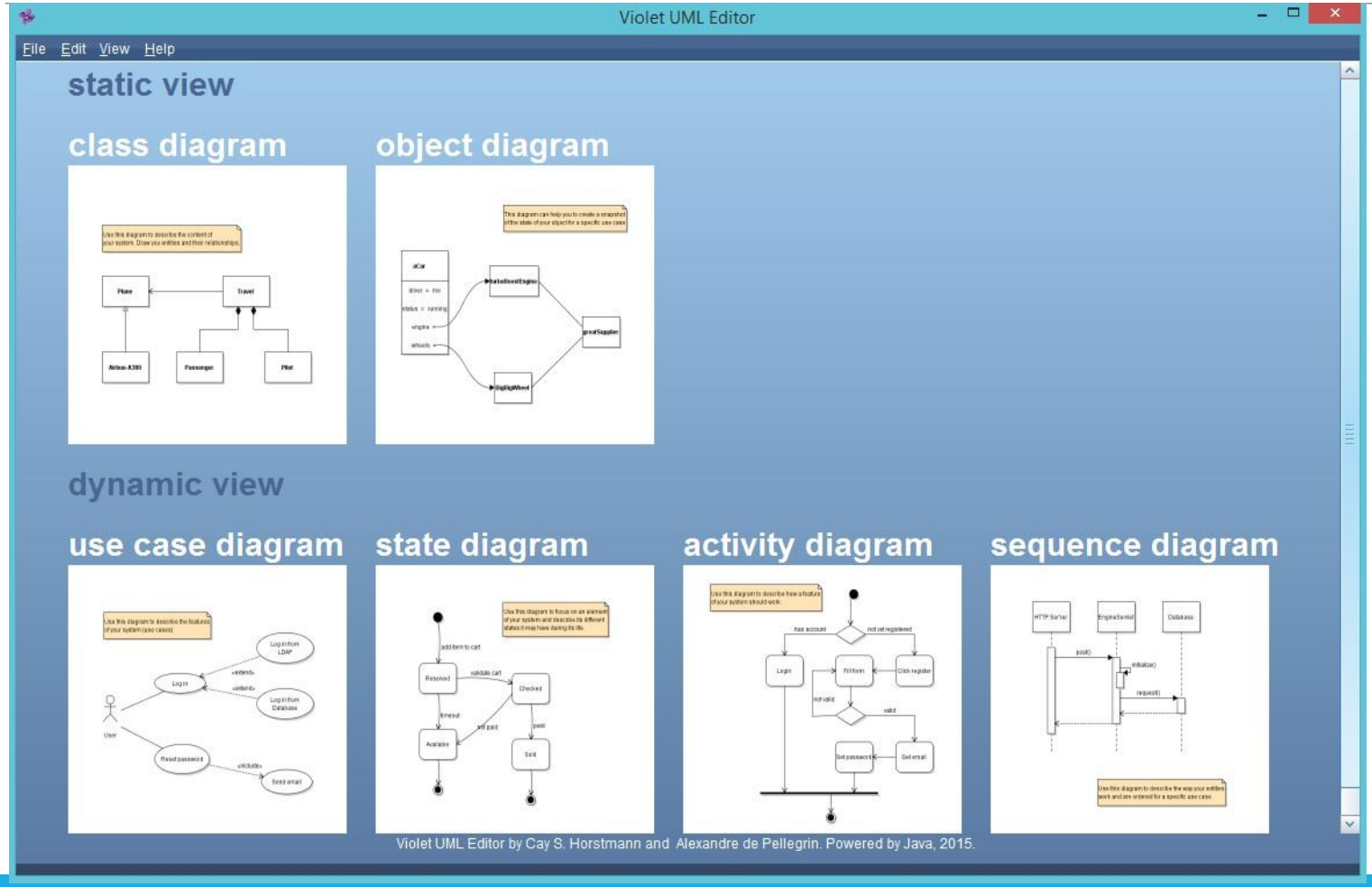
Java Programming AP Edition
U3R3 Review on Unit 3

OBJECT-ORIENTED PROGRAMMING PLANNING USING VIOLET UML EDITOR (CLASS DIAGRAM)

ERIC Y. CHOU, PH.D. IEEE SENIOR MEMBER

Violet UML Editor is a Free Software



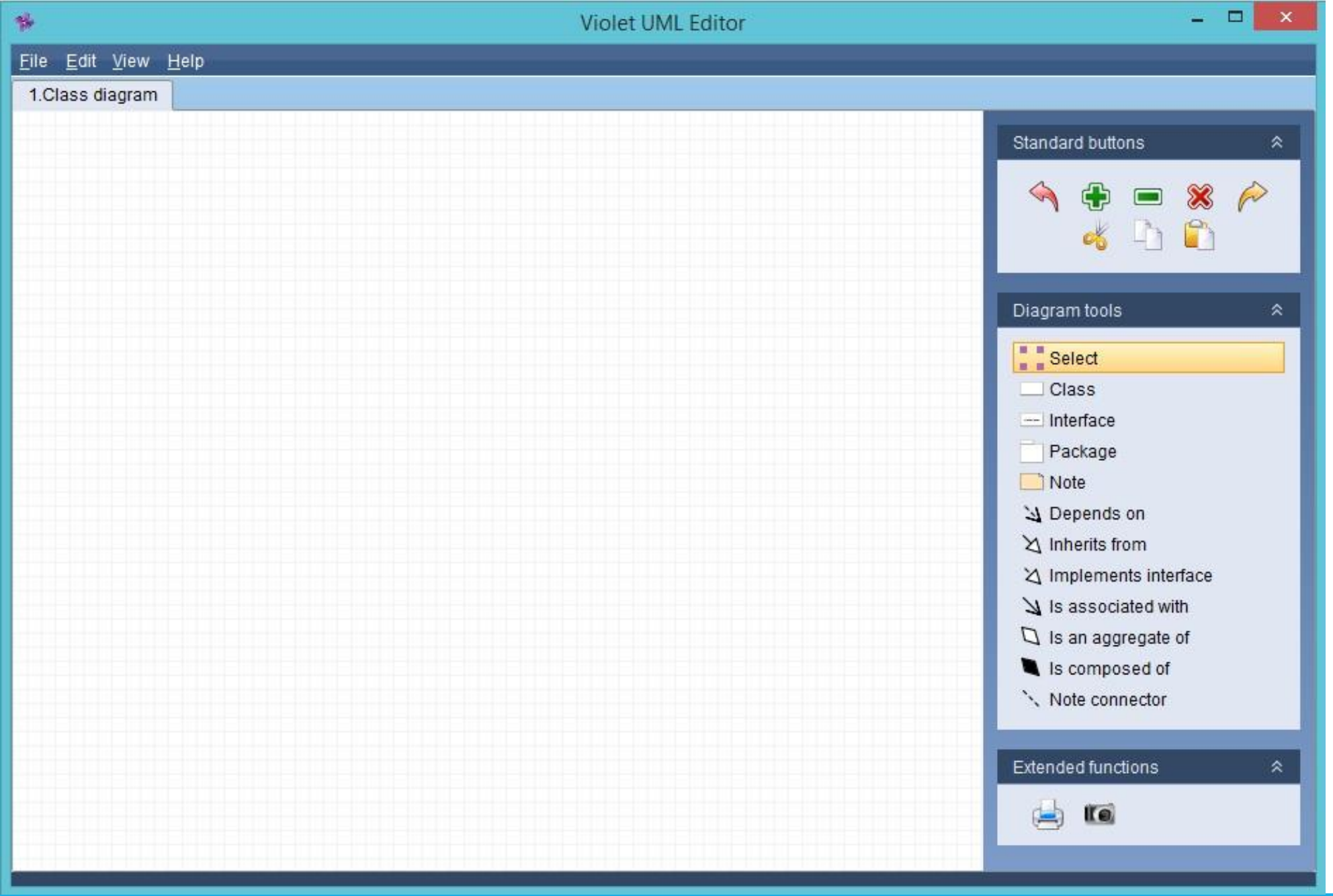
Static View:

- Class Diagram
- Object Diagram

Dynamic View:

- Use Case Diagram
- State Diagram
- Activity Diagram
- Sequence Diagram

By Cay S. Horstmann and Alexandre de Pellegrin. Special thanks to the developer and other supporters.



1st Row: L-R:

Undo

Zoom In

Zoom Out

Delete

Redo

2nd Row: L-R:

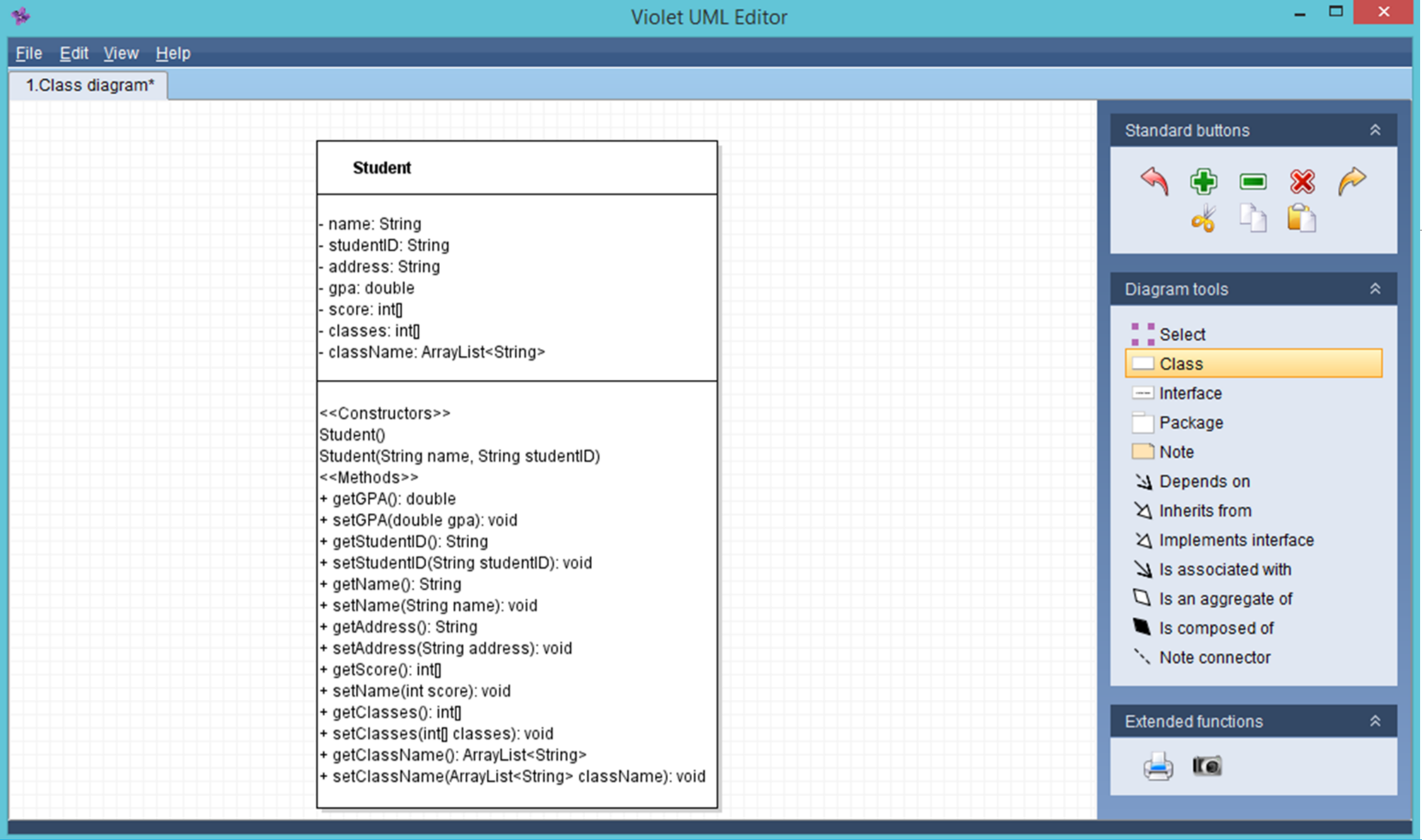
Cut

Copy

Paste

Your first diagram

Now, let's look at diagrams. A **diagram** is composed of nodes (for example, classes or objects) and **edges** (for example, inheritance arrows or call arrows). To add a node, ***first click on the matching UML tool on the side bar (on the right), then click anywhere on the diagram to add it.*** You can also use your mouse scroll button to select another UML tool. To connect two nodes with an edge, first select an edge tool. Then click on the first node and hold down the left mouse button. Drag the mouse to the second node and release the mouse button. The edge is inserted between the two nodes. There is one exception : You can insert a “note connector” edge simply by dragging from a “note” node to anywhere on the diagram.


Violet UML Editor
File Edit View Help
1.Class diagram*
Student
- name: String
- studentID: String
- address: String
- gpa: double
- score: int[]
- classes: int[]
- className: ArrayList<String>
<<Constructors>>
Student()
Student(String name, String studentID)
<<Methods>>
+ getGPA(): double
+ setGPA(double gpa): void
+ getStudentID(): String
+ setStudentID(String studentID): void
+ getName(): String
+ setName(String name): void
+ getAddress(): String
+ setAddress(String address): void
+ getScore(): int[]
+ setName(int score): void
+ getClasses(): int[]
+ setClasses(int[] classes): void
+ getClassName(): ArrayList<String>
+ setClassName(ArrayList<String> className): void
Standard buttons
Diagram tools
Select
Class
Interface
Package
Note
Depends on
Inherits from
Implements interface
Is associated with
Is an aggregate of
Is composed of
Note connector
Extended functions

Other UML Tools

Violet UML is free. It is not the best but good for students.

We will be using this tool often in Part 2 of this course regarding package, inheritance, abstraction, interface, use, aggregation, and many other topics about Object-Oriented Programming.

Class Diagram is a static view of OOP. Object Diagram is a dynamic view of OOP. (It is a advanced version of **Data Flow Diagram**.)

Other 4 diagrams are models of code. They are not the essential part of OOP but used for general programming.

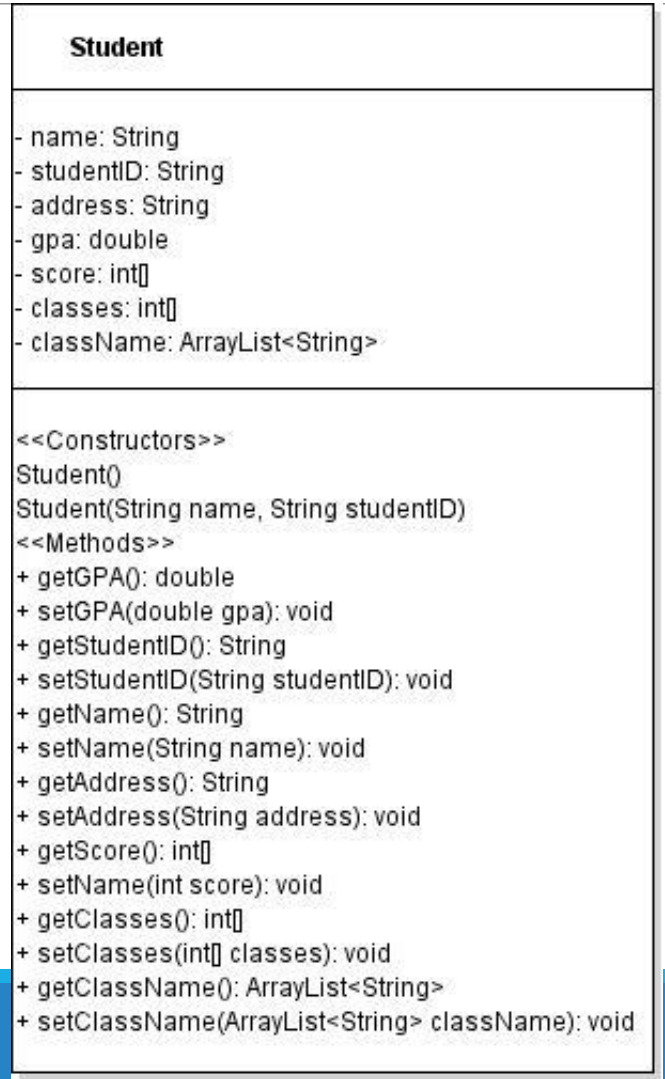
We will learn to use this tool as a documentation tool along the advancement of course works in part 2(B).

Save as (violet file and image file)

To save as violet file, click File=> save as => browse through the directory.

The design will be saved as a html file.

To export to an image file. File=> export. (The output .jpg file is on the right for Student class.



Our Unit Project

PokerGame (4-player Text Mode Poker)

What we have developed so far about poker game?

(1) DeckOfCards.java: simple Card and Deck in array, one random shuffle method.

(2) DeckOfCardsImage.java: DeckOfCards in simple GUI setup.

(3) DeckOfCardsShuffle.java: Deck of Cards in more shuffling methods and distribution to hands.

(4) DeckOfCardsPoker.java: Deck of Cards with ASCII representation and finding the category of a hand of cards.

(5) Card.java: Conversion of Card representation from int to an object.

(6) Deck.java: Conversion of Deck representation from int[] array to an object of Card[].

(7) Hand.java: Conversion of Hand representation from int[] array to an object of Card[].

(8) PokerOnePlayer.java: Addition of valueOf method for calculation of strength code for a hand of cards.

(9) DeckView.java and HandView.java: DeckView.java (GUI for a Deck), HandView.java (GUI for a Hand of cards,

(10) PokerGame.java (Unit Project): A complete text mode poker game for four people.

Software Development Knowledge We Learn from this Series of Programs.

(1) Top Down design.

(2) Bottom Up Implementation.

(3) Stepwise Refinement.

(4) Using static methods for structural programming paradigm.

(5) Using classes and objects for object-oriented programming.

(6) Divide and Conquer of a bigger software project.

Techniques used in this Series of Projects

(1) 1-D/2-D arrays.

(2) Shuffling, Reverse, Sorting, Block Sorting, Split and Merge algorithms.

(3) do-while-loop for text-mode menu design.

(4) Using program handler to convert code in static method to instance methods (hand.analyze()) as a way to grow a class.

(5) selection sort algorithm with availability array for small data size and non-destructive sort.

(6) occurrence array (arraylist) generation for strength of a hand (**and word count**).

(7) game board evaluation (strength code by **arank** array as a simple way to evaluate game status). This is very important when we get involved with more advanced chess game or poker game such as AI-based computer program to play games (at least tic-tac-toe game).