CS1083
Assignment #2

Daniyal Khan
3765942

Purchasable Interface:

```
public interface Purchasable {
    public String getTitle();
    public double getSellingPrice();
}
```

Item.java:

```
public abstract class Item implements Purchasable,
Comparable<Item> {
    private String title;
    private double initialPrice;

    public Item(String title, double initialPrice) {
        this.title = title;
        this.initialPrice = initialPrice;
    }

    public String getTitle() {
        return title;
    }

    public double getInitialPrice() {
        return initialPrice;
    }

    public String toString() {
        return title;
    }

    public int compareTo(Item other) {
        int titleComparison =
this.title.compareTo(other.getTitle()); // calculate the
difference in the titles aplhabetically

        if (titleComparison < 0) { // we only want to return -1,
1 or 0
            return -1;
        } else if (titleComparison > 0) {
            return 1;
        }
```

```
        // if title were aplhabetically; compare the prices
        if (this.initialPrice < other.getInitialPrice()) {
            return -1;
        } else if (this.initialPrice > other.getInitialPrice())
{
            return 1;
        } else {    // if prices and titles were equal return 0
            return 0;
        }
    }
}
```

AudioItems.java:

```
public abstract class AudioItems extends Item {
    private String artist;
    private int releaseYear;

    public AudioItems(String title, String artist, double
initialPrice, int releaseYear) {
        super(title, initialPrice);
        this.artist = artist;
        this.releaseYear = releaseYear;
    }

    public int getReleaseYear() {
        return releaseYear;
    }

    public String artist() {
        return artist;
    }

    public String toString() {
        return super.toString() + " (" + artist + ")\tCost: $" +
super.getInitialPrice();
    }
}
```

Dvd.java:

```
public class Dvd extends Item{

    public Dvd(String title, double price) {
        super(title, price);
    }

    public double getSellingPrice() {
        return super.getInitialPrice();
    }

    public String toString() {
        return super.toString() + "\tCost: $" +
getSellingPrice();
    }
}
```

Record.java:

```
public class Record extends AudioItems {

    public Record(String title, String artist, double
initialPrice, int releaseYear) {
        super(title, artist, initialPrice, releaseYear);
    }

    public double getSellingPrice() {
       return (super.getInitialPrice()) * ((2024 -
super.getReleaseYear()) / 4.0);
    }
}
```

Cassette.java:

```
public class Cassette extends AudioItems {

    public Cassette(String title, String artist, double
initialPrice, int releaseYear) {
        super(title, artist, initialPrice, releaseYear);
    }

    public double getSellingPrice() {
        return (super.getInitialPrice()) +
(super.getInitialPrice()) / ((2024 - super.getReleaseYear()) /
6.0);
    }
}
```

Catalogue.java:

```
import java.util.ArrayList;

public class Catalogue {
    private double storeValue;
    private ArrayList<Item> items;

    public Catalogue(double storeValue) {
        this.storeValue = storeValue;
        items = new ArrayList<Item>();
    }

    public boolean sellItem(Item i) {
        if (searchItemBinary(i) != -1) {
            storeValue += i.getSellingPrice();
            items.remove(i);
            return true;
        } else {
            return false;
        }
    }

    public boolean buyItem(Item i) {
        if (storeValue >= i.getInitialPrice()) {
            storeValue -= i.getInitialPrice();
```

```
            items.add(i);
            return true;
        } else {
            return false;
        }
    }

    public int searchItemLinear(Item i) {
        int index = 0;
        for (Item item : items) {
            if (item.compareTo(i) == 0) {
                return index;
            }
            index++;
        }
        return -1;
    }

    public String printCatalogue()a {
        String catalogue = "";
        for (Item item: items) {
            catalogue += item + "\n";
        }
        return catalogue;
    }

    public void sortItem(ArrayList<Item> itemsCopy) {
            for(int outer = 0; outer < itemsCopy.size() - 1;
outer++) {
                int min = outer;
                for(int inner = outer + 1; inner <
itemsCopy.size(); inner++) {

if(itemsCopy.get(min).compareTo(itemsCopy.get(inner)) > 0) {
                        min = inner;
                    }
                }

                Item holder = itemsCopy.get(outer);    // Store
the current element at 'outer'
                itemsCopy.set(outer, itemsCopy.get(min)); // Set
the minimum item to the 'outer' position
                itemsCopy.set(min, holder);              //
Place the 'outer' element in the 'min' position
```

```
            }
    }

    public int searchItemBinary(Item i) {
        ArrayList<Item> itemsCopy = items;
        sortItem(itemsCopy);

        int start = 0;
        int end = itemsCopy.size()-1;

        while(start <= end) {
            int middle = (start+end)/2;
            int difference = itemsCopy.get(middle).compareTo(i);

            if (difference == 0) {
                return middle;
            }
            if (difference < 0) {
                start = middle + 1;
            }
            if (difference > 0) {
                end = middle - 1;
            }
        }
        return -1;
    }



}
```

Driver:

```
public class Driver {
    public static void main(String[] args) {
        Record record1 = new Record("Record1", "A", 120, 2022);
        AudioItems record2 = new Record("Record2", "B", 150,
2024);
        Cassette cassette1 = new Cassette("Record1", "C", 200,
2000);
        Dvd dvd1 = new Dvd("Dvd1", 50);
        Item dvd2 = new Dvd("Dvd1", 60);
```

```
        Cassette cassette2 = new Cassette("Cassette2", "D", 100,
2000);

        Catalogue catalogue1 = new Catalogue(1200);
        Catalogue catalogue2 = new Catalogue(0);

        // TEST CASE 1: Add 5 items to Catalogue
        catalogue1.buyItem(record1);
        catalogue1.buyItem(record2);
        catalogue1.buyItem(cassette1);
        catalogue1.buyItem(dvd1);
        catalogue1.buyItem(dvd2);

        // TEST CASE 2: Remove items from Catalogue until it is
empty
        catalogue1.sellItem(record1);
        catalogue1.sellItem(record2);
        catalogue1.sellItem(cassette1);
        catalogue1.sellItem(dvd1);
        catalogue1.sellItem(dvd2);

        // TEST CASE 3: Remove an item which is not in the
catalogue
        catalogue1.sellItem(cassette2);

        // TEST CASE 4: Add an item to catalogue when store does
not have enough money to buy it
        catalogue2.buyItem(cassette2);

        catalogue1.buyItem(record1);
        catalogue1.buyItem(record2);
        catalogue1.buyItem(cassette1);
        catalogue1.buyItem(dvd1);
        catalogue1.buyItem(dvd2);

        // TEST CASE 5: Print the catalogue
        System.out.println(catalogue1.printCatalogue());
        System.out.println(catalogue2.printCatalogue());
    }
}
```

Output:

```
~/O/CS-XXXX/CS1083/Assignments/As3  main !1 ?2                ✔  10:21:57 am
 java Driver
Record1 (A)     Cost: $120.0
Record2 (B)     Cost: $150.0
Record1 (C)     Cost: $200.0
Dvd1    Cost: $50.0
Dvd1    Cost: $60.0

~/O/CS-XXXX/CS1083/Assignments/As3  main !1 ?2                ✔  10:21:59 am
```