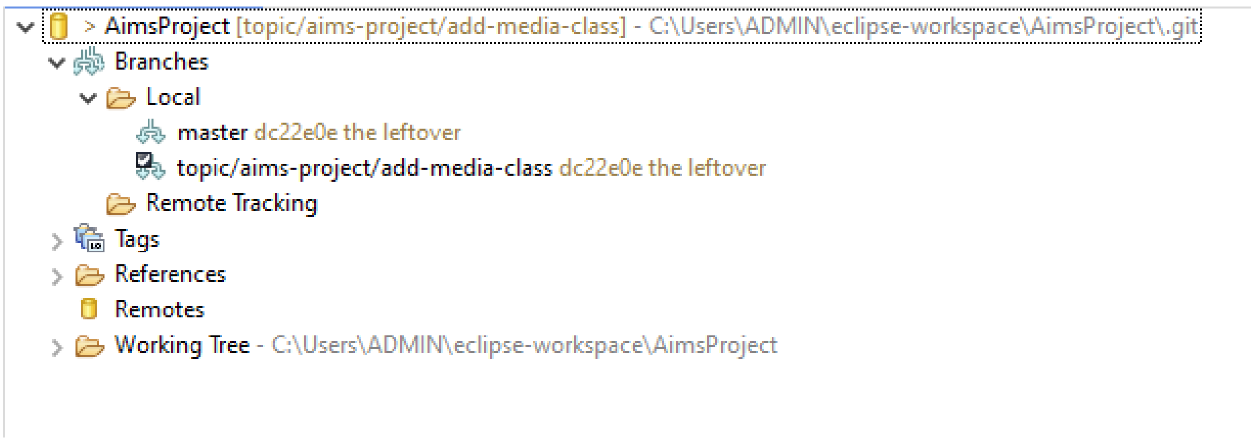
Report OOP Lab 04

Student: Lê Văn Pháp

Mssv: 20226118

0. branch topic


```
AimsProject [topic/aims-project/add-media-class] - C:\Users\ADMIN\eclipse-workspace\AimsProject\.git
  Branches
    Local
      master dc22e0e the leftover
      topic/aims-project/add-media-class dc22e0e the leftover
    Remote Tracking
  Tags
  References
  Remotes
  Working Tree - C:\Users\ADMIN\eclipse-workspace\AimsProject
```

1.Creating the Book class

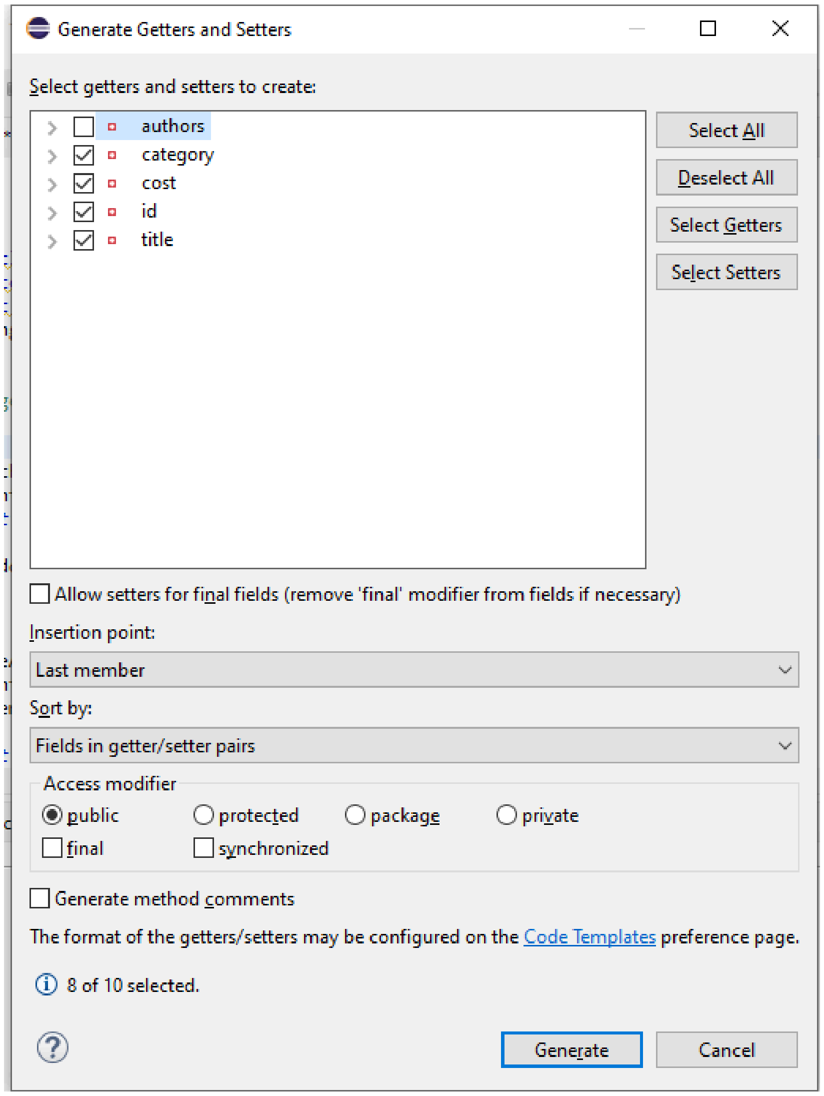
1.1 store the information about a Book with 5 fields, with public accessor methods for all but the authors field.

```
public class Book {

	private int id;
	private String title;
	private String category;
	private float cost;
	private List<String> authors = new ArrayList<String>();

	public Book() {
		// TODO Auto-generated constructor stub
	}

	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}
	public String getTitle() {
		return title;
	}
	public void setTitle(String title) {
		this.title = title;
	}
	public String getCategory() {
		return category;
	}
	public void setCategory(String category) {
		this.category = category;
	}
	public float getCost() {
		return cost;
	}
	public void setCost(float cost) {
		this.cost = cost;
	}
```
Using generate getter and setter option



Generate Getters and Setters

Select getters and setters to create:

- [ ] authors
- [x] category
- [x] cost
- [x] id
- [x] title

[ ] Allow setters for final fields (remove 'final' modifier from fields if necessary)

Insertion point: Last member

Sort by: Fields in getter/setter pairs

Access modifier

[✓ public] protected package private

[ ] final [ ] synchronized

[ ] Generate method comments

The format of the getters/setters may be configured on the Code Templates preference page.

8 of 10 selected.

1.2 create addAuthor(String authorName) and removeAuthor(String authorName) for the Book class

1.2.1 addAuthor

```
    public void addAuthor(String authorName) {
        if(authors.contains(authorName)) {
            System.out.println("already existed");
        } else {
            authors.add(authorName);
        }
    }
```

1.2.2 removeAuthor

```
    public void removeAuthor(String authorName) {
        if(authors.contains(authorName)) {
            authors.remove(authorName);
        } else {
            System.out.println(authorName + " not found");
        }
    }
```

1.2.3 Reference to some useful methods of the ArrayList class

```
    public void listAuthors() {
        if (authors.isEmpty()) {
            System.out.println("No authors available.");
        } else {
            System.out.println("Authors: " + String.join(", ", authors));
        }
    }
```

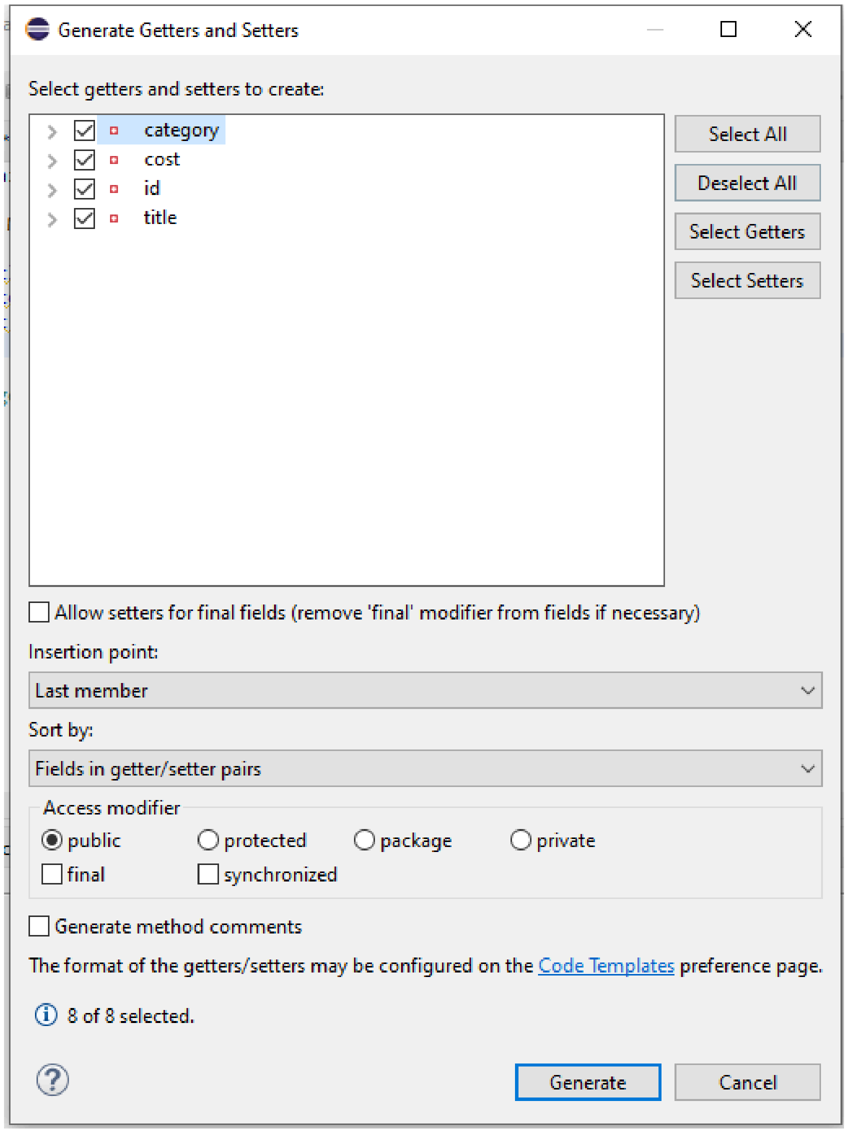
2. Creating the abstract Media class

2.1. Add fields and make public accessor methods for these fields

```
public abstract class Media {
    private int id;
    private String title;
    private String category;
    private float cost;

        public Media() {
                // TODO Auto-generated constructor stub
        }
        public int getId() {
                return id;
        }
        public void setId(int id) {
                this.id = id;
        }
        public String getTitle() {
                return title;
        }
        public void setTitle(String title) {
                this.title = title;
        }
        public String getCategory() {
                return category;
        }
        public void setCategory(String category) {
                this.category = category;
        }
        public float getCost() {
                return cost;
        }
        public void setCost(float cost) {
                this.cost = cost;
        }
```

Using generate getter and setter option



Select getters and setters to create:

- [✓] category
- [✓] cost
- [✓] id
- [✓] title

[ ] Allow setters for final fields (remove 'final' modifier from fields if necessary)

Insertion point: Last member

Sort by: Fields in getter/setter pairs

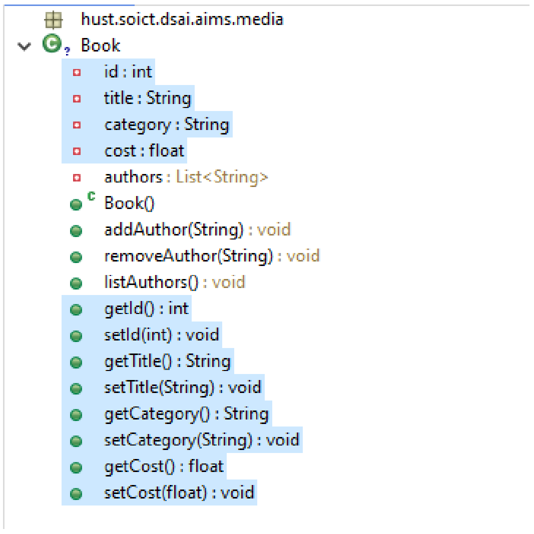
Access modifier
[✓ public] protected package private
[ ] final [ ] synchronized

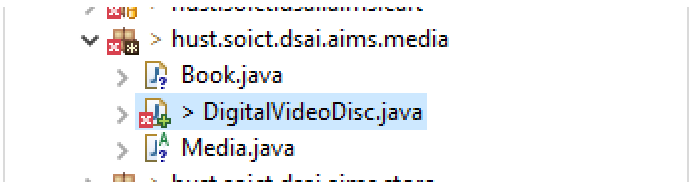
[ ] Generate method comments

The format of the getters/setters may be configured on the Code Templates preference page.

8 of 8 selected.

2.2 Remove fields and methods from Book and DigitalVideoDisc classes and move DigitalVideoDisc classes to the package


```
hust.soict.dsai.aims.media
Book
    id : int
    title : String
    category : String
    cost : float
    authors : List<String>
    Book()
    addAuthor(String) : void
    removeAuthor(String) : void
    listAuthors() : void
    getId() : int
    setId(int) : void
    getTitle() : String
    setTitle(String) : void
    getCategory() : String
    setCategory(String) : void
    getCost() : float
    setCost(float) : void
```


```
> hust.soict.dsai.aims.media
    Book.java
    > DigitalVideoDisc.java
    Media.java
```

2.3 Extend the Media class for both Book and DigitalVideoDisc

```
package hust.soict.dsai.aims.media;

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

public class Book extends Media {

            private List<String> authors = new ArrayList<String>();

            public Book(int id, String title, String category, float cost, List<String> authors) {
                        // TODO Auto-generated constructor stub
                        super(id, title, category, cost);
      this.authors = authors;
   }

   public void addAuthor(String authorName) {
      if(authors.contains(authorName)) {
         System.out.println("already existed");
      } else {
         authors.add(authorName);
      }
   }

   public void removeAuthor(String authorName) {
        if(authors.contains(authorName)) {
            authors.remove(authorName);
        } else {
            System.out.println(authorName + " not found");
        }
    }

    public void listAuthors() {
        if (authors.isEmpty()) {
            System.out.println("No authors available.");
        } else {
            System.out.println("Authors: " + String.join(", ", authors));
        }
    }
}
```

```
package hust.soict.dsai.aims.media;
public class DigitalVideoDisc extends Media {
	private String director;
	private int length;

   public DigitalVideoDisc(int id, String title, String category, float cost,
String director, int length) {
		super(id, title, category, cost);
	   this.director = director;
	   this.length = length;
   }

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

	public void setLength(int length) {
		this.length = length;
	}
}
```

3.Creating the CompactDisc class

```
package hust.soict.dsai.aims.media;
public class CompactDisc extends Media {
	public CompactDisc(int id, String title, String category, float cost) {
		// TODO Auto-generated constructor stub
```

```java
            super(id, title, category, cost);
    }
}
```

3.1. Create the Disc class extending the Media class

3.1.1 Disc class:

```java
package hust.soict.dsai.aims.media;
public class Disc extends Media{
   private String director;
   private float length;
   public Disc(int id, String title, String category, float cost, String director,
float length) {
       super(id, title, category, cost);
       this.director = director;
       this.length = length;
   }

   public Disc(int id, String title, String category, float cost) {
       super(id, title, category, cost);
   }

      public String getDirector() {
            return director;
      }
      public float getLength() {
            return length;
      }
}
```

3.1.2 Make the DigitalVideoDisc extending the Disc class

```java
package hust.soict.dsai.aims.media;
public class DigitalVideoDisc extends Disc {

   public DigitalVideoDisc(int id, String title, String category, float cost,
String director, float length) {
       super(id, title, category, cost, director, length);
   }

}
```

3.1.3 Create the CompactDisc extending the Disc class

```java
package hust.soict.dsai.aims.media;
public class CompactDisc extends Disc {
      public CompactDisc(int id, String title, String category, float cost) {
            // TODO Auto-generated constructor stub
            super(id, title, category, cost);
   }
}
```

3.2 Create the Track class

```java
package hust.soict.dsai.aims.media;
public class Track {
   private String title;
   private int length;
       public Track(String title, int length) {
             // TODO Auto-generated constructor stub
       this.title = title;
       this.length = length;
       }
       public String getTitle() {
             return title;
       }
       public int getLength() {
             return length;
       }
}
```

3.3 Open the CompactDisc class

```
package hust.soict.dsai.aims.media;
import java.util.ArrayList;
public class CompactDisc extends Disc {
   private String artist;
   private ArrayList<Track> tracks;

   public CompactDisc(int id, String title, String category, float cost,String
artist, ArrayList<Track> tracks) {
        super(id, title, category, cost);
        this.tracks = tracks;
        this.artist = artist;
        this.setLength(getLength());
   }
       public CompactDisc(int id, String title, String category, float cost) {
              // TODO Auto-generated constructor stub
              super(id, title, category, cost);
   }
       public String getArtist() {
              return artist;
       }

   public void addTrack(Track song) {
        if(tracks.contains(song)) {
            System.out.println(song.getTitle() + " already existed");
        } else {
            tracks.add(song);
        }
   }

   public void removeTrack(Track song) {
        if(tracks.contains(song)) {
            tracks.remove(song);
        } else {
            System.out.println(song.getTitle() + " not found");
        }
   }
```

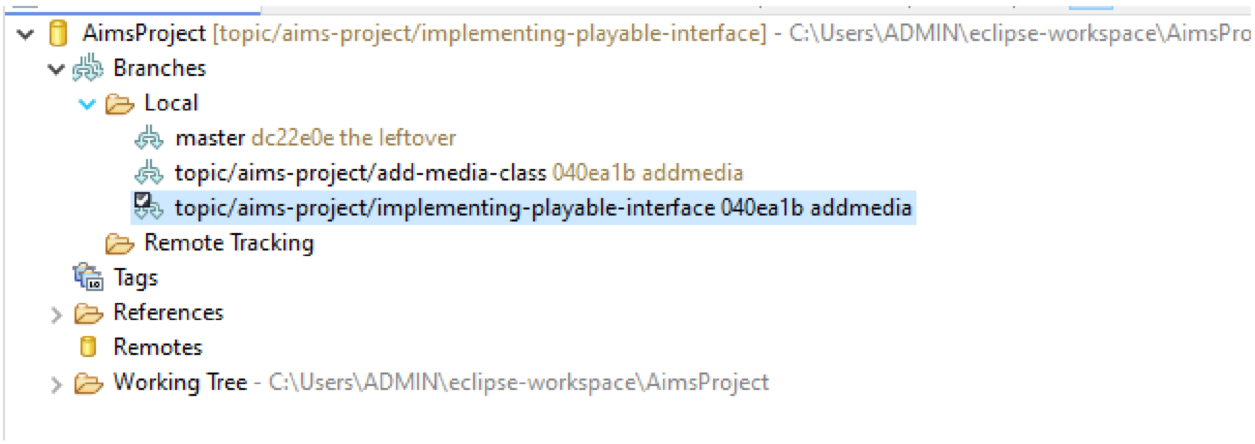
```
    @Override
    public float getLength() {
        float sum = 0;
        for(Track song : tracks) {
            sum += song.getLength();
        }
        return sum;
    }
}
```

add setLength in Disc class to get the length of the CD

```
    public void setLength(float length) {
        this.length = length;
    }
```

4.Create the Playable interface

create branch



```
AimsProject [topic/aims-project/implementing-playable-interface] - C:\Users\ADMIN\eclipse-workspace\AimsPro
  Branches
    Local
      master dc22e0e the leftover
      topic/aims-project/add-media-class 040ea1b addmedia
      topic/aims-project/implementing-playable-interface 040ea1b addmedia
    Remote Tracking
  Tags
  References
  Remotes
  Working Tree - C:\Users\ADMIN\eclipse-workspace\AimsProject
```

4.1 Create Playable interface:

```
package hust.soict.dsai.aims.media;
public interface Playable {
    public void play();
}
```

4.2 Implement the Playable with CompactDisc, DigitalVideoDisc and Track

```
public class CompactDisc extends Disc implements Playable
```

```
public class DigitalVideoDisc extends Disc implements Playable
```

```
public class Track implements Playable
```

4.3 Implement play() for DigitalVideoDisc and Track

```
package hust.soict.dsai.aims.media;
```

```java
public class Track implements Playable{
   private String title;
   private int length;
       public Track(String title, int length) {
              // TODO Auto-generated constructor stub
        this.title = title;
        this.length = length;
       }
       public String getTitle() {
              return title;
       }
       public int getLength() {
              return length;
       }

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

```java
package hust.soict.dsai.aims.media;
public class DigitalVideoDisc extends Disc implements Playable {

   public DigitalVideoDisc(int id, String title, String category, float cost,
String director, float length) {
        super(id, title, category, cost, director, length);
   }

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

4.4 Implement play() for CompactDisc

```java
   public void play() {
        System.out.println("Playing CD: " + this.getTitle());
        System.out.println("Artist: " + this.getArtist());
        System.out.println("Total length: " + this.getLength() + " seconds");
        for (Track track : tracks) {
            track.play();
        }
   }
```

5.Update the Cart class to work with Media:

```java
package hust.soict.dsai.aims.cart;
import java.util.ArrayList;
import hust.soict.dsai.aims.media.Media;
public class Cart {
   private ArrayList<Media> itemsOrdered = new ArrayList<>();
```

```
    public void addMedia(Media item) {
        if(itemsOrdered.contains(item)) {
            System.out.println(item.getTitle() + " already existed");
        } else {
            itemsOrdered.add(item);
            System.out.println("done");
        }
    }
    public void removeMedia(Media item) {
        if (itemsOrdered.remove(item)) {
            System.out.println("The media has been removed.");
        } else {
            System.out.println("The media is not found in the cart.");
        }
    }
    public float totalCost() {
        float total = 0.0f;
        for (Media item : itemsOrdered) {
            total += item.getCost();
        }
        return total;
    }

    public void print() {
        StringBuilder output = new
StringBuilder("***********************CART***********************\n")
               .append("Ordered Items:\n");
        if(itemsOrdered.isEmpty()) {
            output.append("No time\n");
        } else {
            int i = 1;
            for (Media item : itemsOrdered) {
                output.append(i + ".[" + item.getTitle() + "] - [" +
item.getCategory() + "] - "
                        + item.getCost() + " $\n");
                i++;
            }
        }
    output.append("Total cost: ").append(totalCost()).append(" $\n")
        .append("***************************************************\n");
    System.out.println(output);
    }
}
```

6. Update the Store class to work with Media

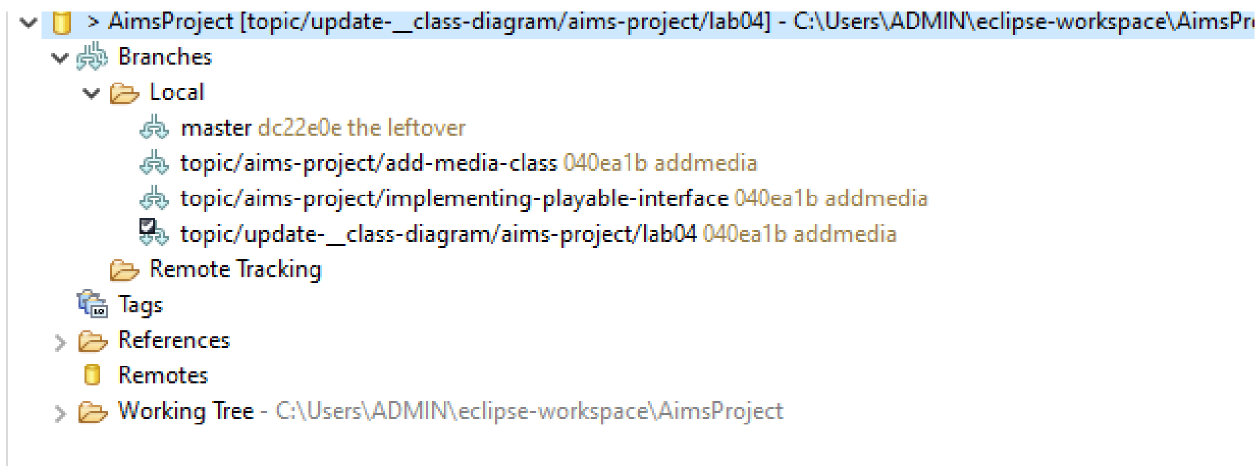
```
package hust.soict.dsai.aims.cart;
import java.util.ArrayList;
import hust.soict.dsai.aims.media.Media;
```

```java
public class Cart {
   private ArrayList<Media> itemsOrdered = new ArrayList<>();
   public void addMedia(Media item) {
       if(itemsOrdered.contains(item)) {
           System.out.println(item.getTitle() + " already existed");
       } else {
           itemsOrdered.add(item);
           System.out.println("done");
       }
   }
   public void removeMedia(Media item) {
       if (itemsOrdered.remove(item)) {
           System.out.println("The media has been removed.");
       } else {
           System.out.println("The media is not found in the cart.");
       }
   }
   public float totalCost() {
       float total = 0.0f;
       for (Media item : itemsOrdered) {
           total += item.getCost();
       }
       return total;
   }

   public void print() {
       StringBuilder output = new
StringBuilder("***********************CART***********************\n")
               .append("Ordered Items:\n");
       if(itemsOrdered.isEmpty()) {
           output.append("No time\n");
       } else {
           int i = 1;
           for (Media item : itemsOrdered) {
               output.append(i + ".[" + item.getTitle() + "] - [" +
item.getCategory() + "] - "
                       + item.getCost() + " $\n");
               i++;
           }
       }
   output.append("Total cost: ").append(totalCost()).append(" $\n")
       .append("***************************************************\n");
   System.out.println(output);
   }
}
```

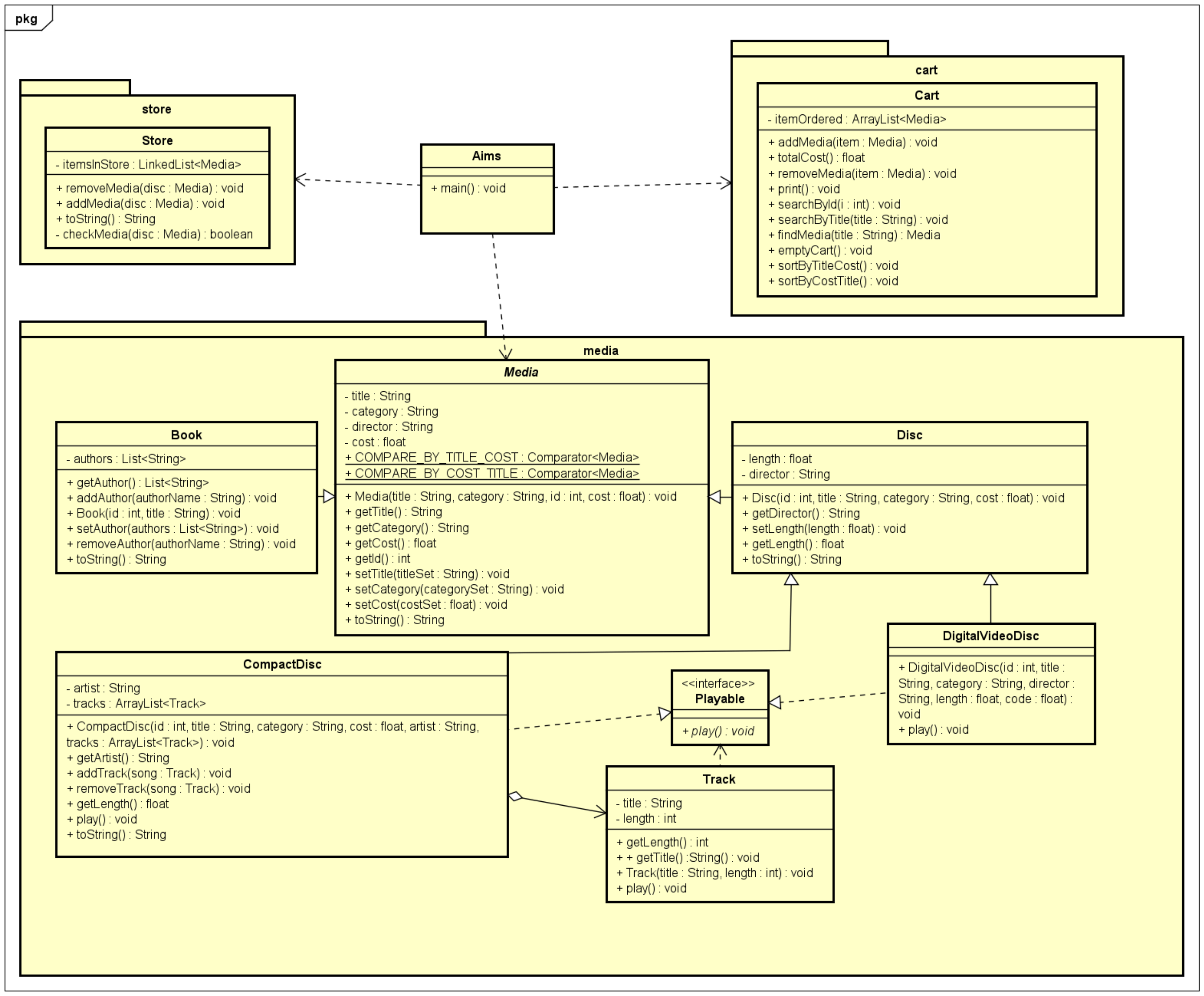
7. Constructors of whole classes and parent classes

create new branch



```
> AimsProject [topic/update-__class-diagram/aims-project/lab04] - C:\Users\ADMIN\eclipse-workspace\AimsPr
  Branches
    Local
      master dc22e0e the leftover
      topic/aims-project/add-media-class 040ea1b addmedia
      topic/aims-project/implementing-playable-interface 040ea1b addmedia
      topic/update-__class-diagram/aims-project/lab04 040ea1b addmedia
    Remote Tracking
  Tags
  References
  Remotes
  Working Tree - C:\Users\ADMIN\eclipse-workspace\AimsProject
```

Class diagram


pkg
store
Store
- itemsInStore : LinkedList<Media>
+ removeMedia(disc : Media) : void
+ addMedia(disc : Media) : void
+ toString() : String
- checkMedia(disc : Media) : boolean
Aims
+ main() : void
cart
Cart
- itemOrdered : ArrayList<Media>
+ addMedia(item : Media) : void
+ totalCost() : float
+ removeMedia(item : Media) : void
+ print() : void
+ searchById(i : int) : void
+ searchByTitle(title : String) : void
+ findMedia(title : String) : Media
+ emptyCart() : void
+ sortByTitleCost() : void
+ sortByCostTitle() : void
media
Media
- title : String
- category : String
- director : String
- cost : float
+ COMPARE_BY_TITLE_COST : Comparator<Media>
+ COMPARE_BY_COST_TITLE : Comparator<Media>
+ Media(title : String, category : String, id : int, cost : float) : void
+ getTitle() : String
+ getCategory() : String
+ getCost() : float
+ getId() : int
+ setTitle(titleSet : String) : void
+ setCategory(categorySet : String) : void
+ setCost(costSet : float) : void
+ toString() : String
Book
- authors : List<String>
+ getAuthor() : List<String>
+ addAuthor(authorName : String) : void
+ Book(id : int, title : String) : void
+ setAuthor(authors : List<String>) : void
+ removeAuthor(authorName : String) : void
+ toString() : String
Disc
- length : float
- director : String
+ Disc(id : int, title : String, category : String, cost : float) : void
+ getDirector() : String
+ setLength(length : float) : void
+ getLength() : float
+ toString() : String
DigitalVideoDisc
+ DigitalVideoDisc(id : int, title : String, category : String, director : String, length : float, code : float) : void
+ play() : void
CompactDisc
- artist : String
- tracks : ArrayList<Track>
+ CompactDisc(id : int, title : String, category : String, cost : float, artist : String, tracks : ArrayList<Track>) : void
+ getArtist() : String
+ addTrack(song : Track) : void
+ removeTrack(song : Track) : void
+ getLength() : float
+ play() : void
+ toString() : String
<<interface>>
Playable
+ play() : void
Track
- title : String
- length : int
+ getLength() : int
+ + getTitle() :String() : void
+ Track(title : String, length : int) : void
+ play() : void

Question: - Which classes are aggregates of other classes? Checking all constructors of whole classes if they initialize for their parts?

Ans:

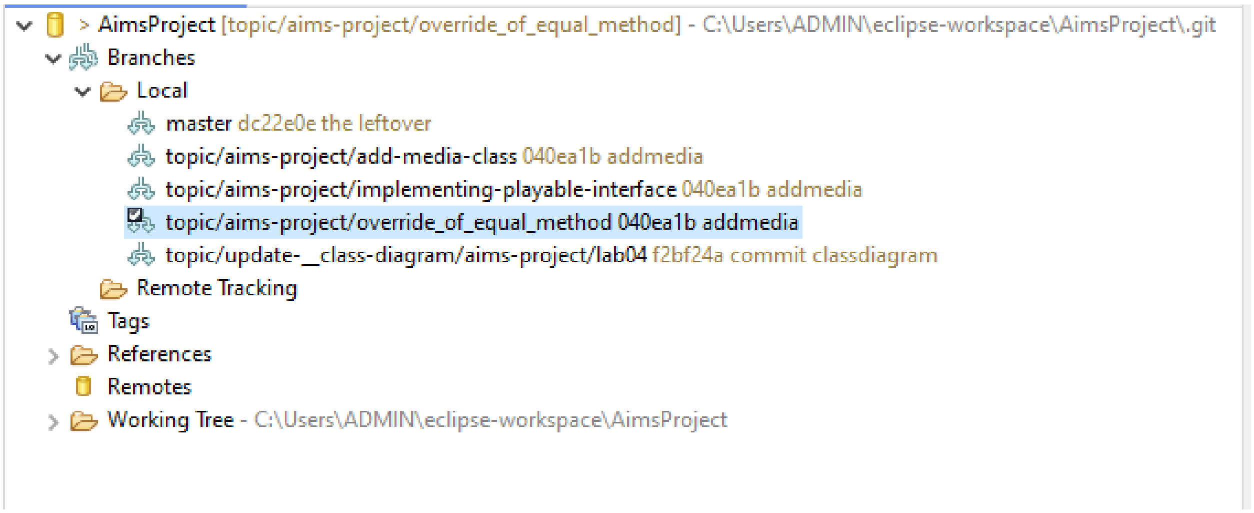
In object-oriented design, aggregates are classes that manage collections of other objects. In your case, the Cart class aggregates Media objects (e.g., Book, CompactDisc, DigitalVideoDisc) through the itemsOrdered list, while the Book class aggregates a list of authors managed by specific methods. Similarly, the CompactDisc class aggregates Track objects via a tracks list, with methods for managing them. In contrast, DigitalVideoDisc and Disc are individual items, not aggregates. Thus, Cart and Book are true aggregates as they hold and manage references to other objects.

question: - Write constructors for parent and child classes. Remove redundant setter methods if any

Ans: no redundant setter method

8. Unique item in a list

branch override



```
> AimsProject [topic/aims-project/override_of_equal_method] - C:\Users\ADMIN\eclipse-workspace\AimsProject\.git
  Branches
    Local
      master dc22e0e the leftover
      topic/aims-project/add-media-class 040ea1b addmedia
      topic/aims-project/implementing-playable-interface 040ea1b addmedia
      topic/aims-project/override_of_equal_method 040ea1b addmedia
      topic/update-__class-diagram/aims-project/lab04 f2bf24a commit classdiagram
    Remote Tracking
  Tags
  References
  Remotes
  Working Tree - C:\Users\ADMIN\eclipse-workspace\AimsProject
```

8.1 Override the equals() method of the Media class

```
    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Media media = (Media) o;
        return title.equals(media.title);
    }
```

8.2 Override the equals() method of the Track class

```
    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Track track = (Track) o;
        return length == track.length && title.equals(track.title);
        }
```

8.3 QA

Q: When overriding the equals() method of the Object class, you will have to cast the Object

parameter obj to the type of Object that you are dealing with. For example, in the Media class, you

must cast the Object obj to a Media, and then check the equality of the two objects' attributes as the above requirements (i.e. title for Media; title and length for Track). If the passing object is

not an instance of Media, what happens?

A: If the passed object is not an instance of the Media class (or its subclasses) when overriding the equals() method, the method should return false. When overriding the equals() method in the Object class, it's essential to ensure that the parameter obj is of the correct type before proceeding with the equality check. This is typically done using the instanceof operator. Returning flase avoids attempting an unsafe cast, which would otherwise result in a ClassCastException. By checking the type, the equals() method adheres to its contract, ensuring robustness, type safety, and consistency.

9. Polymorphism with toString() method

9.1 Add toString() into Book class:

```
    @Override
    public String toString() {
        return "Book [Title=" + getTitle() + ", Category=" + getCategory() + ",
Cost=" + getCost() + "]";
    }
```

9.2 Add toString() into DigitalVideoDisc class:

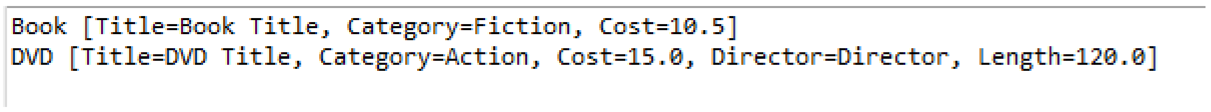
```
    @Override
    public String toString() {
        return "DVD [Title=" + getTitle() + ", Category=" + getCategory() +
               ", Cost=" + getCost() + ", Director=" + getDirector() +
               ", Length=" + getLength() + "]";
    }
```

9.3 Test the code in TestPolymorphism class:

```
package hust.soict.dsai.test.polymorphism;
import hust.soict.dsai.aims.media.*;
import java.util.ArrayList;
public class TestPolymorphism {
	public static void main(String[] args) {
		ArrayList<Media> mediaList = new ArrayList<>();
		mediaList.add(new Book(1, "Book Title", "Fiction", 10.5f, new
ArrayList<>()));
		mediaList.add(new DigitalVideoDisc(2, "DVD Title", "Action", 15.0f,
"Director", 120));
		for (Media media : mediaList) {
```

```
                  System.out.println(media.toString());
              }
         }
}
```

9.4 Result:



```
Book [Title=Book Title, Category=Fiction, Cost=10.5]
DVD [Title=DVD Title, Category=Action, Cost=15.0, Director=Director, Length=120.0]
```

Explanation:

When iterating through the mediaList and calling the toString() method on each object, the specific implementation of toString() executed depends on the runtime type of the object. This demonstrates polymorphism in Java. For example:

When the object is a Book, the overridden toString() in the Book class is executed, printing details specific to a Book.

When the object is a DigitalVideoDisc, its overridden toString() is called, displaying information about the DVD.

Polymorphism allows the same method (toString) to behave differently based on the actual object type, making the code more flexible and reusable.

10. Sort media in the cart

add this code to the Media class:

```
   public static final Comparator<Media> COMPARE_BY_TITLE_COST = new
MediaComparatorByTitleCost();
   public static final Comparator<Media> COMPARE_BY_COST_TITLE = new
MediaComparatorByCostTitle();
```

10.1 Create two classes of comparators

```
package hust.soict.dsai.aims.media;
import java.util.Comparator;
public class MediaComparatorByCostTitle implements Comparator<Media>{
   @Override
   public int compare(Media o1, Media o2) {
       if(o1.getCost() != o2.getCost()) {
           int i = (int) (o1.getCost() - o2.getCost());
           return i;
       } else {
           return o1.getTitle().compareTo(o2.getTitle());
       }
   }
}
```

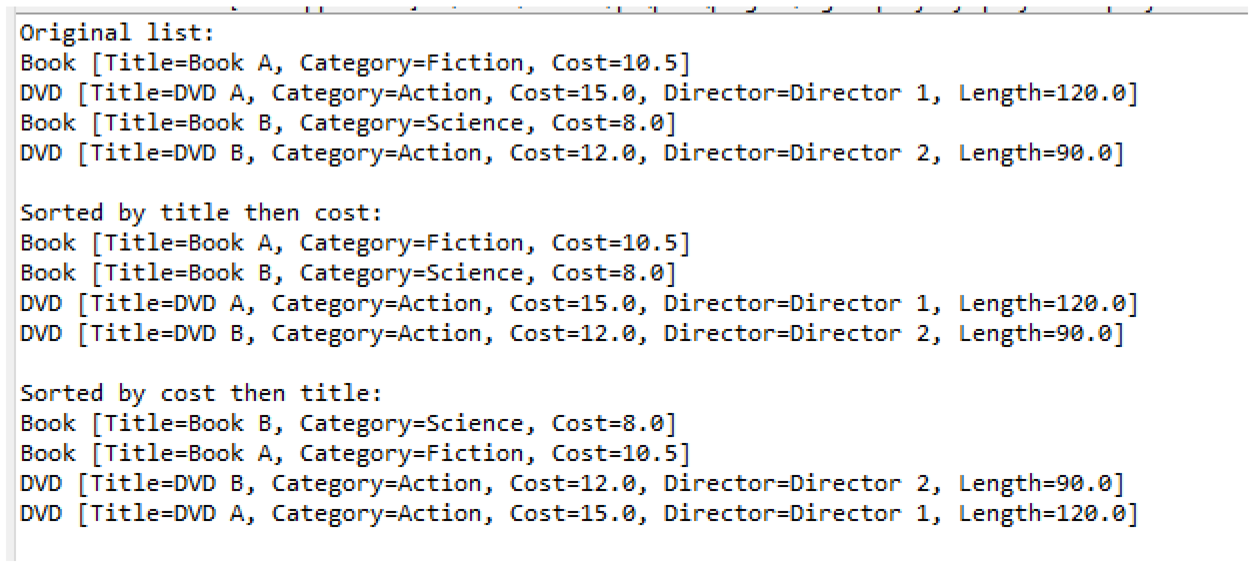
```
package hust.soict.dsai.aims.media;
import java.util.Comparator;
public class MediaComparatorByTitleCost implements Comparator<Media>{
   @Override
   public int compare(Media o1, Media o2) {
       if(o1.getTitle().compareTo(o2.getTitle()) != 0) {
           return o1.getTitle().compareTo(o2.getTitle());
       } else {
           if(o1.getCost() > o2.getCost()) return 1;
           else return -1;
       }
   }
}
```

10.2 Testing code:

```
package hust.soict.dsai.aims;

import hust.soict.dsai.aims.media.*;
import java.util.*;
public class Aims {
   public static void main(String[] args) {
       ArrayList<Media> mediaList = new ArrayList<>();
       mediaList.add(new Book(1, "Book A", "Fiction", 10.5f, new ArrayList<>()));
       mediaList.add(new DigitalVideoDisc(2, "DVD A", "Action", 15.0f, "Director
1", 120));
       mediaList.add(new Book(3, "Book B", "Science", 8.0f, new ArrayList<>()));
       mediaList.add(new DigitalVideoDisc(4, "DVD B", "Action", 12.0f, "Director
2", 90));
       System.out.println("Original list:");
       for (Media media : mediaList) {
           System.out.println(media);
       }
       Collections.sort(mediaList, Media.COMPARE_BY_TITLE_COST);
       System.out.println("\nSorted by title then cost:");
       for (Media media : mediaList) {
           System.out.println(media);
       }
       Collections.sort(mediaList, Media.COMPARE_BY_COST_TITLE);
       System.out.println("\nSorted by cost then title:");
       for (Media media : mediaList) {
           System.out.println(media);
       }
   }
}
```

Result:



```
Original list:
Book [Title=Book A, Category=Fiction, Cost=10.5]
DVD [Title=DVD A, Category=Action, Cost=15.0, Director=Director 1, Length=120.0]
Book [Title=Book B, Category=Science, Cost=8.0]
DVD [Title=DVD B, Category=Action, Cost=12.0, Director=Director 2, Length=90.0]

Sorted by title then cost:
Book [Title=Book A, Category=Fiction, Cost=10.5]
Book [Title=Book B, Category=Science, Cost=8.0]
DVD [Title=DVD A, Category=Action, Cost=15.0, Director=Director 1, Length=120.0]
DVD [Title=DVD B, Category=Action, Cost=12.0, Director=Director 2, Length=90.0]

Sorted by cost then title:
Book [Title=Book B, Category=Science, Cost=8.0]
Book [Title=Book A, Category=Fiction, Cost=10.5]
DVD [Title=DVD B, Category=Action, Cost=12.0, Director=Director 2, Length=90.0]
DVD [Title=DVD A, Category=Action, Cost=15.0, Director=Director 1, Length=120.0]
```

10.3 QA

1. What class should implement the Comparable interface?

The Media class should implement the Comparable interface since it is the parent class of all media types (Book, CompactDisc, DigitalVideoDisc) and is used for comparisons in the Cart.

2. How should you implement the compareTo() method to reflect the ordering that we want?

In the Media class, the compareTo() method should compare items first by title (alphabetically) and then by cost (numerically) if the titles are equal. This can be implemented as:

```
"@Override

public int compareTo(Media other) {

  int titleComparison = this.title.compareTo(other.title);

  if (titleComparison != 0) {

    return titleComparison;

  }

  return Float.compare(this.cost, other.cost);

}"
```

3. 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?

No, the Comparable interface supports only one natural ordering (defined by the compareTo() method). To support multiple ordering rules, we would need to use the Comparator interface, which allows defining different comparison logic.

4. Suppose the DVDs have a different ordering rule (by title, then decreasing length, then cost). How would you modify your code to allow this?

To handle a different ordering rule for DVDs, the DigitalVideoDisc class can override the compareTo() method inherited from Media and implement its specific logic, like this:

```
"@Override
public int compareTo(Media other) {
  if (other instanceof DigitalVideoDisc) {
    DigitalVideoDisc dvd = (DigitalVideoDisc) other;
    int titleComparison = this.title.compareTo(dvd.title);
    if (titleComparison != 0) {
      return titleComparison;
    }
    int lengthComparison = Float.compare(dvd.length, this.length); // Decreasing length
    if (lengthComparison != 0) {
      return lengthComparison;
    }
    return Float.compare(this.cost, dvd.cost);
  }
  return super.compareTo(other); // Default comparison for other media types
}"
```

11. Create a complete console application in the Aims class

I have run the code successfully

```
package hust.soict.dsai.aims;
import hust.soict.dsai.aims.cart.Cart;
import hust.soict.dsai.aims.media.*;
import hust.soict.dsai.aims.store.Store;
import java.util.ArrayList;
import java.util.Scanner;
public class Aims {
```

```java
    public static void main(String[] args) {
        Store store = new Store();
        Cart cart = new Cart();
        Scanner scanner = new Scanner(System.in);
        // Add sample items to store for testing
        store.addMedia(new Book(1, "Book A", "Fiction", 10.5f, new ArrayList<>()));
        store.addMedia(new DigitalVideoDisc(2, "DVD A", "Action", 15.0f, "Director
1", 120));
        store.addMedia(new DigitalVideoDisc(3, "DVD B", "Action", 12.0f, "Director
2", 90));
        store.addMedia(new Book(4, "Book B", "Science", 8.0f, new ArrayList<>()));
        int choice;
        while (true) {
            showMenu();
            choice = scanner.nextInt();
            scanner.nextLine(); // Consume the newline
            switch (choice) {
                case 1:
                    handleViewStore(store, cart, scanner);
                    break;
                case 2:
                    handleUpdateStore(store, scanner);
                    break;
                case 3:
                    handleSeeCart(cart, scanner);
                    break;
                case 0:
                    System.out.println("Exiting AIMS. Goodbye!");
                    scanner.close();
                    return;
                default:
                    System.out.println("Invalid choice. Please try again.");
            }
        }
    }
    public static void showMenu() {
        System.out.println("AIMS: ");
        System.out.println("--------------------------------");
        System.out.println("1. View store");
        System.out.println("2. Update store");
        System.out.println("3. See current cart");
        System.out.println("0. Exit");
        System.out.println("--------------------------------");
        System.out.println("Please choose a number: 0-1-2-3");
    }
    public static void storeMenu() {
        System.out.println("Options: ");
        System.out.println("--------------------------------");
        System.out.println("1. See a media’s details");
        System.out.println("2. Add a media to cart");
        System.out.println("3. Play a media");
        System.out.println("4. See current cart");
        System.out.println("0. Back");
        System.out.println("--------------------------------");
        System.out.println("Please choose a number: 0-1-2-3-4");
    }
    public static void cartMenu() {
        System.out.println("Options: ");
        System.out.println("--------------------------------");
        System.out.println("1. Filter medias in cart");
        System.out.println("2. Sort medias in cart");
        System.out.println("3. Remove media from cart");
        System.out.println("4. Play a media");
        System.out.println("5. Place order");
        System.out.println("0. Back");
        System.out.println("--------------------------------");
        System.out.println("Please choose a number: 0-1-2-3-4-5");
    }
    private static void handleViewStore(Store store, Cart cart, Scanner scanner) {
        System.out.println("\nStore Items:");
        store.print();
        int choice;
        do {
            storeMenu();
            choice = scanner.nextInt();
            scanner.nextLine();
            switch (choice) {
                case 1:
                    System.out.print("Enter media title: ");
                    String title = scanner.nextLine();
                    Media media = store.findMediaByTitle(title);
                    if (media != null) {
                        System.out.println(media.toString());
                        if (media instanceof Playable) {
                            System.out.println("1. Add to cart\n2. Play\n0. Back");
                            int option = scanner.nextInt();
                            scanner.nextLine();
                            if (option == 1) {
                                cart.addMedia(media);
                            } else if (option == 2) {
                                ((Playable) media).play();
                            }
                        } else {
                            System.out.println("1. Add to cart\n0. Back");
                            int option = scanner.nextInt();
                            scanner.nextLine();
                            if (option == 1) {
                                cart.addMedia(media);
                            }
                        }
                    } else {
                        System.out.println("Media not found.");
                    }
                    break;
                case 2:
                    System.out.print("Enter media title to add to cart: ");
                    title = scanner.nextLine();
                    media = store.findMediaByTitle(title);
                    if (media != null) {
                        cart.addMedia(media);
                    } else {
                        System.out.println("Media not found.");
                    }
                    break;
                case 3:
                    System.out.print("Enter media title to play: ");
                    title = scanner.nextLine();
                    media = store.findMediaByTitle(title);
                    if (media instanceof Playable) {
                        ((Playable) media).play();
                    } else {
                        System.out.println("Media cannot be played.");
                    }
                    break;
                case 4:
                    handleSeeCart(cart, scanner);
                    break;
                case 0:
                    break;
                default:
                    System.out.println("Invalid choice.");
            }
        } while (choice != 0);
    }
    private static void handleUpdateStore(Store store, Scanner scanner) {
        System.out.println("1. Add media\n2. Remove media\n0. Back");
        int choice = scanner.nextInt();
        scanner.nextLine();
        if (choice == 1) {
            System.out.println("Adding a new media is currently not implemented.");
        } else if (choice == 2) {
            System.out.print("Enter media title to remove: ");
            String title = scanner.nextLine();
            Media media = store.findMediaByTitle(title);
            if (media != null) {
                store.removeMedia(media);
                System.out.println("Media removed from store.");
            } else {
                System.out.println("Media not found.");
            }
        }
    }
    private static void handleSeeCart(Cart cart, Scanner scanner) {
        cart.printAll();
        int choice;
        do {
            cartMenu();
            choice = scanner.nextInt();
            scanner.nextLine();
            switch (choice) {
                case 1:
                    System.out.println("Filter by ID or Title?");
                    String filterType = scanner.nextLine();
                    cart.filterBy(filterType);
                    break;
                case 2:
                    System.out.println("Sort by title or cost?");
                    String sortType = scanner.nextLine();
                    if (sortType.equalsIgnoreCase("title")) {
                        cart.sortBy(Media.COMPARE_BY_TITLE_COST);
                    } else if (sortType.equalsIgnoreCase("cost")) {
                        cart.sortBy(Media.COMPARE_BY_COST_TITLE);
                    }
                    break;
                case 3:
                    System.out.print("Enter media title to remove: ");
                    String title = scanner.nextLine();
                    Media media = cart.findMediaByTitle(title);
                    if (media != null) {
                        cart.removeMedia(media);
                    } else {
                        System.out.println("Media not found in cart.");
                    }
                    break;
                case 4:
                    System.out.print("Enter media title to play: ");
                    title = scanner.nextLine();
                    Media media1 = cart.findMediaByTitle(title);
                    if (media1 instanceof Playable) {
                        ((Playable) media1).play();
                    } else {
                        System.out.println("Media cannot be played.");
                    }
                    break;
                case 5:
                    cart.placeOrder();
                    break;
                case 0:
                    break;
                default:
                    System.out.println("Invalid choice.");
            }
        } while (choice != 0);
    }
}
```