**BÁO CÁO THỰC HÀNH LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG – LAB 02**

**Họ và tên: Trần Đại Hiệp**

**MSSV: 20226081**

# 3. Use case diagram

Based on the problem statement in section 2, please draw the use case diagram using Astah UML for the AIMS project

A diagram of a network

Description automatically generated with medium confidence

# 4. UML diagram for use cases related to cart management

A screenshot of a computer

Description automatically generated

# 5. Create Aims Project

# 6. Create DigitalVideoDisc class and its attributes

# A screen shot of a computer program Description automatically generated

# *7. Create accessors and mutators for the class DigitalVideoDisc*

A screen shot of a computer program

Description automatically generated

**Reading Assignment: When should accessor methods be used?**

**Answer:** Accessor methods, or "getters," provide controlled access to an object's attributes. They are commonly used in object-oriented programming to enforce encapsulation, data integrity, and readability. Here are the key reasons for using accessor methods:

**1. Encapsulation**

* **Prevent Direct Data Access**: Accessor methods are fundamental to encapsulation in object-oriented programming. They prevent direct access to an object's internal state, allowing data to remain private. This reduces the risk of accidental or unauthorized modifications by external classes and ensures that the object maintains control over its data.
* **Control Data Access**: By restricting access to an object’s internal fields, accessors keep the memory structure or data location hidden. This practice not only enhances security but also allows the class designer to change internal implementations without affecting code that relies on the public methods.

**2. Read-Only Access**

* **Limit Data Modification**: Accessor methods allow certain attributes to be read-only by providing only a "getter" without a corresponding "setter." This is particularly useful when a field should be immutable or its value should remain constant after being initialized (e.g., a unique identifier or creation timestamp).
* **Maintain Data Integrity**: In cases where multiple parts of a program depend on a particular piece of data remaining unchanged, accessors prevent unintended modification. For example, final configuration values or constants accessed by multiple components can be safely exposed without the risk of being altered.

**3. Enhanced Validation**

* **Control Data Access Logic**: Accessor methods provide a point to introduce additional business logic or data transformation every time the attribute is accessed. For instance, an accessor can apply formatting, return computed values, or hide certain sensitive information based on user roles.
* **Ensure Data Accuracy**: With accessors, you can include validation checks to ensure that only valid, expected data is returned. This can help enforce consistency in data usage across the system, reducing the risk of incorrect values being processed.

**4. Improved Readability**

* **Consistent Interface**: Accessor methods offer a clear, consistent interface for interacting with an object's attributes, making the code easier for other developers to understand and use. Even without knowledge of the internal data structure, developers can retrieve information using well-defined getter methods.
* **Simplified Interpretation**: By naming accessors descriptively, you clarify the intent and role of each attribute. For example, calling *getTitle()* instead of directly accessing *title* makes the code more readable and meaningful, especially in larger codebases where understanding the purpose of each field at a glance is essential.

**Mindmap:**

A diagram of a diagram

Description automatically generated with medium confidence

# 8. Create constructor methods

A screen shot of a computer program

Description automatically generated

**Question: If you create a constructor method to build a DVD by title then create a constructor method to build a DVD by category. Does JAVA allow you to do this?**

**Answer**: Java doesn’t allow us to do this. Since “title” and “category” are String, they are basically the same parameters and the same object type. So if we add the constructor DigitalVideoDisc(String category), Java will not be able to differentiate between this and the constructor DigitalVideoDisc(String title).

# 9. Create the Cart class to work with DigitalVideoDisc

A computer screen shot of a program code

Description automatically generatedA screen shot of a computer code

Description automatically generatedA computer code on a black background

Description automatically generated

# 10. Create Aims

A screen shot of a computer program

Description automatically generated

**Compile results:**

# **A screenshot of a computer error Description automatically generated**

**Explaining:**

* + After adding a disk successfully, the checkout box “The disc has been added!” appears. The first three lines mean there are already 3 DVDS in the cart.
  + The method anOrder.totalCost() (line 20) calculates the total cost of the included DVDS.
  + The method anOrder.removeDigitalVideoDisc(dvd1) (line 22) means remove the dvd1 in the cart. After successfully removing the disk, the checkout box “The disc is successfully removed!” appears.
  + Now the total cost has been changed, due to the removal of the disk dvd1.
  + When we try to remove the disk dvd1 one more time (line 26), the system could not find it in the cart and print “Removed Failed. The disk …”
  + Because there are dvd2 and dvd3 left, it would be successfully removed them without any errors.
  + The last command tries to remove the disk 1, but now there aren’t any DVDs left in the cart. As the result, the system print out “The cart is empty!”.