Lê Minh Nghĩa - 20225894

Lab 04: Inheritance and Polymorphism

3. Creating the Book class

New Code:

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

import java.util.ArrayList;

public class Book extends Media {
    private ArrayList<String> authors;

    // Constructor
    public Book(int id, String title, String category, double cost) {
        super(id, title, category, cost); // Call the Media constructor
        this.authors = new ArrayList<>();
    }

    public ArrayList<String> getAuthors() {
        return authors;
    }

    public void addAuthor(String authorName) {
        if (!authors.contains(authorName)) {
            authors.add(authorName);
        } else {
            System.out.println("Author already exists: " + authorName);
        }
    }

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

```java
    }
    @Override
    public void displayInfo() {
        System.out.println("Book - Title: " + getTitle());
        System.out.println("Category: " + getCategory());
        System.out.println("Cost: $" + getCost());
        System.out.println("Authors: " + getAuthors());
    }

    @Override
    public String toString() {
        return "Book - Title: " + getTitle() +
                ", Category: " + getCategory() +
                ", Cost: $" + getCost() +
                ", Authors: " + authors;
    }
}
```

4. Creating the abstract Media class

Media.java

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

public abstract class Media implements Comparable<Media> {
    private int id;
    private String title;
    private String category;
    private double cost;

    // Constructor
    public Media(int id, String title, String category, double cost) {
        this.id = id;
        this.title = title;
        this.category = category;
        this.cost = cost;
    }

    // Getters
    public int getId() {
        return id;
    }

    public String getTitle() {
        return title;
    }

    public String getCategory() {
        return category;
    }

    public double getCost() {
        return cost;
    }

    // Abstract methods
    public abstract void displayInfo();
```

```java
    @Override
    public abstract String toString();

    // Implement Comparable
    @Override
    public int compareTo(Media other) {
        return this.title.compareToIgnoreCase(other.title); // Sort by title alphabetically
    }
}
```

Book.java

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

import java.util.ArrayList;

public class Book extends Media {
    private ArrayList<String> authors;

    // Constructor
    public Book(int id, String title, String category, double cost) {
        super(id, title, category, cost); // Call the Media constructor
        this.authors = new ArrayList<>();
    }

    public ArrayList<String> getAuthors() {
        return authors;
    }
```

DigitalVideoDisc.java

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

public class DigitalVideoDisc extends Disc implements Playable {
    private static int nbDigitalVideoDiscs = 0;

    // Constructors
    public DigitalVideoDisc(String title) {
        super(++nbDigitalVideoDiscs, title, category:null, cost:0.0f);
    }

    public DigitalVideoDisc(String title, String category, double cost) {
        super(++nbDigitalVideoDiscs, title, category, cost);
    }

    public DigitalVideoDisc(String title, String category, String director, double cost) {
        super(++nbDigitalVideoDiscs, title, category, cost, director, length:0);
    }

    public DigitalVideoDisc(String title, String category, String director, int length, double cost) {
        super(++nbDigitalVideoDiscs, title, category, cost, director, length);
    }
```

5. Creating the CompactDisc class

```java
public class Disc extends Media {
    private int length;
    private String director;

    // Constructors
    public Disc(int id, String title, String category, double cost) {
        super(id, title, category, cost);
    }
```

```java
public Disc(int id, String title, String category, double cost, String director, int length) {
    super(id, title, category, cost);
    this.director = director;
    this.length = length;
}

// Getters
public int getLength() {
    return length;
}

public String getDirector() {
    return director;
}

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

public void setDirector(String director) {
    this.director = director;
}

// Display Information (default implementation)
public void displayInfo() {
    System.out.println("ID: " + getId());
    System.out.println("Title: " + getTitle());
    System.out.println("Category: " + getCategory());
    System.out.println("Director: " + (director != null ? director : "N/A"));
    System.out.println("Length: " + (length > 0 ? length + " minutes" : "N/A"));
    System.out.println("Cost: $" + getCost());
}
```

```java
    @Override
    public String toString() {
        return "DVD - Title: " + getTitle() +
                ", Category: " + getCategory() +
                ", Director: " + getDirector() +
                ", Length: " + getLength() + " minutes" +
                ", Cost: $" + getCost();
    }

}
```

Track.java

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

public class Track {
    private String title;
    private int length;

    // Constructor
    public Track(String title, int length) {
        this.title = title;
        this.length = length;
    }

    // Getters
    public String getTitle() {
        return title;
    }

    public int getLength() {
        return length;
    }

    public void play() {
        System.out.println("Playing track: " + title);
        System.out.println("Track length: " + length + " minutes");
    }

    @Override
    public boolean equals(Object obj) {
        if (obj instanceof Track) {
            Track other = (Track) obj;
            return this.title.equals(other.title) && this.length == other.length;
        }
        return false;
    }
}
```

CompactDisc.java

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

import java.util.ArrayList;

public class CompactDisc extends Disc implements Playable {
    private String artist;
    private ArrayList<Track> tracks;

    // Constructor
    public CompactDisc(String title, String category, String artist, String director, double cost) {
        super(id:0, title, category, cost, director, length:0);
        this.artist = artist;
        this.tracks = new ArrayList<>();
    }

    // Getters
    public String getArtist() {
        return artist;
    }
```

```java
// Methods to manage tracks
public void addTrack(Track track) {
    if (!tracks.contains(track)) {
        tracks.add(track);
    } else {
        System.out.println("Track already exists: " + track.getTitle());
    }
}

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

public int getLength() {
    return tracks.stream().mapToInt(Track::getLength).sum();
}

@Override
public void displayInfo() {
    System.out.println("CD - Title: " + getTitle());
    System.out.println("Category: " + getCategory());
    System.out.println("Artist: " + getArtist());
    System.out.println("Director: " + getDirector());
    System.out.println("Cost: $" + getCost());
    System.out.println("Tracks:");
    for (Track track : tracks) {
        track.play();
    }
}
```

```java
@Override
public void play() {
    if (tracks.isEmpty()) {
        System.out.println("No tracks to play on CD: " + getTitle());
    } else {
        System.out.println("Playing CD: " + getTitle());
        System.out.println("Artist: " + artist);
        for (Track track : tracks) {
            track.play(); // Call the play() method of each Track
        }
    }
}

@Override
public String toString() {
    return "CD - Title: " + getTitle() +
           ", Category: " + getCategory() +
           ", Artist: " + artist +
           ", Director: " + getDirector() +
           ", Total Length: " + getLength() + " minutes" +
           ", Cost: $" + getCost();
}
```

6. Create the Playable interface

Playable.java

```
sProject > src > hust > soict > dsai > aims > media > J Playable.java > ...
1   package hust.soict.dsai.aims.media;
2
3   public interface Playable {
4       void play(); // Method to be implemented by playable media classes
5   }
6
```

DigitalVideoDisc.java

```
    @Override
    public void play() {
        if (getLength() > 0) {
            System.out.println("Playing DVD: " + getTitle());
            System.out.println("DVD length: " + getLength() + " minutes");
        } else {
            System.out.println("Cannot play DVD: " + getTitle() + " (invalid length)");
        }
    }
}
```

Track.java

```
    }

    public void play() {
        System.out.println("Playing track: " + title);
        System.out.println("Track length: " + length + " minutes");
    }
```

CompactDisc.java

```
@Override
public void play() {
    if (tracks.isEmpty()) {
        System.out.println("No tracks to play on CD: " + getTitle());
    } else {
        System.out.println("Playing CD: " + getTitle());
        System.out.println("Artist: " + artist);
        for (Track track : tracks) {
            track.play(); // Call the play() method of each Track
        }
    }
}
```

7. Update the Cart class to work with Media (Cart.java)

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

import hust.soict.dsai.aims.media.Media;
import java.util.Arrays;
import java.util.List;

public class Cart {
    public static final int MAX_NUMBERS_ORDERED = 20;
    private final Media[] itemsOrdered = new Media[MAX_NUMBERS_ORDERED];
    private int qtyOrdered = 0;

    // Add a single media item to the cart
    public void addMedia(Media media) {
        if (qtyOrdered < MAX_NUMBERS_ORDERED) {
            itemsOrdered[qtyOrdered++] = media;
            System.out.println("The media item has been added: " + media.getTitle());
        } else {
            System.out.println("The cart is full. Cannot add: " + media.getTitle());
        }
    }
    // Sort items by title (default order)
    public void sortByTitle() {
        Arrays.sort(itemsOrdered, 0, qtyOrdered); // Uses Media's compareTo
        System.out.println("Cart items sorted by title.");
    }

    // Sort items by cost
    public void sortByCost() {
        Arrays.sort(itemsOrdered, 0, qtyOrdered, (media1, media2) -> Double.compare(media1.getCost(), media2.getCost()));
        System.out.println("Cart items sorted by cost.");
    }
    // Add multiple media items using varargs
    public void addMedia(Media... mediaItems) {
        for (Media media : mediaItems) {
            addMedia(media);
        }
    }
```

```java
// Search by title
public Media searchByTitle(String title) {
    for (int i = 0; i < qtyOrdered; i++) {
        if (itemsOrdered[i] != null && itemsOrdered[i].getTitle().equalsIgnoreCase(title)) {
            return itemsOrdered[i];
        }
    }
    System.out.println("No media found with title: " + title);
    return null;
}

// Search by ID
public Media searchById(int id) {
    for (int i = 0; i < qtyOrdered; i++) {
        if (itemsOrdered[i] != null && itemsOrdered[i].getId() == id) {
            return itemsOrdered[i];
        }
    }
    System.out.println("No media found with ID: " + id);
    return null;
}
```

```java
// Display detailed cart information
public void displayCartDetailed() {
    System.out.println("**********************CART DETAILED**********************");
    System.out.println("Ordered Items:");
    for (int i = 0; i < qtyOrdered; i++) {
        Media media = itemsOrdered[i];
        media.displayInfo();
        System.out.println();
    }
    System.out.println("Total cost: $" + totalCost());
    System.out.println("*********************************************************");
}

// Display cart summary
public void displayCart() {
    System.out.println("**********************CART**********************");
    System.out.println("Ordered Items:");
    for (int i = 0; i < qtyOrdered; i++) {
        Media media = itemsOrdered[i];
        System.out.println((i + 1) + ". " + media.toString());
    }
    System.out.println("Total cost: $" + totalCost());
    System.out.println("************************************************");
}
```

```java
// Remove a media item from the cart
public void removeMedia(Media media) {
    boolean removed = false;
    for (int i = 0; i < qtyOrdered; i++) {
        if (itemsOrdered[i] == media) {
            for (int j = i; j < qtyOrdered - 1; j++) {
                itemsOrdered[j] = itemsOrdered[j + 1];
            }
            itemsOrdered[--qtyOrdered] = null;
            System.out.println("The media item has been removed: " + media.getTitle());
            removed = true;
            break;
        }
    }
    if (!removed) {
        System.out.println("The media item was not found in the cart.");
    }
}

// Filter cart by title
public void filterByTitle(String title) {
    System.out.println("Filtered items by title:");
    for (Media media : itemsOrdered) {
        if (media != null && media.getTitle().toLowerCase().contains(title.toLowerCase())) {
            System.out.println(media.toString());
        }
    }
}
```

```java
    public List<Media> getItems() {
        return Arrays.asList(itemsOrdered).subList(0, qtyOrdered);
    }

    // Filter cart by cost range
    public void filterByCost(double minCost, double maxCost) {
        System.out.println("Filtered items by cost range:");
        for (Media media : itemsOrdered) {
            if (media != null && media.getCost() >= minCost && media.getCost() <= maxCost) {
                System.out.println(media.toString());
            }
        }
    }

    // Clear the cart
    public void clearCart() {
        for (int i = 0; i < qtyOrdered; i++) {
            itemsOrdered[i] = null; // Clear each element
        }
        qtyOrdered = 0; // Reset quantity
        System.out.println("Cart cleared.");
    }


    // Calculate the total cost of all media items
    public double totalCost() {
        double total = 0;
        for (int i = 0; i < qtyOrdered; i++) {
            total += itemsOrdered[i].getCost();
        }
        return total;
    }
}
```

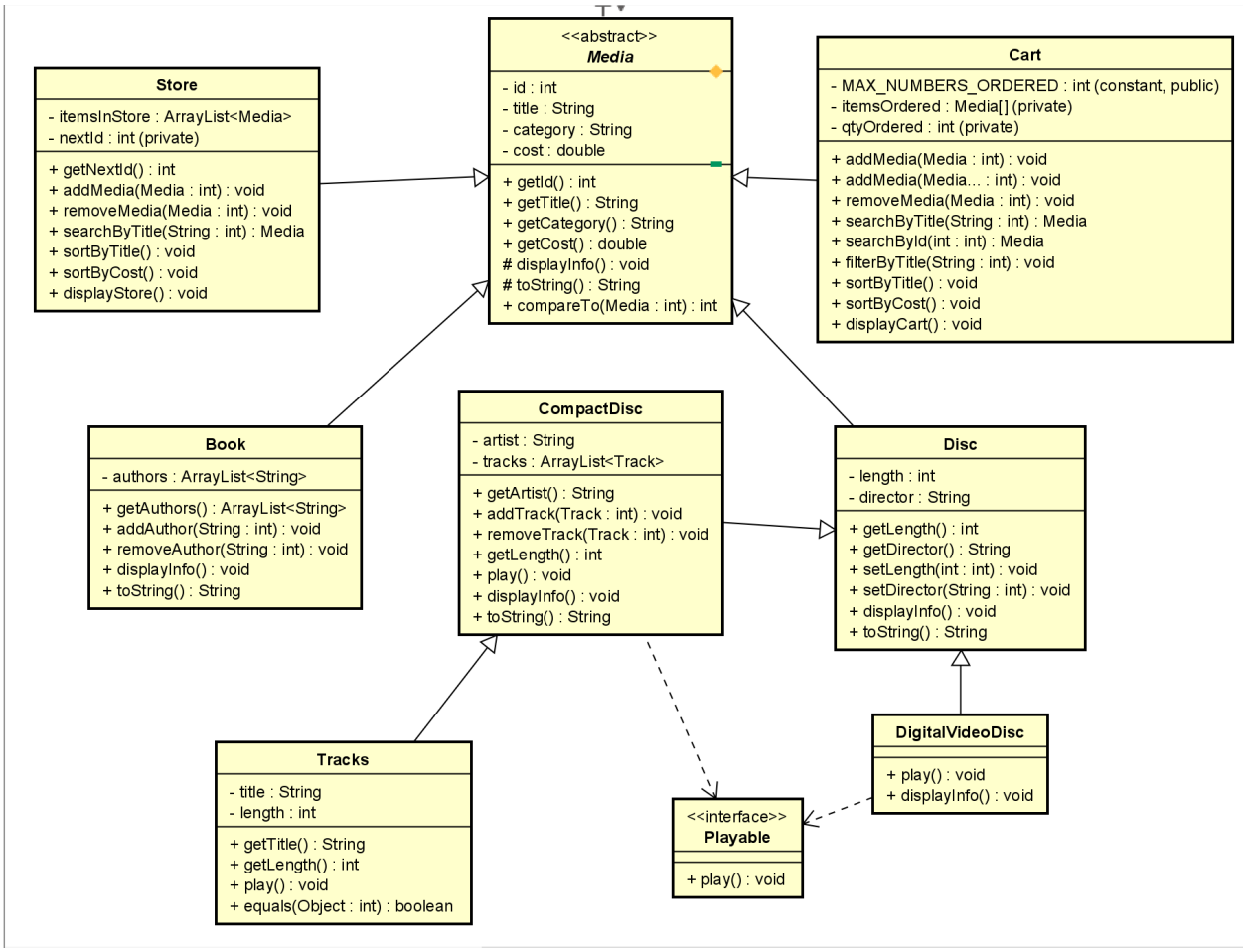
8. Update the Store class to work with Media

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

import hust.soict.dsai.aims.media.Media;
import java.util.ArrayList;
import java.util.Collections;

public class Store {
    private final ArrayList<Media> itemsInStore = new ArrayList<>();
    private int nextId = 1;

    public int getNextId() {
        return nextId++;
    }

    // Add a media item to the store
    public void addMedia(Media media) {
        if (!itemsInStore.contains(media)) {
            itemsInStore.add(media);
            System.out.println("Added to store: " + media.getTitle());
        } else {
            System.out.println("Media item already exists: " + media.getTitle());
        }
    }
```

```java
// Sort items by title (default order)
public void sortByTitle() {
    Collections.sort(itemsInStore); // Uses Media's compareTo()
    System.out.println("Store items sorted by title.");
}

public Media searchByTitle(String title) {
    for (Media media : itemsInStore) {
        if (media.getTitle().equalsIgnoreCase(title)) {
            return media;
        }
    }
    return null;
}

public void removeMedia(Media media) {
    if (itemsInStore.remove(media)) {
        System.out.println("Media removed from store: " + media.getTitle());
    } else {
        System.out.println("Media not found in store.");
    }
}

// Sort items by cost
public void sortByCost() {
    itemsInStore.sort((media1, media2) -> Double.compare(media1.getCost(), media2.getCost()));
    System.out.println("Store items sorted by cost.");
}
```

```java
// Display all media items in the store
public void displayStore() {
    System.out.println("**********STORE**********");
    for (Media media : itemsInStore) {
        System.out.println(media.toString());
    }
    System.out.println("**************************************************");
}
```

9. Constructors of whole classes and parent classes



10. Unique item in a list

Book.java

```java
public void addAuthor(String authorName) {
    if (!authors.contains(authorName)) {
        authors.add(authorName);
    } else {
        System.out.println("Author already exists: " + authorName);
    }
}

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

CompactDisc.java

```java
public void addTrack(Track track) {
    if (!tracks.contains(track)) {
        tracks.add(track);
    } else {
        System.out.println("Track already exists: " + track.getTitle());
    }
}

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

Store.java

```java
// Add a media item to the store
public void addMedia(Media media) {
    if (!itemsInStore.contains(media)) {
        itemsInStore.add(media);
        System.out.println("Added to store: " + media.getTitle());
    } else {
        System.out.println("Media item already exists: " + media.getTitle());
    }
}
```

11. Polymorphism with toString() method

Media.java

```
    // Abstract methods
    public abstract void displayInfo();

    @Override
    public abstract String toString();

    // Implement Comparable
```

Disc.java, DigitalVideoDisc.java, CompactDisc.java, Book.java

```
    @Override
    public String toString() {
        return "Book - Title: " + getTitle() +
                ", Category: " + getCategory() +
                ", Cost: $" + getCost() +
                ", Authors: " + authors;
    }
}
```

```
    @Override
    public String toString() {
        return "CD - Title: " + getTitle() +
                ", Category: " + getCategory() +
                ", Artist: " + artist +
                ", Director: " + getDirector() +
                ", Total Length: " + getLength() + " minutes" +
                ", Cost: $" + getCost();
    }
}
```

```
    }

    @Override
    public String toString() {
        return "DVD - Title: " + getTitle() +
                ", Category: " + getCategory() +
                ", Director: " + getDirector() +
                ", Length: " + getLength() + " minutes" +
                ", Cost: $" + getCost();
    }
}
```

12. Sort media in the cart (I am using comparable)

```
    public abstract String toString();

    // Implement Comparable
    @Override
    public int compareTo(Media other) {
        return this.title.compareToIgnoreCase(other.title); // Sort by title alphabetically
    }
}
```

```
// Sort items by title (default order)
public void sortByTitle() {
    Collections.sort(itemsInStore); // Uses Media's compareTo()
    System.out.println("Store items sorted by title.");
}
```

```
// Sort items by cost
public void sortByCost() {
    itemsInStore.sort((media1, media2) -> Double.compare(media1.getCost(), media2.getCost()));
    System.out.println("Store items sorted by cost.");
}
```

13. Create a complete console application in the Aims class

```
    private 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");
    }

    private static void storeMenu(Scanner scanner) {
        while (true) {
            store.displayStore();
            System.out.println("Options: ");
            System.out.println("--------------------------------");
            System.out.println("1. See media details");
            System.out.println("2. Add media to cart");
            System.out.println("3. Play media");
            System.out.println("4. See current cart");
            System.out.println("0. Back to main menu");
            System.out.println("--------------------------------");
            System.out.println("Please choose a number: 0-1-2-3-4");
            int choice = getChoice(scanner);

            switch (choice) {
```

```
    private static void updateStore(Scanner scanner) {
        while (true) {
            System.out.println("Update Store: ");
            System.out.println("--------------------------------");
            System.out.println("1. Add media to store");
            System.out.println("2. Remove media from store");
            System.out.println("0. Back to main menu");
            System.out.println("--------------------------------");
            System.out.println("Please choose a number: 0-1-2");
            int choice = getChoice(scanner);

            switch (choice) {
                case 1:
                    addMediaToStore(scanner);
                    break;
                case 2:
                    removeMediaFromStore(scanner);
                    break;
                case 0:
                    return;
                default:
                    System.out.println("Invalid choice. Please select a valid option.");
            }
        }
```

```
    private static void mediaDetailsMenu(Scanner scanner) {
        System.out.print("Enter the title of the media: ");
        String title = scanner.nextLine();
        Media media = store.searchByTitle(title);

        if (media != null) {
            System.out.println(media.toString());
            System.out.println("Options: ");
            System.out.println("--------------------------------");
            System.out.println("1. Add to cart");
            System.out.println("2. Play");
            System.out.println("0. Back to store menu");
            System.out.println("--------------------------------");
            System.out.println("Please choose a number: 0-1-2");
            int choice = getChoice(scanner);

            switch (choice) {
                case 1:
                    cart.addMedia(media);
                    System.out.println("Media added to cart.");
                    break;
                case 2:
                    if (media instanceof Playable) {
                        ((Playable) media).play();
                    } else {
                        System.out.println("This media cannot be played.");
                    }
                    break;
                case 0:
                    return;
                default:
                    System.out.println("Invalid choice. Returning to store menu.");
```

```
    }

    private static void filterCartMenu(Scanner scanner) {
        System.out.println("Filter by:");
        System.out.println("1. ID");
        System.out.println("2. Title");
        System.out.println("0. Cancel");
        System.out.println("--------------------------------");
        System.out.println("Please choose a number: 0-1-2");

        int filterChoice = getChoice(scanner);

        switch (filterChoice) {
            case 1:
```

```
}

private static void cartMenu(Scanner scanner) {
    while (true) {
        cart.displayCartDetailed();
        System.out.println("Options: ");
        System.out.println("--------------------------------");
        System.out.println("1. Filter cart");
        System.out.println("2. Sort cart by title");
        System.out.println("3. Sort cart by cost");
        System.out.println("4. Remove media from cart");
        System.out.println("5. Play a media");
        System.out.println("6. Checkout");
        System.out.println("0. Back to main menu");
        System.out.println("--------------------------------");
        System.out.println("Please choose a number: 0-1-2-3-4-5-6");
        int choice = getChoice(scanner);
```

```
}

private static void addMediaToStore(Scanner scanner) {
    System.out.println("Choose media type to add: ");
    System.out.println("1. Digital Video Disc (DVD)");
    System.out.println("2. Book");
    System.out.println("3. Compact Disc (CD)");
    System.out.println("0. Cancel");
    System.out.print("Your choice: ");
    int type = getChoice(scanner);

    switch (type) {
        case 1: // Add DVD
            System.out.print("Enter title: ");
            String dvdTitle = scanner.nextLine();
```

Questions:

1. Lớp nào nên triển khai interface Comparable?

Bất kỳ lớp nào cần được sắp xếp dựa trên các thuộc tính của nó nên triển khai interface Comparable. Ví dụ, nếu bạn có một lớp gọi là Item đại diện cho các mục trong giỏ hàng, nó nên triển khai Comparable<Item>.

```
public class Item implements Comparable<Item> {
  private String title;
  private double cost;

  // Constructor, getters, và setters bị bỏ qua để tiết kiệm không gian

  @Override
  public int compareTo(Item other) {
    // Logic so sánh sẽ được thực hiện ở đây
  }
}
```

2. Trong những lớp đó, bạn nên triển khai phương thức compareTo() như thế nào để phản ánh thứ tự mà chúng ta muốn?

Phương thức compareTo() phải trả về một số nguyên cho biết thứ tự của đối tượng hiện tại so với một đối tượng khác:

- Một số nguyên âm nếu đối tượng này nhỏ hơn đối tượng đã chỉ định.

- Zero nếu đối tượng này bằng với đối tượng đã chỉ định.

- Một số nguyên dương nếu đối tượng này lớn hơn đối tượng đã chỉ định.

Nếu bạn muốn sắp xếp các mục trước tiên theo tiêu đề và sau đó theo chi phí, việc triển khai của bạn có thể như sau:

```
@Override
public int compareTo(Item other) {
  int titleComparison = this.title.compareTo(other.title);
  if (titleComparison != 0) {
    return titleComparison; // Sắp xếp theo tiêu đề
  }
  return Double.compare(this.cost, other.cost); // Sắp xếp theo chi phí
}
```

3. Chúng ta có thể có hai quy tắc sắp xếp của mục (theo tiêu đề rồi đến chi phí và theo chi phí rồi đến tiêu đề) nếu chúng ta sử dụng cách tiếp cận interface Comparable không?

Sử dụng interface Comparable, bạn chỉ có thể định nghĩa một thứ tự tự nhiên cho mỗi lớp. Tuy nhiên, nếu bạn cần nhiều quy tắc sắp xếp (như sắp xếp theo tiêu đề rồi đến chi phí và theo chi phí rồi đến tiêu đề), bạn thường sẽ sử dụng interface Comparator thay thế.

Ví dụ:

```
Comparator<Item> byCostThenTitle = Comparator.comparingDouble(Item::getCost)
                                   .thenComparing(Item::getTitle);
```

4. Giả sử các DVD có quy tắc sắp xếp khác với các loại phương tiện khác, đó là theo tiêu đề, sau đó là độ dài giảm dần, rồi đến chi phí. Bạn sẽ sửa đổi mã của mình như thế nào để cho phép điều này?

Nếu các DVD có quy tắc sắp xếp cụ thể (theo tiêu đề, sau đó là độ dài giảm dần, rồi đến chi phí), bạn có thể đạt được điều này bằng cách tạo một lớp riêng cho DVD mà triển khai Comparable:

```
public class DVD implements Comparable<DVD> {
  private String title;
  private double length; // Giả sử độ dài tính bằng phút
  private double cost;

  @Override
  public int compareTo(DVD other) {
    int titleComparison = this.title.compareTo(other.title);
    if (titleComparison != 0) {
      return titleComparison; // Sắp xếp theo tiêu đề
    }
    int lengthComparison = Double.compare(other.length, this.length); // Độ dài giảm dần
    if (lengthComparison != 0) {
      return lengthComparison; // Sắp xếp theo độ dài theo thứ tự giảm dần
    }
    return Double.compare(this.cost, other.cost); // Sắp xếp theo chi phí
  }
}
```