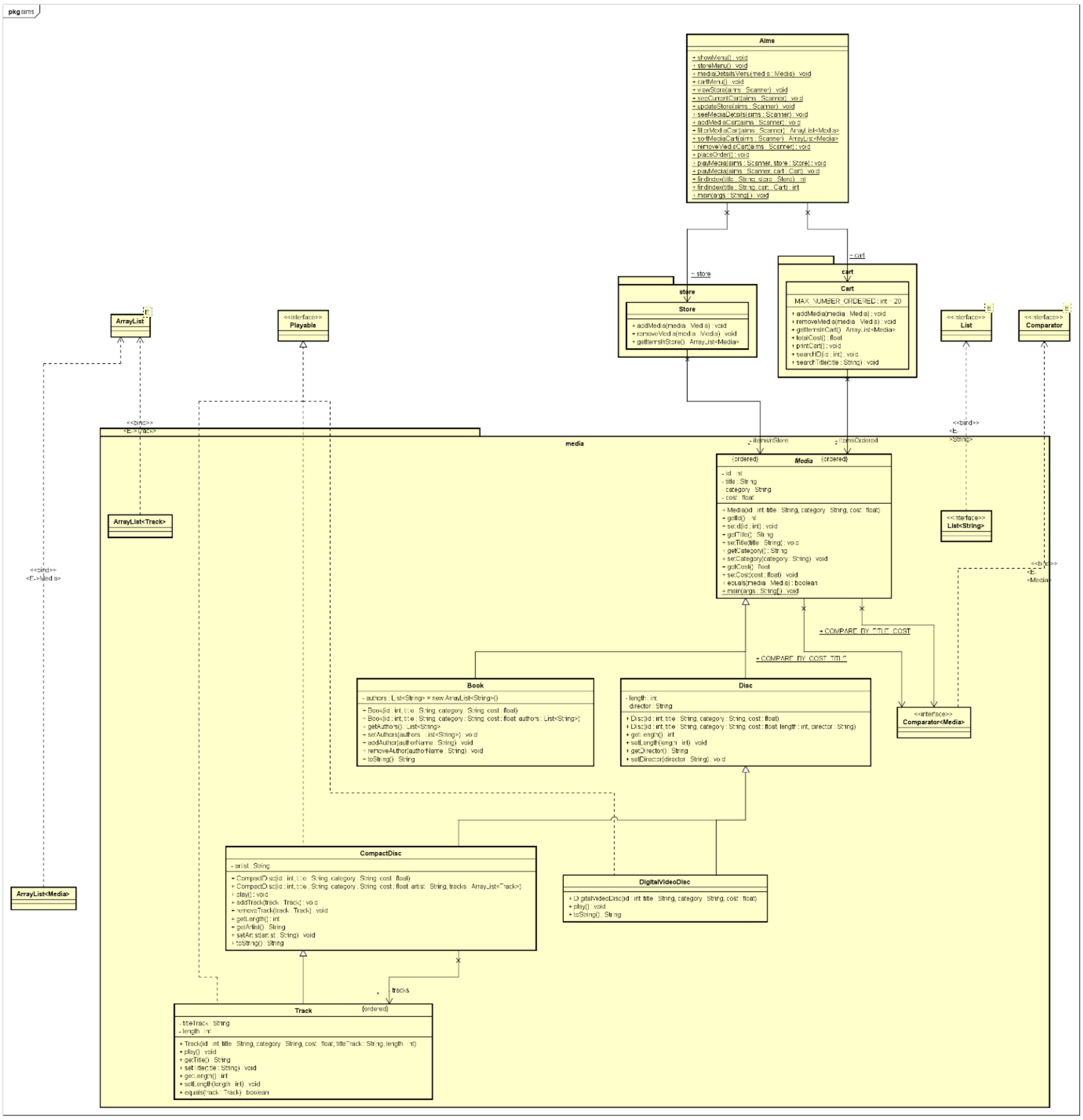
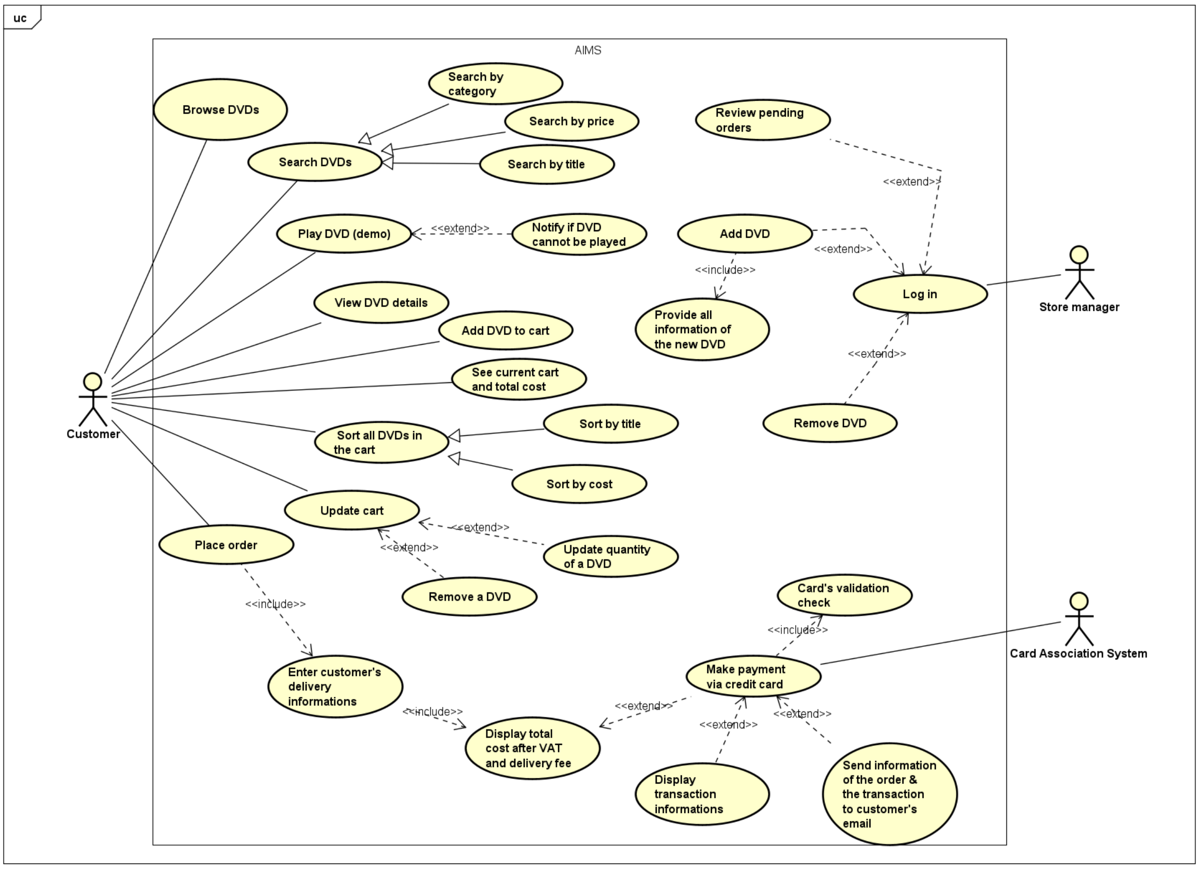
1. Updated diagram


pkg aims
Aims
+ showMenu() : void
+ storeMenu() : void
+ mediaDetailsMenu(media : Media) : void
+ cartMenu() : void
+ viewStore(aims : Scanner) : void
+ seeCurrentCart(aims : Scanner) : void
+ updateStore(aims : Scanner) : void
+ seeMediaDetails(aims : Scanner) : void
+ addMediaCart(aims : Scanner) : void
+ filterMediaCart(aims : Scanner) : ArrayList<Media>
+ sortMediaCart(aims : Scanner) : ArrayList<Media>
+ removeMediaCart(aims : Scanner) : void
+ placeOrder() : void
+ playMedia(aims : Scanner, store : Store) : void
+ playMedia(aims : Scanner, cart : Cart) : void
+ findIndex(title : String, store : Store) : int
+ findIndex(title : String, cart : Cart) : int
+ main(args : String[]) : void
~ store
~ cart
store
Store
+ addMedia(media : Media) : void
+ removeMedia(media : Media) : void
+ getItemsInStore() : ArrayList<Media>
cart
Cart
- MAX_NUMBER_ORDERED : int = 20
+ addMedia(media : Media) : void
+ removeMedia(media : Media) : void
+ getItemsInCart() : ArrayList<Media>
+ totalCost() : float
+ printCart() : void
+ searchID(id : int) : void
+ searchTitle(title : String) : void
ArrayList
E
<<interface>>
Playable
<<interface>>
List
E
<<interface>>
Comparator
E
<<bind>>
<E->Track>
<<bind>>
<E->Media>
<<bind>>
<E->String>
<<bind>>
<E->Media>
media
* - itemsInStore
* - itemsOrdered
{ordered}
Media
{ordered}
- id : int
- title : String
- category : String
- cost : float
+ Media(id : int, title : String, category : String, cost : float)
+ getId() : int
+ setId(id : int) : void
+ getTitle() : String
+ setTitle(title : String) : void
+ getCategory() : String
+ setCategory(category : String) : void
+ getCost() : float
+ setCost(cost : float) : void
+ equals(media : Media) : boolean
+ main(args : String[]) : void
+ COMPARE_BY_TITLE_COST
+ COMPARE_BY_COST_TITLE
ArrayList<Track>
<<interface>>
List<String>
<<interface>>
Comparator<Media>
Book
- authors : List<String> = new ArrayList<String>()
+ Book(id : int, title : String, category : String, cost : float)
+ Book(id : int, title : String, category : String, cost : float, authors : List<String>)
+ getAuthors() : List<String>
+ setAuthors(authors : List<String>) : void
+ addAuthor(authorName : String) : void
+ removeAuthor(authorName : String) : void
+ toString() : String
Disc
- length : int
- director : String
+ Disc(id : int, title : String, category : String, cost : float)
+ Disc(id : int, title : String, category : String, cost : float, length : int, director : String)
+ getLength() : int
+ setLength(length : int) : void
+ getDirector() : String
+ setDirector(director : String) : void
ArrayList<Media>
CompactDisc
- artist : String
+ CompactDisc(id : int, title : String, category : String, cost : float)
+ CompactDisc(id : int, title : String, category : String, cost : float, artist : String, tracks : ArrayList<Track>)
+ play() : void
+ addTrack(track : Track) : void
+ removeTrack(track : Track) : void
+ getLength() : int
+ getArtist() : String
+ setArtist(artist : String) : void
+ toString() : String
DigitalVideoDisc
+ DigitalVideoDisc(id : int, title : String, category : String, cost : float)
+ play() : void
+ toString() : String
* - tracks
Track
{ordered}
- titleTrack : String
- length : int
+ Track(id : int, title : String, category : String, cost : float, titleTrack : String, length : int)
+ play() : void
+ getTitle() : String
+ setTitle(title : String) : void
+ getLength() : int
+ setLength(length : int) : void
+ equals(track : Track) : boolean



2. Answer for questions

9. Which classes are aggregates of other classes?

Class CompactDisc is the aggregate of class Track.

Class Cart and Store are the aggregates of class Media.

10. If the passing object is not an instance of Media, what happens?

If the passing object is not an instance of Media, it returns false.

11. The output is:

ID - 5. Book - khonggiadinh - slice of life - [] - $55.6

ID - 1. DVD - starWar - action - null - 0s - $1563.6

ID - 3. CD - superman - action - null - null - 0s - $305.5

The toString() method is overridden in a subclass to provide a more meaningful or customized string representation of the object.

12.

-What class should implement the Comparable interface?

The class that have the item to compare should implement the Comparable interface. In this case is class Media.

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

If we want to sort the items by title, then the compareTo() method would compare the titles of the items. If we want to sort the items by cost, then the compareTo() method would compare the costs of the items.

-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?

Yes, we can have two ordering rules of the item (by title then cost and by cost then title) if we use Comparable interface approach. We can do this by creating two different implementations of the Comparable interface. One implementation would sort the items by title, then cost. The other implementation would sort the items by cost, then title.

-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?

To allow DVDs to have a different ordering rule from the other media types. The DVD class would override the compareTo() method to sort the DVDs by title, then decreasing length, then cost.