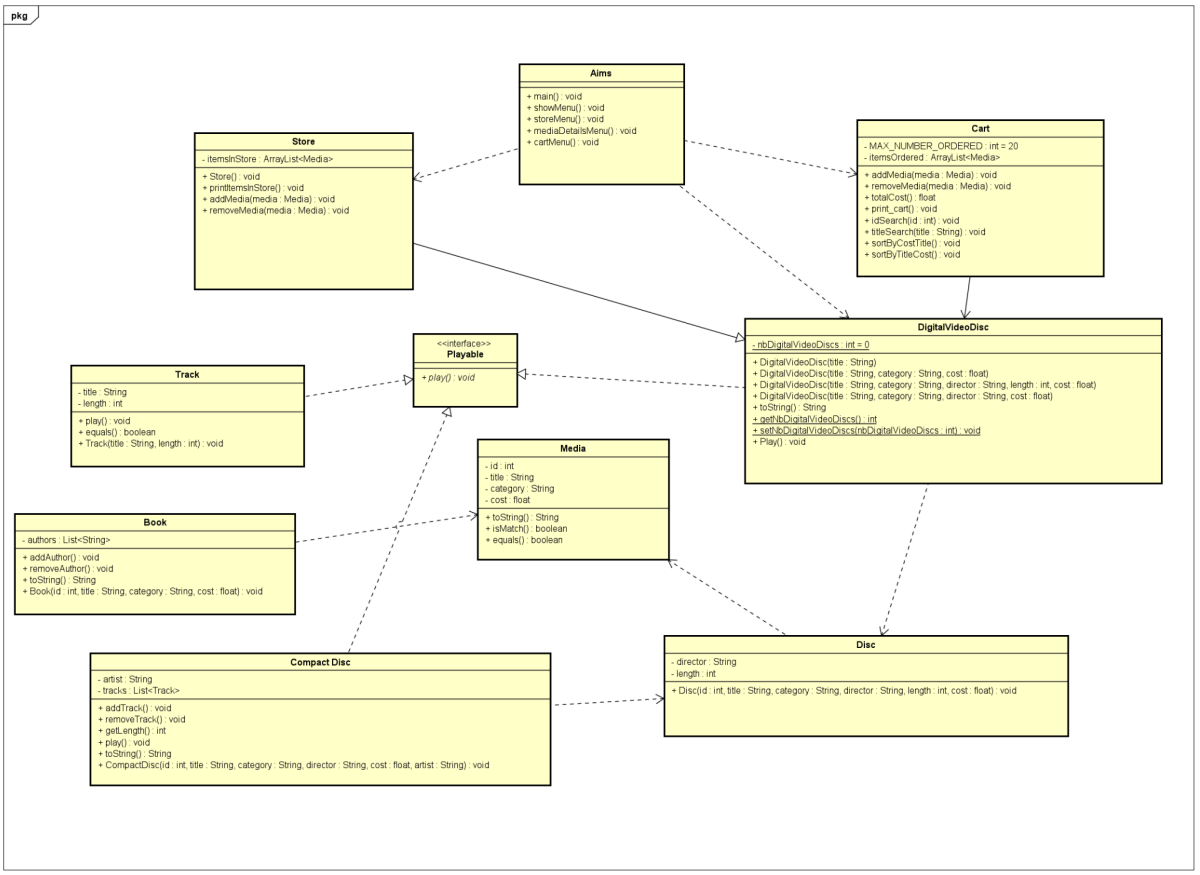
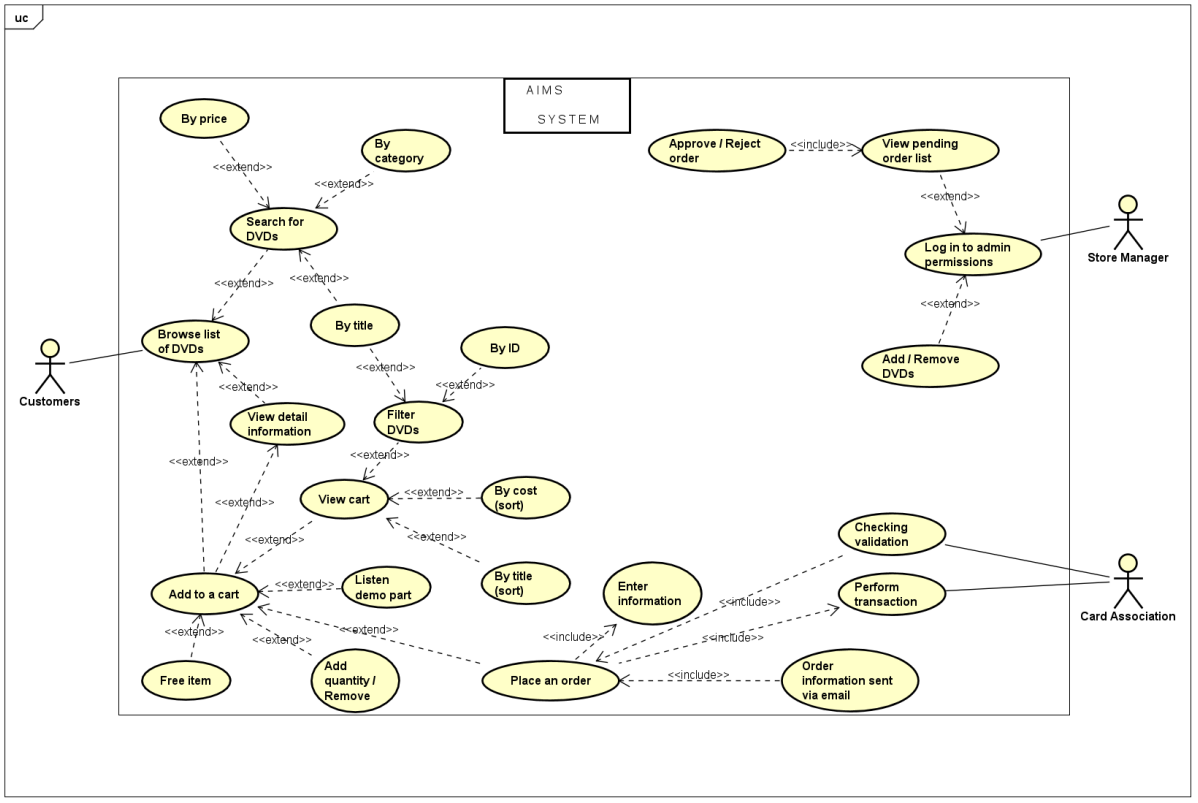
REPORT LAB 04:

1. **Class diagram && Use case diagram (updated)**
2. **Answers**

1. Class diagram && Use case diagram





2. Answers

9. Constructors of whole classes and parent classes

Q: Which classes are aggregates of other classes?

A: Aggregates:

+ Store && Cart aggregates Media.
+ CompactDisc aggregates Track.

10. Unique item in a list

Q: If the passing object is not an instance of Media, what happens?

A: If the passing object (obj) is not an instance of Media, the equals() method will return false because the comparison cannot be performed. The equals() method in the Media class includes a check to ensure that obj belongs to the same class as Media using the getClass() method.

11. Polymorphism with toString() method

Q: Iterate through the list and print out the information of the media by using toString() method. Observe what happens and explain in detail.

A: When we iterate through the list using a for-each loop, the toString() method is called on each Media object. The toString() method is dynamically bound, meaning it is called based on the actual type of the object at runtime.

So, for each media object in the list, the appropriate toString() method of the respective class (Book, DVD, or CompactDisc) is invoked, and the information specific to that type is printed.

This demonstrates the concept of polymorphism, where objects of different types can be treated as objects of a common superclass (Media in this case), allowing for code reuse and flexibility.

12. Sort media in the cart

Q: What class should implement the Comparable interface?

A: The Media class since we want to define a default ordering for media objects.

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

A: e.g.

```
public class Media implements Comparable<Media> {
   // Other attributes and methods
   @Override
   public int compareTo(Media other) {
      if (this.cost < other.cost) {
         return -1;
      } else if (this.cost > other.cost) {
         return 1;
      } else {
         return 0;
      }
   }
}
```

Q: 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?

A: No, we can't because the Comparable interface allows only one natural ordering for a class

Q: 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?

A: Implement another compareTo() method in the DVD class