
import java.util.List;
import java.util.Random;
```

```
public class Main {
 public static void main(String[] args) {
       List<Product>items=generateList();
       Company ABC =new Company("ABC","San Marcos",items);
       System.out.println("---------------");
       System.out.println("Original list");
       System.out.println(ABC.getProducts());
       System.out.println("--------------");

       System.out.println("Testing Sorting proxy with Quick Sort");
       SortingIF service=new SortingServiceProxy();
       ABC.sortService(service,2);

       System.out.println("------------");
       items=generateList();
       ABC.setProducts(items);
       System.out.println("Original list");
       System.out.println(ABC.getProducts());
       System.out.println("---------------");
       System.out.println("Testing Sorting proxy with Bubble Sort");
       ABC.sortService(service,1);

       System.out.println("---------------");
       System.out.println("Testing Sorting Utility");
       service=new SortingUtility();
       System.out.println("Original list");
       items=generateList();
       ABC.setProducts(items);
       System.out.println(ABC.getProducts());
       ABC.sortService(service,1);
       System.out.println("Result after sorting");
       System.out.println(ABC.getProducts());
          }
   private static List<Product>generateList(){
       String names="ABCDEFGHIJKLMNOPQRSTUVWXYZ";
       Random r=new Random();
       List<Product>items=new ArrayList();
       int totalNum=10;
       for(int i=0;i<totalNum;i++){
          StringBuilder sb=new StringBuilder();
          sb.append(names.charAt(r.nextInt(names.length())));
```

```
            String name=sb.toString();
            int ID=r.nextInt(50);
            double price=0.0+(r.nextDouble()*(100.0-0.0));
            price=(double)Math.round(price*100)/100;
            items.add(new Product(name,ID,price));
        }
        return items;
    }
}
```