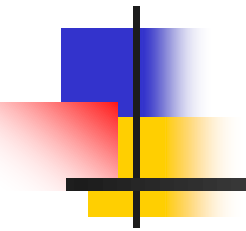
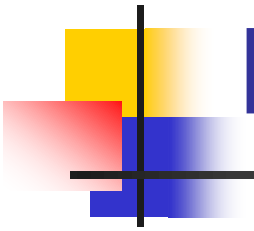


Lab 06. Aggregation and Inheritance

Lý thuyết và ngôn ngữ hướng đối tượng (bài tập)



Lab's Objectives

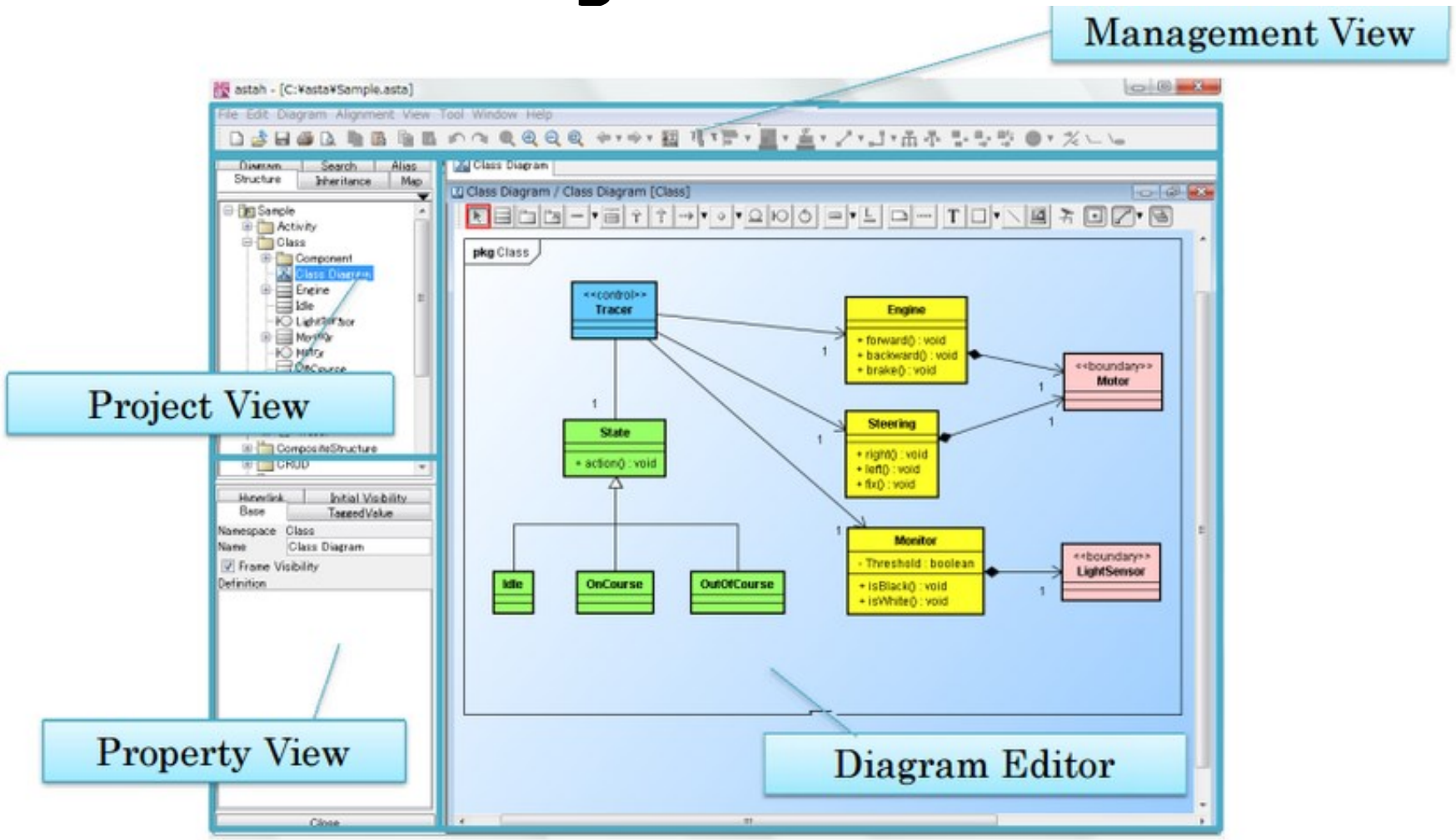
- In this lab, you will practice with:
  - JAVA Inheritance mechanism
  - Use the Collections framework (specifically, ArrayList)
  - Refactoring your JAVA code

Introduction to Astah

- **What is Astah?**
  - Astah is an editor that integrates features for software development
  - Build chart types in UML: class diagram, sequence diagram, activity chart, ...
- **Install Astah**
  - Visit http://astah.net/download and select the astah version that matches your device configuration.
  - For linux, download the .deb file to your computer. After downloading, just run the file to install astah.

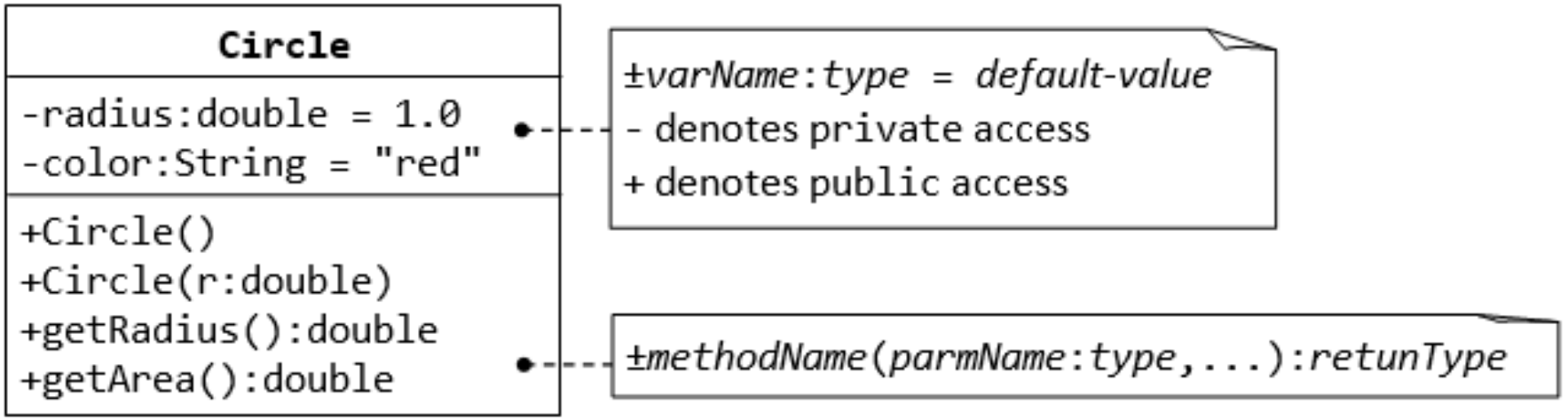
Introduction to Astah

- **Astah's working screen**



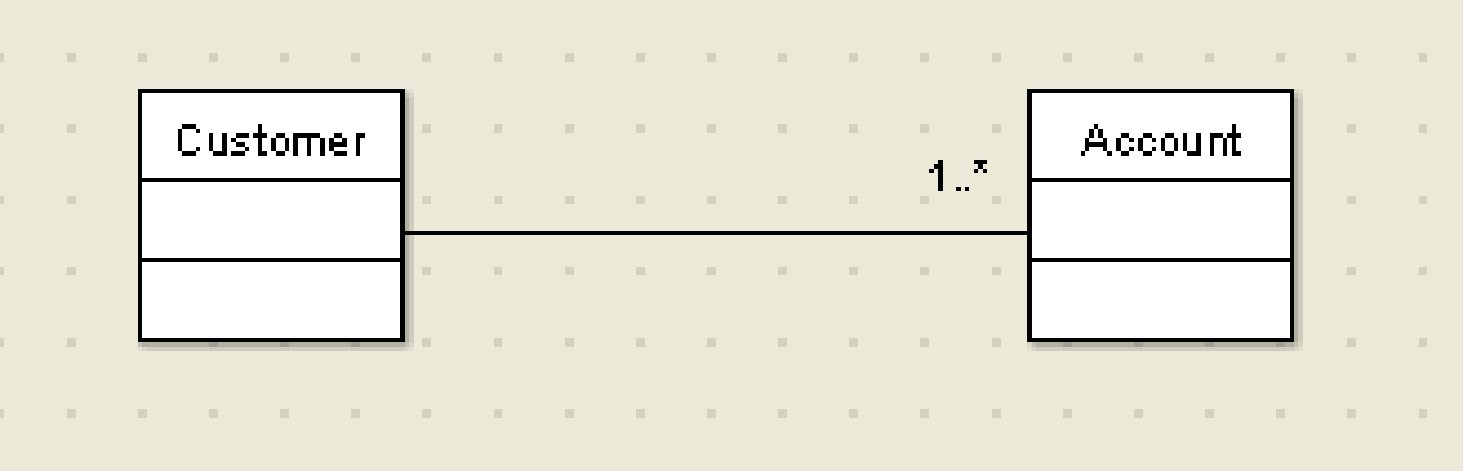
Introduction to Astah

- **Draw UML Class Diagram and export as an image**



Relationship

- **Associations**
  - Associations are **structural relationships** between classes and represent connections between objects.
  - Associations are used to indicate that an object of one class is linked to an object of the other class.



Relationship

- **Navigability**
  - Arrows can be applied to an association to indicate the direction of a relationship.
  - The direction of the arrow tells us which class has knowledge of the other.

Relationship

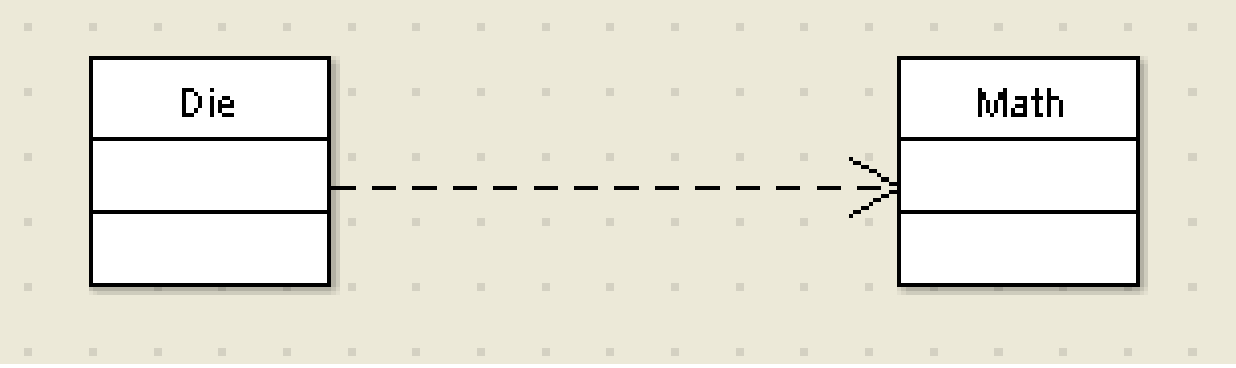
- **Aggregations**
  - An aggregation is a special kind of association that is used to denote a whole-part relationship between classes.

Relationship

- **Composition**
  - A composition is a special type of aggregation in which the lifetime of the whole and the parts is linked.
  - When the whole is destroyed, the parts are necessarily destroyed.

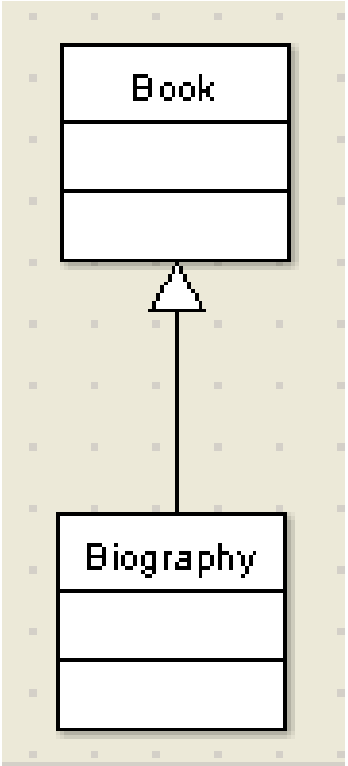
Relationship

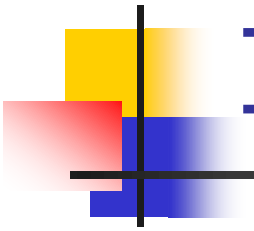
- **Dependency / Use**
  - A dependency between two classes indicates that the functionality of one class depends on that of the other.
  - This relationship is NOT structural like an association is, but it indicates that one class USES one or more instances of the other class in order to perform its tasks.



Relationship

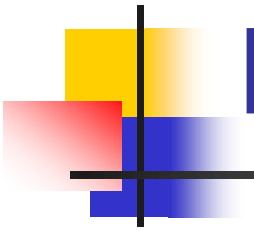
- **Generalization / Inheritance**
  - Generalizations are used to show that one class is a more specialized type of another class.
  - They are used to illustrate inheritance relationships.





Inheritance

- Mechanism to create new classes using existing classes
- The existing class is called the *parent class,* or *superclass*, or *base class*
- The derived class is called the *child class* or *subclass*
- As the name implies, the child inherits characteristics of the parent
- That is, the child class inherits the methods and data defined by the parent class



Benefit of Inheritance

- Reusability
  - Once a behavior (method) is defined in a super class, that behavior is automatically inherited by all subclasses
  - Thus, you write a method only once and it can be used by all subclasses.
  - Once a set of properties (fields) are defined in a super class, the same set of properties are inherited by all subclasses
  - A class and its children share common set of properties
  - A subclass only needs to implement the differences between itself and the parent.

Definition of sub-class

- To derive a child class, we use the extends keyword.
- Syntax:

[Modifier] class Sub-class_name ***extends*** *SuperClass_name{*

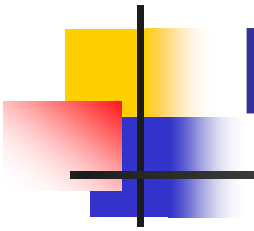
Definition of member variable to add

Method definition to add or redefine

}

```
// Account class (superclass)
class Account {
    String name; //Account name
    int balance; //Balance

    void display(){.....}
                    // Display
}
```

```
// Integrated account class (sub-class)
class IntegratedAccount extends Account {
    int limitOverdraft; //Borrowing amount limit
    int overdraft; //Borrowing amount
    void loan(int money){...} //Borrowing()
    void display(){.....}        // Display
}
IntegratedAccount obj= new IntegratedAccount();
```



Lab's content

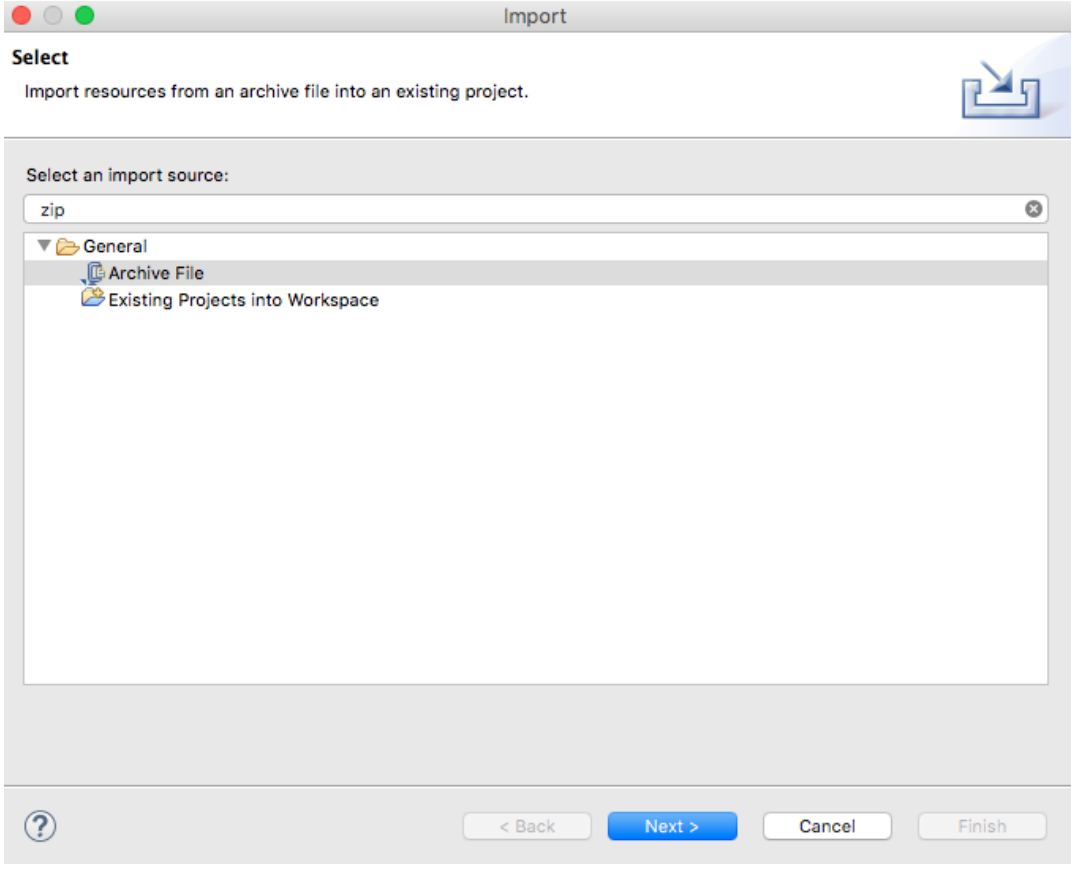
- In this exercise you extend the AIMS system that you created in the previous exercises to allow the ordering of books.
- You will create a **Book** class which stores the title, category, cost and an **ArrayList** of authors.
- Since the **Book** and **DigitalVideoDisc** classes share some common fields and methods, the classes are refactored, and a common superclass **Media** is created.
- The **Order** class is updated to accept any type of media object (either books or DVDs)

Lab's content

- **1. Import the existing project into the workspace of Eclipse**
  - **Open Eclipse**
  - Open File -> Import. Type zip to find Archive File if you have exported as a zip file before.
  - You may choose Existing Projects into Workspace if you want to open an existing project in your computer.
    - Ignore this step if the AimsProject is already opened in the workspace.

Lab's content

- Click Next and Browse to a zip file or a project to open
- Once the project is imported, you should see the classes you created in the previous lab, namely, **Aims**, **Order**, **DigitalVideoDisc**.



Lab's content

- **2. Creating the Book class**
  - In the Package Explorer view, right-click the project and select New -> Class. Adhere to the following specifications:
    - Package: **hust.soict.aims.media**
    - Name: **Book**
    - Access modifier: **public**
    - Superclass: **java.lang.Object**
    - **public static void main(String[] args)**: **do not check**
    - Constructors from Superclass: **Check**
    - All other boxes: **Do not check**

Lab's content

- **2. Creating the Book class**
  - **Add fields to the Book class**
  - To store the information about a **Book**, the class requires four fields: **String** fields **title** and **category**, a **float** field **cost** and an **ArrayList** of **authors**. You will want to make these fields private, with public accessor methods for all but the authors field

```
public class Book {

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

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

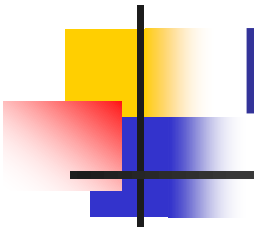
Lab's content

- **2. Creating the Book class**
  - Instead of typing the accessor methods for these fields, you may use the **Generate Getter and Setter** option in the **Outline** view pop-up menu

```
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;
}

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

public void setAuthors(List<String> authors) {
    this.authors = authors;
}
```

Lab's content

- **2. Creating the Book class**
  - Next, create **addAuthor**(**String authorName**) and **removeAuthor**(**String authorName**) for the **Book** class
    - The **addAuthor**(...) method should ensure that the author is not already in the ArrayList before adding
    - The **removeAuthor**(...) method should ensure that the author is present in the ArrayList before removing
    - Reference to some useful methods of the **ArrayList** class



Lab's content

- **3. Creating the Media class**
  - At this point, the **DigitalVideoDisc** and the **Book** classes have some fields in common namely title, category and cost.
    - Here is a good opportunity to create a common superclass between the two, to eliminate the duplication of code. This process is known as *refactoring*.
  - You will create a class called **Media** which contains these three fields and their associated get and set methods.

Lab's content

- **3. Creating the Media class**
  - In the Package Explorer view, right-click the project and select New -> Class. Adhere to the following specifications:
    - Package: **hust.soict.aims.media**
    - Name: **Media**
    - Access modifier: **public**
    - Superclass: **java.lang.Object**
    - **public static void main(String[] args)**: **do not check**
    - Constructors from Superclass: **Check**
    - All other boxes: **Do not check**

Lab's content

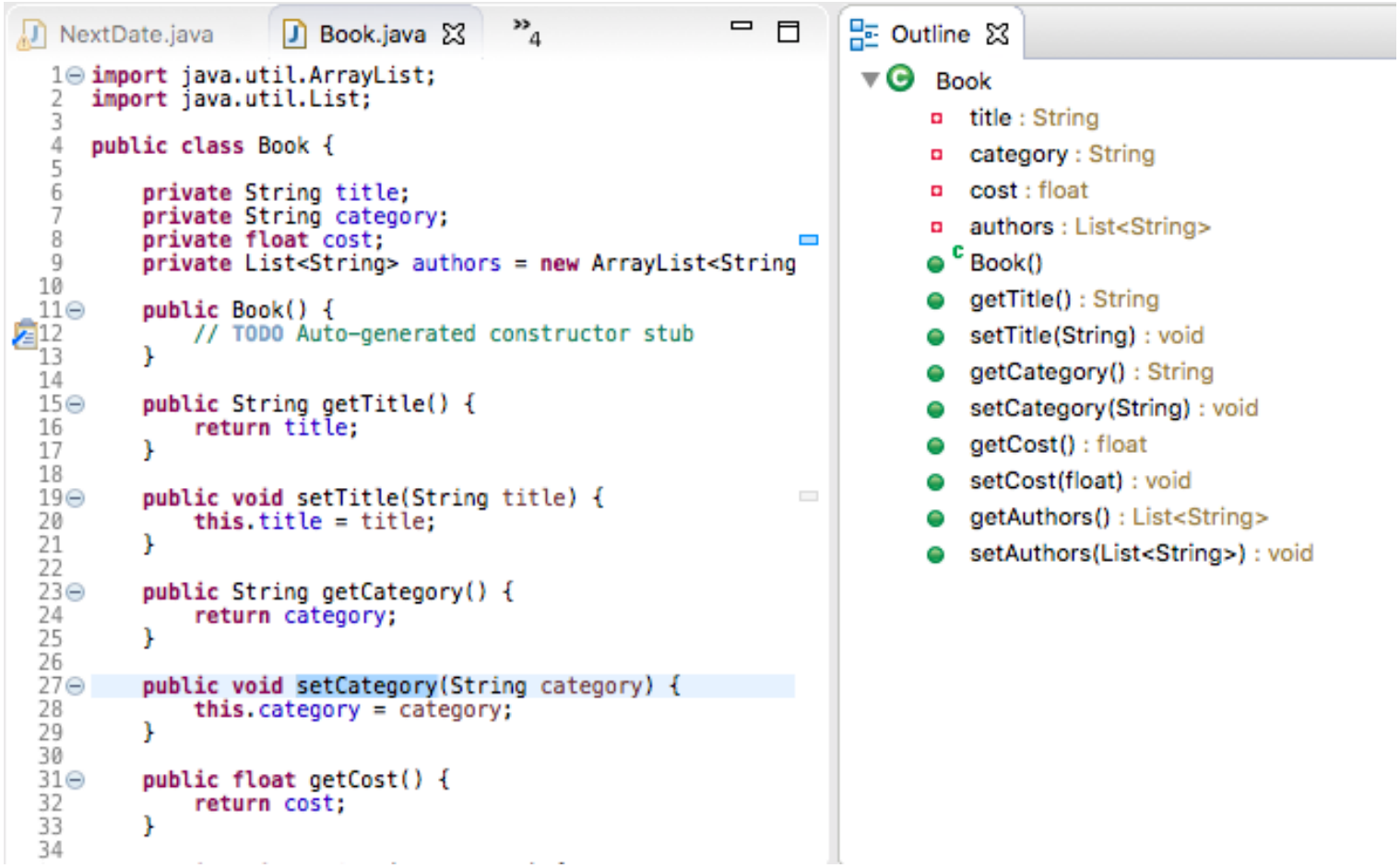
- **3. Creating the Media class**
  - Add fields to the **Media** class
  - To store the information common to the **DigitalVideoDisc** and the **Book** classes, the **Media** class requires three fields: **String title**, **String category** and **float cost**
  - You will want to make these fields private with public accessor methods (by using **Generate Getter and Setter** option in the **Outline** view pop-up menu)

Lab's content

- 3.2 Remove fields and methods from **Book** and **DigitalVideoDisc** classes
  - Open the Book.java in the editor
  - Locate the Outline view on the right-hand side
  - Select the fields title, category, cost and the methods getCategory(), getCost(), getTitle(), setTitle(), setCategory(), setCost()
  - Right click the selection and select Delete from the pop-up menu
  - Save your changes

Lab's content

- 3.2 Remove fields and methods from **Book** and **DigitalVideoDisc** classes



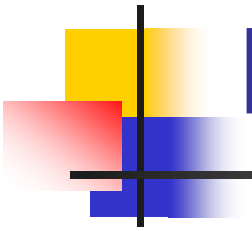

```
1  import java.util.ArrayList;
2  import java.util.List;
3
4  public class Book {
5
6      private String title;
7      private String category;
8      private float cost;
9      private List<String> authors = new ArrayList<String
10
11     public Book() {
12         // TODO Auto-generated constructor stub
13     }
14
15     public String getTitle() {
16         return title;
17     }
18
19     public void setTitle(String title) {
20         this.title = title;
21     }
22
23     public String getCategory() {
24         return category;
25     }
26
27     public void setCategory(String category) {
28         this.category = category;
29     }
30
31     public float getCost() {
32         return cost;
33     }
34
```

Lab's content

- 3.2 Remove fields and methods from **Book** and **DigitalVideoDisc** classes
  - Do the same for the **DigitalVideoDisc** class, move it to the package **hust.soict.aims.media**.
  - After doing that you will see alot of errors because of the missing fields
  - Extend the Media class for both Book and DigitalVideoDisc
    - **public class Book extends Media**
    - **public class DigitalVideoDisc extends Media**
  - Save your changes.

Lab's content

- **4. Update the Order class to work with Media**
  - You must now update the **Order** class to accept both **DigitalVideoDisc** and **Book**. Currently, the **Order** class has methods:
    - **addDigitalVideoDisc**()
    - **removeDigitalVideoDisc**().
  - You could add two more methods to add and remove **Book**, but since **DigitalVideoDisc** and **Book** are both subclasses of type **Media**, you can simply change **Order** to maintain a collection of **Media** objects.
  - So you can add either a **DigitalVideoDisc** or a **Book** using the same methods.



Lab's content

- **4.1** Remove the **itemsOrdered** array, as well as its add and remove methods.
  - From the **Package Explorer** view, expand the project
  - Double-click **Order.java** to open it in the editor
  - In the **Outline** view, seclect the **itemsOrdered** array and the methods **addDigitalVideoDisc**() and **removeDigitalVideoDisc**() and hit the **Delete** key
  - Click **Yes** when prompted to confirm the deletion

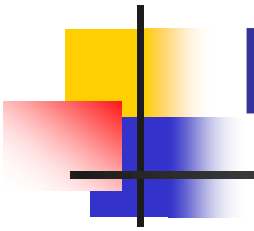
Lab's content

- **4.2** The **qtyOrdered** field is no longer needed since it was used to track the number of **DigitalVideoDiscs** in the **itemsOrdered** array, so remove it and its accessors: **getQtyOrdered**() and **setQtyOrdered**().
  - Add the **itemsOrdered** to the **Order** class
    - Recreate the **itemsOrdered** field, this time as a **ArrayList** instead of an array.
    - To create this field, type the following code in the **Order** class, in place of the **itemsOrdered** array declaration that you deleted:

```
private ArrayList<Media> itemsOrdered = new
ArrayList<Media>();
```

Lab's content

- **4.3** Note that you should import the **java.util.ArrayList** in the **Order** class
  - A quicker way to achieve the same affect is to use the Organize Imports feature within Eclipse
  - Right-click anywhere in the editor for the Order class and select Source -> Organize Imports (Or Ctrl+Shift+O). This will insert the appropriate import statements in your code.
  - Save your class
- Create **addMedia**() and **removeMedia**() to replace **addDigitalVideoDisc**() and **removeDigitalVideoDisc**()
- Update the **totalCost**() method



Lab's content

- **5. Constructors of whole classes and parent classes**
  - Draw a UML class diagram (using Astah UML – please request a free license for students) for the **AimsProject**? Attach the design image in the **design** sub-folder in the **resources** folder of the project (create new folders if necessary).
  - Which classes are aggregates of other classes? Checking all constructors of whole classes if they initialize for their parts?
  - Write constructors for parent and child classes. Remove redundant setter methods if any.

Lab's content

- For example:

In Media class (superclass)

```
Media(String title){
      this.title = title;
}
Media(String title, String category){
      this(title);
      this.category = category;
}
```

In Book class:

```
Book(String title){
      super(title);
}
Book(String title, String category){
      super(title, category);
      //…
}
Book(String title, String category, List<String> authors){
      super(title, category);
      this.authors = authors;
}
```

Lab's content

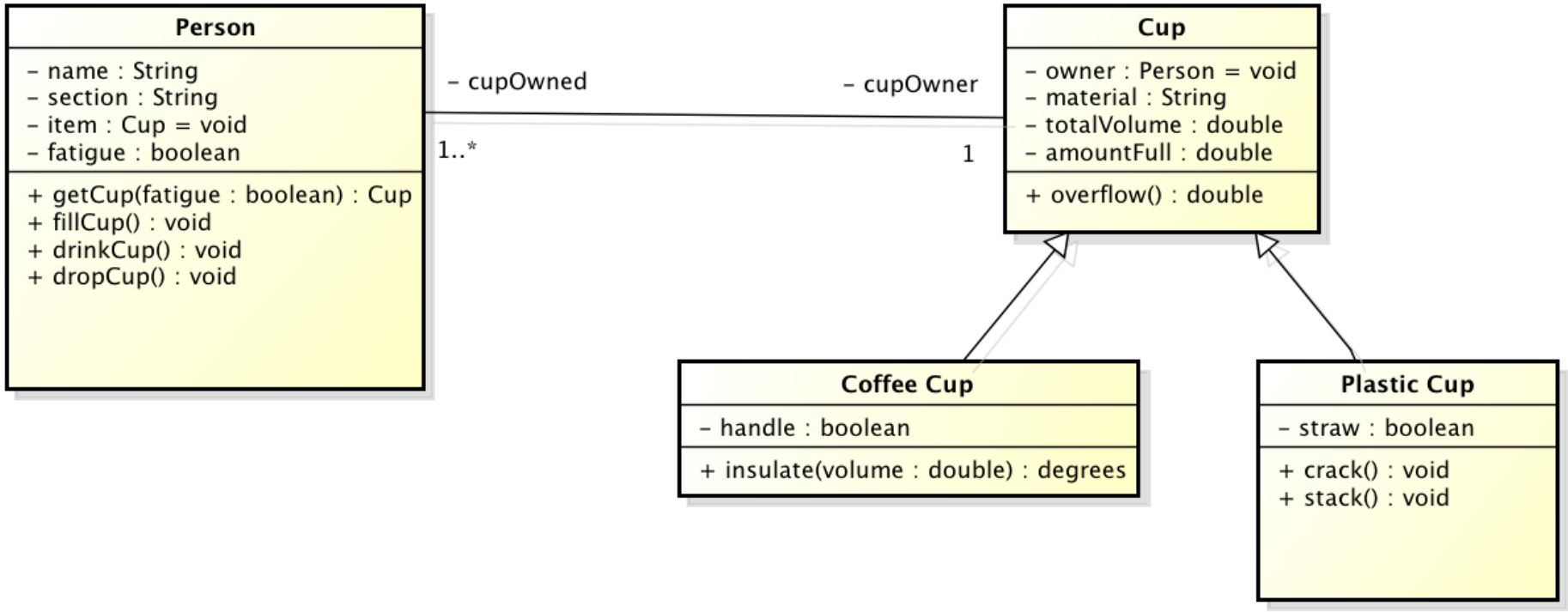
- **6. Create a complete console application in the Aims class**
  - Open the Aims class. You will create a prompted menu as following:

```
public static void showMenu() {
        System.out.println("Order Management Application: ");
        System.out.println("-----------------------------");
        System.out.println("1. Create new order");
        System.out.println("2. Add item to the order");
        System.out.println("3. Delete item by id");
        System.out.println("4. Display the items list of order");
        System.out.println("0. Exit");
        System.out.println("-----------------------------");
        System.out.println("Please choose a number: 0-1-2-3-4");
}
```

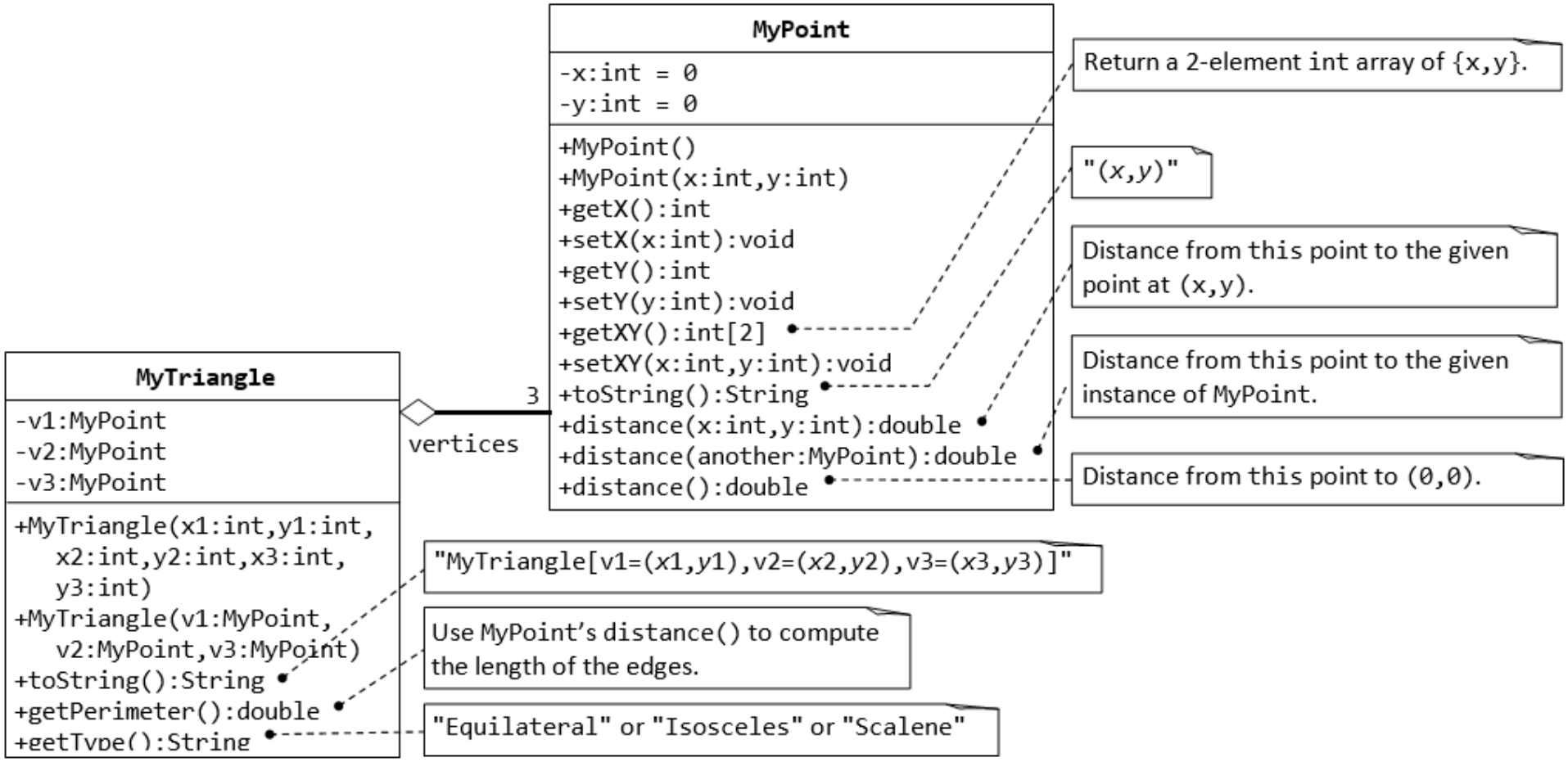
  - You will modify classes **Media**, **Order** by adding more field **id**. Then create a complete application that allows to user to interact with through the above menu. Please use corresponding constructors to create objects.

Homeworks

- 1. Use Astah Draw the class diagram as follows and export it to an image



- 2. Use Astah Draw the class diagram as follows and export it to an image, then write the source code in java language



- 3. Use Astah Draw the class diagram as follows and export it to an image file, then write the source code in java language