Báo cáo thực hành OOP LAB 4

Họ và tên: Phạm Tùng Dương

MSSV: 20225825

Mã lớp: 744520

1. **Import the existing project into the workspace of Eclipse:**
2. **Additional requirements of AIMS:**
3. **Creating the Book class:**

- Tạo lớp Book:

```
- package hust.soict.hedspi.media;

  import java.util.ArrayList;
  import java.util.List;

  public class Book {
      private int id;
      private String title;
      private String category;
      private float cost;
      private list <String> authors = new ArrayList<String>();

      public Book{

      }
      public Book(int id, String title, String category, float cost,
  List<String> authors){
          this.id = id;
          this.title = title;
          this.category = category;
          this.cost = cost;
          this.authors = authors;
      }
  }
```

- **Thêm các phương thức getter và setter trừ authors.**

- **Thêm phương thức addAuthor() và removeAuthor():**

```java
//Phuong thuc them tac gia
public void addAuthor(String authorName){  1 usage
    if(!authors.contains(authorName)){
        authors.add(authorName);
        System.out.println("Author " + authorName + " added.");
    }else{
        System.out.println("Author " + authorName + " is already in the list.");
    }
}
//Phuong thuc xoa tac gia
public void removeAuthor(String authorName){  no usages
    if(authors.contains(authorName)){
        authors.remove(authorName);
        System.out.println("Author " + authorName + " removed.");
    }else{
        System.out.println("Author " + authorName + " not found in the list.");
    }
}
```

4. Creating the abstract Media class:

```java
public abstract class Media {  4 inheritors

    private int id;  3 usages
    private String title;  5 usages
    private String category;  3 usages
    private float cost;  3 usages

    public static void main(String[] args) {
    }
```

- **Thêm các phương thức getter và setter.**
- **Loại bỏ các thuộc tính và phương thức của lớp Book và DigitalVideoDisc**
- **Cho hai lớp Book và DigitalVideoDisc kế thừa lớp Media:**

```
- public class Book extends Media{
- public class DigitalVideoDisc extends Media{
```

5. Creating the CompactDisc class:

5.1 Create the Disc class extending the Media class

```java
package hust.soict.hedspi.media;

public class Disc extends Media{ 2 inheritors
    private int length; 4 usages
    private String director; 4 usages

    public Disc(int length, String director) { no usages
        this.length = length;
        this.director = director;
    }
    public Disc(int id, String title, String category, float cost, int length, String director) { 2 usages
        super(id, title, category, cost);
        this.length = length;
        this.director = director;
    }

    public Disc() { 1 usage

    }

    public int getLength() { return length; }
    public void setLength(int length) { this.length = length; }
    public String getDirector() { return director; }
    public void setDirector(String director) { this.director = director; }


}
```

5.2 Create the `Track` class which models a track on a compact disc and will store information incuding the `title` and `length` of the track:

```java
package hust.soict.hedspi.media;

public class Track implements Playable { 24 usages
    private String title; 4 usages
    private int length; 4 usages

    // Constructor to initialize fields
    public Track(String title, int length) { 9 usages
        this.title = title;
        this.length = length;
    }

    // Getter for title
    public String getTitle() { return title; }

    // Getter for length
    public int getLength() { return length; }
```

5.3 Open the `CompactDisc` class:

o Thêm hai thuộc tính cho lớp: artist, và một ArrayList kiểu Track để lưu danh sách các bài nhạc.

o Đặt các thuộc tính này là private và tạo phương thức getter cho thuộc tính artist.

o Tạo constructor, sử dụng super()

o Tạo hai phương thức addTrack(), removeTrack()

o Tạo phương thức getLength()

```java
public class CompactDisc extends Disc implements Playable{
    private String artist;  2 usages
    private ArrayList<Track> tracks = new ArrayList<~>();  6 usages
    public CompactDisc(String cdTitle, String cdCategory, String cdArtist, float cdCost) { super(); }
    public CompactDisc(int id, String title, String category, float cost, int length, String director, String artist)
        super(id, title, category, cost, length, director);

        this.artist = artist;
    }
    public String getArtist() { return artist; }


    // Phương thức thêm track vào danh sách
    public void addTrack(Track track) {  9 usages
        if (tracks.contains(track)) {
            System.out.println("Track " + track.getTitle() + " is already in the list.");
        } else {
            tracks.add(track);
            System.out.println("Track " + track.getTitle() + " added to the list.");
        }
    }

    // Phương thức xóa track khỏi danh sách
    public void removeTrack(Track track) {  no usages
        if (tracks.contains(track)) {
            tracks.remove(track);
            System.out.println("Track " + track.getTitle() + " removed from the list.");
        } else {
            System.out.println("Track " + track.getTitle() + " is not in the list.");
        }
    }

    public int getLength() { return this.getLength(); }
```

6. Creating the Playable interface:

Tạo interface Playable:

```java
package hust.soict.hedspi.media;

public interface Playable {  3 usages  3 implementations
    public void play();  7 usages  3 implementations
}
```

- **Triển khai interface Playable cho các lớp CompactDisc, DigitalVideoDisc và Track:**

public class CompactDisc extends Disc implements Playable{

```java
@Override  6 usages
public void play() {
    System.out.println("Playing CD: " + this.getTitle());
    System.out.println("Track length: " + this.getLength());
}
```

public class DigitalVideoDisc extends Disc implements Playable {

```java
@Override  7 usages
public void play() {
    System.out.println("Playing DVD: " + this.getTitle());
    System.out.println("DVD length: " + this.getLength());
}
```

public class Track implements Playable {

```java
public void play() {  7 usages
    System.out.println("Playing track: " + this.getTitle());
    System.out.println("Track length: " + this.getLength());
}
```

7. Update the Cart class to work with Media:

Thêm các phương thức addMedia(), removeMedia() để quản lý DVD, Book và CompactDisc:

```
public void addMedia(Media media) {  2 usages  new *
    if (!itemsOrdered.contains(media)) {
        itemsOrdered.add(media);
        System.out.println(media.getTitle() + " has been added to the cart.");
    } else {
        System.out.println(media.getTitle() + " is already in the cart.");
    }
}

public void removeMedia(Media media) {  1 usage  new *
    if (itemsOrdered.contains(media)) {
        itemsOrdered.remove(media);
        System.out.println(media.getTitle() + " has been removed from the cart.");
    } else {
        System.out.println(media.getTitle() + " is not in the cart.");
    }
}
```

8. Update the Store class to work with Media:

Tương tự như lớp Cart, thay đổi thuộc tính itemsInStore[] của lớp Store thành kiểu ArrayList.

Thay thế các phương thức addDigitalVideoDisc() và removeDigitalVideoDisc() bằng các phương thức addMedia() và removeMedia().

```java
public static void addMedia(Media media) {  13 usages   nubiz24 *
    boolean existed = false;
    for (Media item : itemsInStore) {
        if (item.getTitle().equals(media.getTitle())) {
            existed = true;
            break;
        }
    }

    if (!existed) {
        itemsInStore.add(media);
        System.out.println("The media has been added in Store.");
    } else {
        System.out.println("The media is already in the store.");
    }
}
// Phương thức xóa DVD khỏi cửa hàng
public void removeMedia(Media media) {  1 usage   new *
    boolean existed = false;
    for (Media item : itemsInStore) {
        if (item.getTitle().equals(media.getTitle())) {
            itemsInStore.remove(item);
            System.out.println("The media has been removed from Store.");
            existed = true;
            break;
        }
    }

    if (!existed) {
        System.out.println("The media is not in the store.");
    }
}
```

9. Constructor of whole classes and parent classes:

10. Unique item in a list:

Thêm phương thức equals cho lớp Media:

```java
@Override  1 override
public boolean equals(Object obj) {
    if (this == obj) return true; // Kiểm tra tham chiếu
    if (obj == null || getClass() != obj.getClass()) return false; // Kiểm tra kiểu đối tượng

    Media media = (Media) obj; // Ép kiểu
    return this.title.equals(media.title); // So sánh thuộc tính title
}
```

Thêm phương thức equals cho lớp Track:

```java
    @Override
    public boolean equals(Object obj) {
        if (this == obj) return true; // Kiểm tra tham chiếu
        if (obj == null || getClass() != obj.getClass()) return false; // Kiểm tra kiểu đối tượng

        Track track = (Track) obj; // Ép kiểu
        return this.title.equals(track.title) && this.length == track.length; // So sánh title và length
    }
}
```

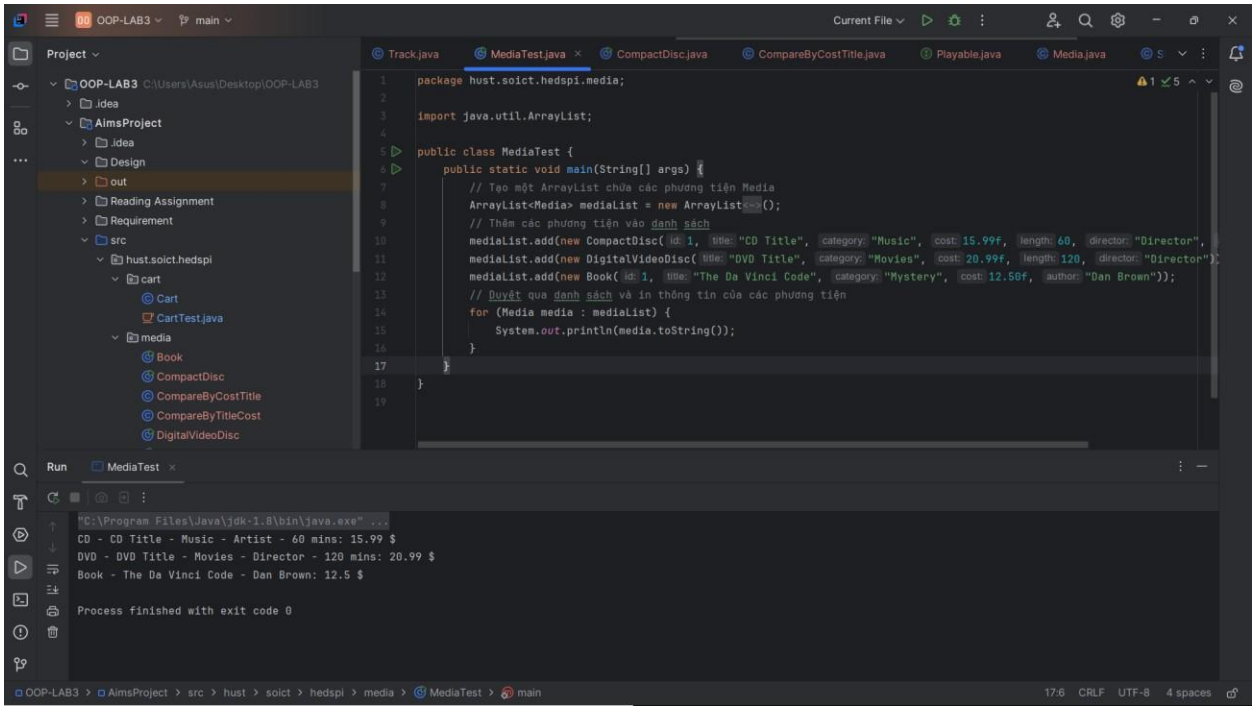
<u>Question</u>: If the passing object is not an instance of Media, what happens?

- Phép so sánh sẽ không được thực hiện nếu đối tượng truyền vào không phải là một instance của Media. Tuy nhiên, nếu đối tượng đó là instance của một lớp con kế thừa từ Media, thì phép so sánh vẫn có thể được thực hiện.

11. Polymorphism with toString() method:

Tạo arrayList của Media, thêm một số sản phẩm (DVD, Book, CD) vào danh sách sau đó duyệt qua từng sản phẩm, sử dụng toString() in ra thông tin của mỗi sản phẩm.

Test trong lớp MediaTest:



Kết luận: Mặc dù các lớp CD, DVD, và Book đều kế thừa từ lớp cha Media và thừa hưởng phương thức toString(), mỗi lớp con có thể triển khai phương thức này theo cách riêng. Khi duyệt qua danh sách sản phẩm và gọi phương thức toString() trên từng đối tượng, phương thức được gọi sẽ là phiên bản cụ thể của lớp tương ứng. Vì vậy, kết quả hiển thị của từng sản phẩm sẽ khác nhau, tùy thuộc vào cách triển khai toString() trong lớp con của nó.

12. Sort media in the Cart:

Lớp sắp xếp MediaComparatorByTitleCost:

```java
package hust.soict.hedspi.media;

import java.util.Comparator;

public class MediaComparatorByTitleCost implements Comparator<Media> {  1 usage

    @Override
    public int compare(Media media1, Media media2) {
        int titleCompare = media1.getTitle().compareTo(media2.getTitle());
        if (titleCompare != 0) {
            return titleCompare;
        }
        return Float.compare(media2.getCost(), media1.getCost()); // Sắp xếp theo giá giảm dần nếu tiêu đề giống nhau
    }
}
```

Lớp sắp xếp MediaComparatorByCostTitle:

```java
package hust.soict.hedspi.media;

import java.util.Comparator;

public class MediaComparatorByCostTitle implements Comparator<Media> {  1 usage

    @Override
    public int compare(Media media1, Media media2) {
        int costCompare = Float.compare(media2.getCost(), media1.getCost()); // Sắp xếp theo giá giảm dần
        if (costCompare != 0) {
            return costCompare;
        }
        return media1.getTitle().compareTo(media2.getTitle()); // Nếu giá giống nhau, sắp xếp theo tiêu đề
    }
}
```

Thêm các Comparator làm thuộc tính của lớp Media:

```
    // Comparator cho việc sắp xếp theo tiêu đề rồi đến giá
    public static final Comparator<Media> COMPARE_BY_TITLE_COST =  1 usage
            new MediaComparatorByTitleCost();

    // Comparator cho việc sắp xếp theo giá rồi đến tiêu đề
    public static final Comparator<Media> COMPARE_BY_COST_TITLE =  1 usage
            new MediaComparatorByCostTitle();

    // Phương thức compareTo để sắp xếp theo tiêu đề rồi đến giá
```

Truyền Comparator vào Collections.sort:

```
public void sortByTitle() {  1 usage  new *
  Collections.sort(itemsOrdered, Media.COMPARE_BY_TITLE_COST);
  Iterator<Media> iterator = itemsOrdered.iterator();

  while (iterator.hasNext()) {
    System.out.println((iterator.next()).toString());
  }
}

public void sortByCost() {  1 usage  new *
  Collections.sort(itemsOrdered, Media.COMPARE_BY_COST_TITLE);
  Iterator<Media> iterator = itemsOrdered.iterator();

  while (iterator.hasNext()) System.out.println((iterator.next()).toString());

}
```

QUESTION

a. What class should implement the Comparable interface?

- Lớp chứa đối tượng cần so sánh, chẳng hạn Media hoặc các lớp con của nó như DigitalVideoDisc, Book, CompactDisc.

b. In those classes, how should you implement the compareTo() method be to reflect the ordering that we want?

- Triển khai phương thức compareTo trong lớp Media:

```
    public int compareTo(Media other) {  1 usage  1 override
        int titleCompare = this.getTitle().compareTo(other.getTitle());
        if (titleCompare != 0) {
            return titleCompare;
        }
        return Float.compare(other.getCost(), this.getCost()); // Sắp xếp theo giá giảm dần nếu tiêu đề giống nhau
    }
}
```

c. Can we have two ordering rules of the item (by title then cost and by cost then title) if we use this Comparable interface approach?

- Nếu muốn sắp xếp một danh sách theo hai tiêu chí khác nhau, như "theo tên rồi đến giá" hoặc "theo giá rồi đến tên", thì chỉ sử dụng giao diện Comparable là không đủ, vì nó chỉ cho phép định nghĩa một quy tắc sắp xếp mặc định. Để linh hoạt hơn, nên sử dụng Comparator.
Comparator cho phép tạo nhiều quy tắc sắp xếp tùy ý mà không cần chỉnh sửa lớp gốc. Với cách này, có thể định nghĩa một bộ so sánh sắp xếp theo tên trước, một bộ so sánh khác sắp xếp theo giá trước, và sử dụng chúng tùy vào nhu cầu.

d. Suppose the DVDs has a different ordering rule from the other media types, that is by title, then decreasing length, then cost. How would you modify your code to allow this?

- Override phương thức compareTo() trong lớp DigitalVideoDisc:

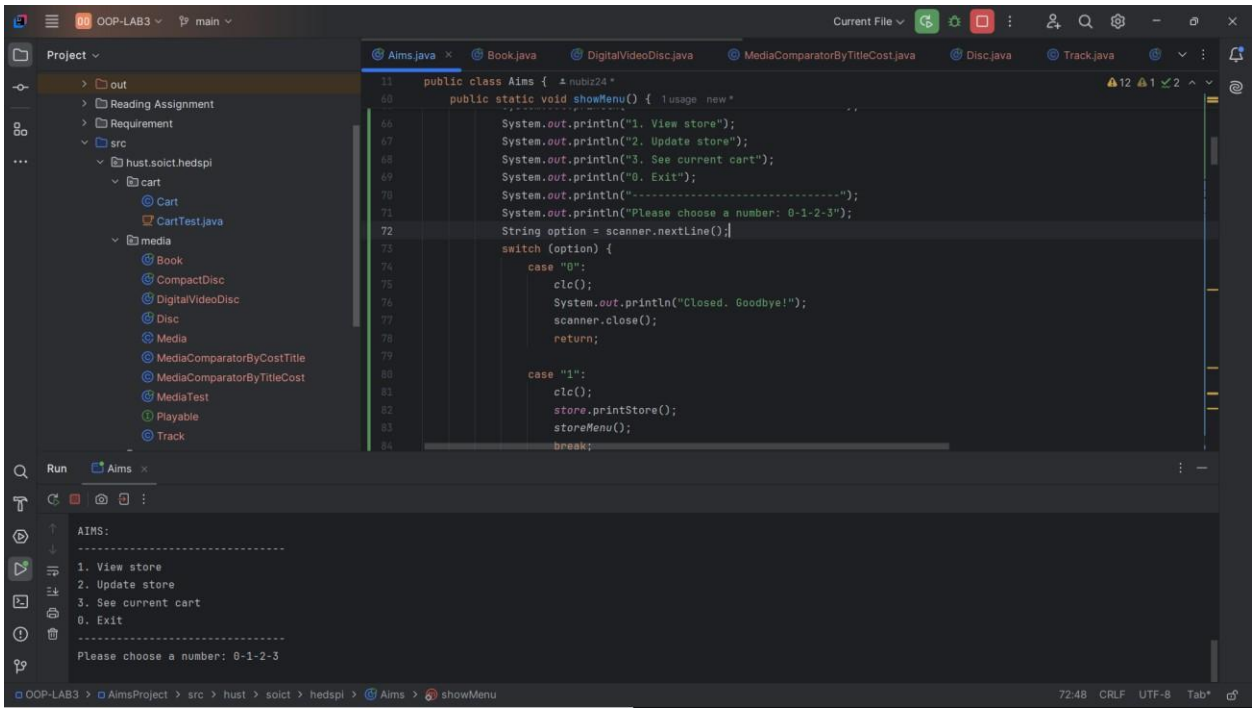
```
    @Override  1 usage
    public int compareTo(Media other) {
        if (other instanceof DigitalVideoDisc) {
            DigitalVideoDisc dvd = (DigitalVideoDisc) other;
            int titleCompare = this.getTitle().compareTo(dvd.getTitle());
            if (titleCompare != 0) {
                return titleCompare;
            }
            int lengthCompare = Integer.compare(dvd.getLength(), this.getLength()); // Sắp xếp theo chiều dài giảm dần
            if (lengthCompare != 0) {
                return lengthCompare;
            }
            return Float.compare(dvd.getCost(), this.getCost()); // Sắp xếp theo giá giảm dần
        }
        return super.compareTo(other); // Nếu không phải DVD, sử dụng phương thức compareTo trong Media
    }
}
```

13. Create a complete console application in the Aims class:

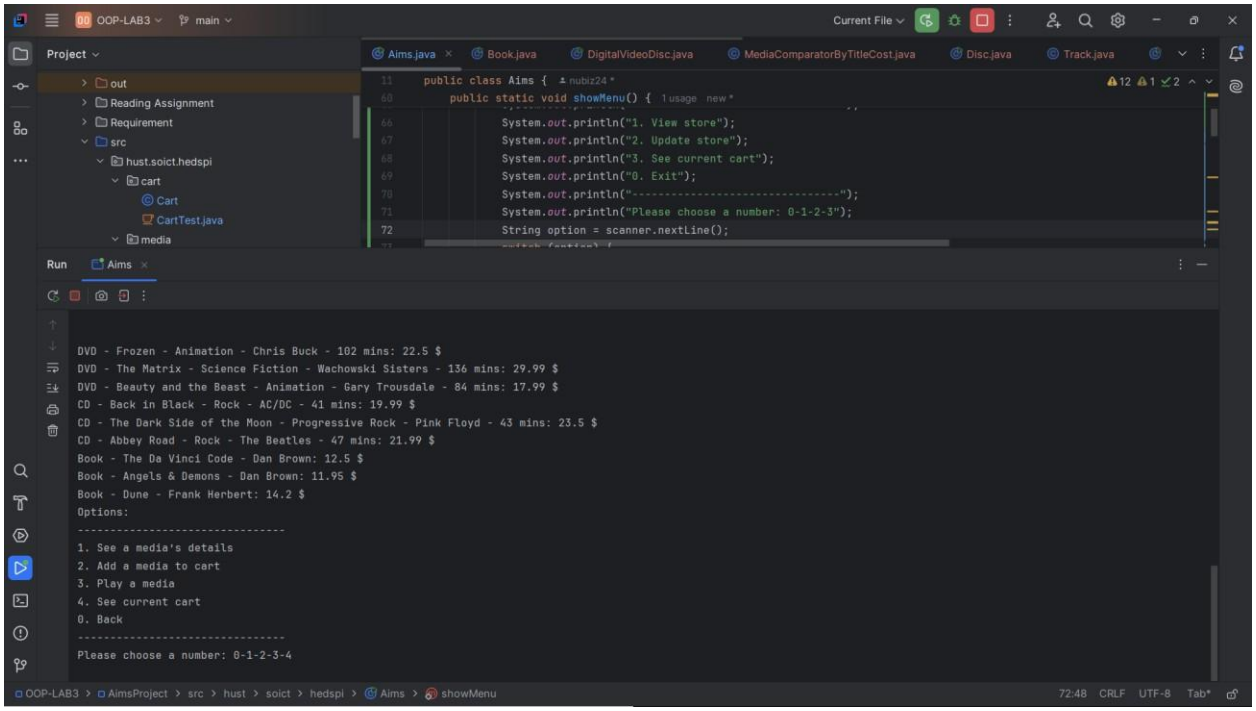
Hàm storeSetup() để thêm các mặt hàng vào Store:

```java
public static void storeSetup() {  1 usage  new *
    // DVD
    DigitalVideoDisc dvd1 = new DigitalVideoDisc( title: "Frozen", category: "Animation", cost: 22.50f, length: 102, director
    DigitalVideoDisc dvd2 = new DigitalVideoDisc( title: "The Matrix", category: "Science Fiction", cost: 29.99f, length: 13
    DigitalVideoDisc dvd3 = new DigitalVideoDisc( title: "Beauty and the Beast", category: "Animation", cost: 17.99f, length
    Store.addMedia(dvd1); Store.addMedia(dvd2); Store.addMedia(dvd3);

    // CD + Track
    CompactDisc cd1 = new CompactDisc( id: 4, title: "Back in Black", category: "Rock", cost: 19.99f, length: 41, director: "AC
    Track track1 = new Track( title: "Hells Bells", length: 312);
    Track track2 = new Track( title: "Back in Black", length: 255);
    Track track3 = new Track( title: "You Shook Me All Night Long", length: 232);
    cd1.addTrack(track1); cd1.addTrack(track2); cd1.addTrack(track3);

    CompactDisc cd2 = new CompactDisc( id: 5, title: "The Dark Side of the Moon", category: "Progressive Rock", cost: 23.50f
    Track track4 = new Track( title: "Speak to Me", length: 130);
    Track track5 = new Track( title: "Breathe", length: 212);
    cd2.addTrack(track4); cd2.addTrack(track5);

    CompactDisc cd3 = new CompactDisc( id: 6, title: "Abbey Road", category: "Rock", cost: 21.99f, length: 47, director: "The B
    Track track6 = new Track( title: "Come Together", length: 259);
    Track track7 = new Track( title: "Something", length: 182);
    Track track8 = new Track( title: "Here Comes the Sun", length: 185);
    cd3.addTrack(track6); cd3.addTrack(track7); cd3.addTrack(track8);

    Store.addMedia(cd1); Store.addMedia(cd2); Store.addMedia(cd3);

    // Book
    Book book1 = new Book( id: 7, title: "The Da Vinci Code", category: "Mystery", cost: 12.50f, author: "Dan Brown");
    Book book2 = new Book( id: 8, title: "Angels & Demons", category: "Thriller", cost: 11.95f, author: "Dan Brown");
    Book book3 = new Book( id: 9, title: "Dune", category: "Science Fiction", cost: 14.20f, author: "Frank Herbert");
    Store.addMedia(book1); Store.addMedia(book2); Store.addMedia(book3);
}
```

showMenu():



viewStore():



See a media's details:

```
Options:
--------------------------------
1. See a media's details
2. Add a media to cart
3. Play a media
4. See current cart
0. Back
--------------------------------
Please choose a number: 0-1-2-3-4
1
Enter the title of the media to view details (Press 0 to return): Frozen
Details: DVD - Frozen - Animation - Chris Buck - 102 mins: 22.5 $
Options:
--------------------------------
1. Add to cart
2. Play
0. Back
--------------------------------
Please choose a number: 0-1-2
```

```
Playing DVD: Frozen
DVD length: 102
Options:
--------------------------------
1. Add to cart
2. Play
0. Back
--------------------------------
Please choose a number: 0-1-2
```

Add a media to cart:

```
Enter the title of the media to view details (Press 0 to return): 0
Options:
--------------------------------
1. See a media's details
2. Add a media to cart
3. Play a media
4. See current cart
0. Back
--------------------------------
Please choose a number: 0-1-2-3-4
2
Enter the title of the media to add to cart (Press 0 to return): Dune
Dune has been added to the cart.
```

See current cart

```
**********CART**********
Ordered Items:
Book - Dune - Frank Herbert: 14.2 $
Total cost: 14.2$
**************************************************
Options:
--------------------------------
1. Filter medias in cart
2. Sort medias in cart
3. Remove media from cart
4. Play a media
5. Place order
0. Back
--------------------------------
Please choose a number: 0-1-2-3-4-5
```

Update store: