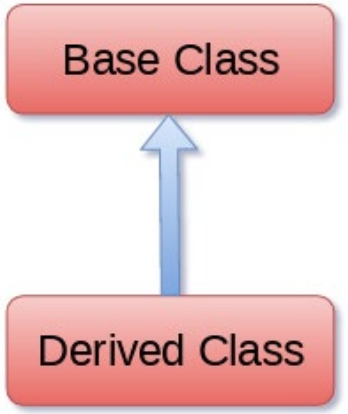
Inheritance – Wide and Narrow

Agenda

- Inheritance Overview: (Initially I'll give an overview of what is inheritance in brief) - 3 mins
- Wide and Narrow inheritance: (Will explain in detail what is Wide and Narrow inheritance with an example, which will have an UML diagram, and its benefits) – 5 mins
- Code walk through: of Python Code on a simple TextEditor where Wide and Narrow inheritance are implemented. – 5 mins
- Industry experience: on Wide and Narrow inheritance with an appropriate UML diagram (7 mins)

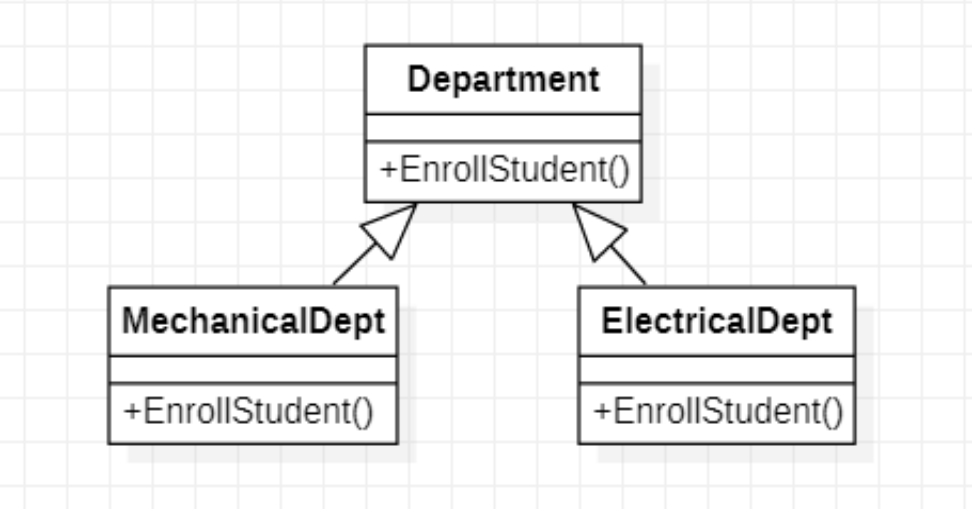
Inheritance - Overview

- **Inheritance** is an important aspect of the object-oriented paradigm. Inheritance is a mechanism of acquiring the features and behaviors of a class by another class.
- The child class acquires the properties and can access all the data members and functions defined in the parent class.
- A child class can also provide its specific implementation to the functions of the parent class.
- Inheritance implements the IS-A relationship.
- It increases re-usability by implementing the common functionality in the base class.



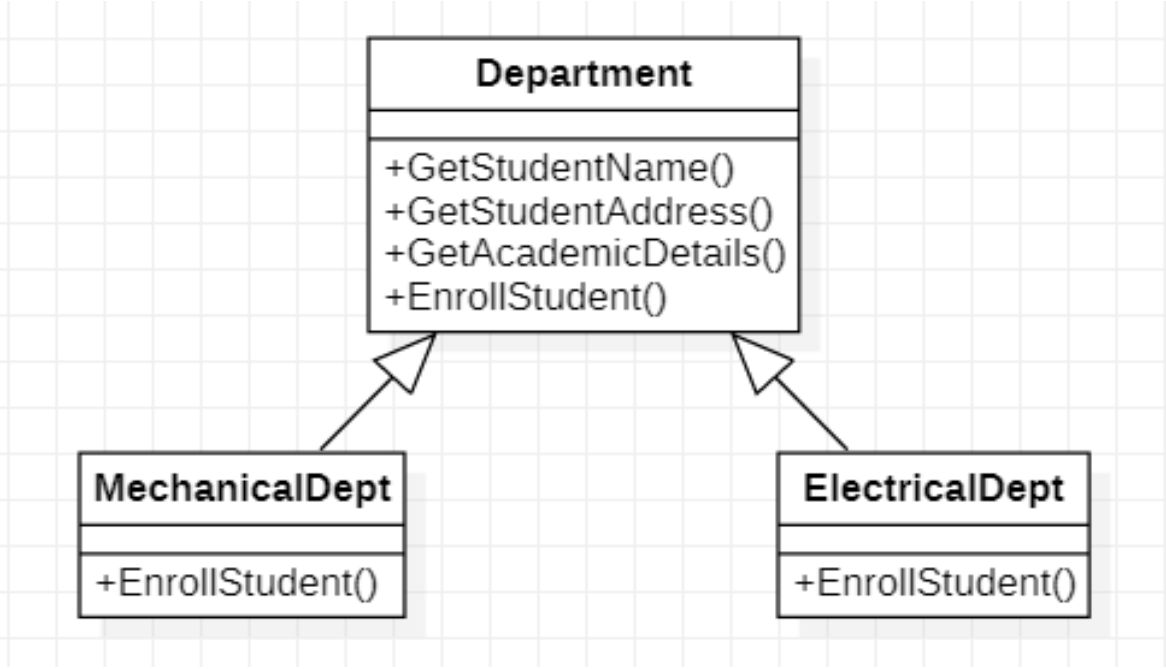
Wide inheritance

- While inheriting methods of base class, if the methods are too large, then it is called as **Wide** inheritance.
- For ex., in the following class the college enrolls candidates for different departments.
- EnrollStudent method takes the input of student info, his academic details and allocates subjects based on the department choice opted by the student, in the respective departments
- The above method is quite large and has multiple smaller responsibilities even though the final goal of the method is to enroll the students.



Narrow inheritance

- Inheriting a smaller method from base class and overriding it in the derived class is called as **Narrow** inheritance.
- In the example the Enrollstudent is split into following smaller common function which is applicable to all departments
  - GetStudentName
  - GetStudentAddress and
  - GetAcademicDetails
- The EnrollStudents now just allocates the subjects which is differing behavior for the respective departments.

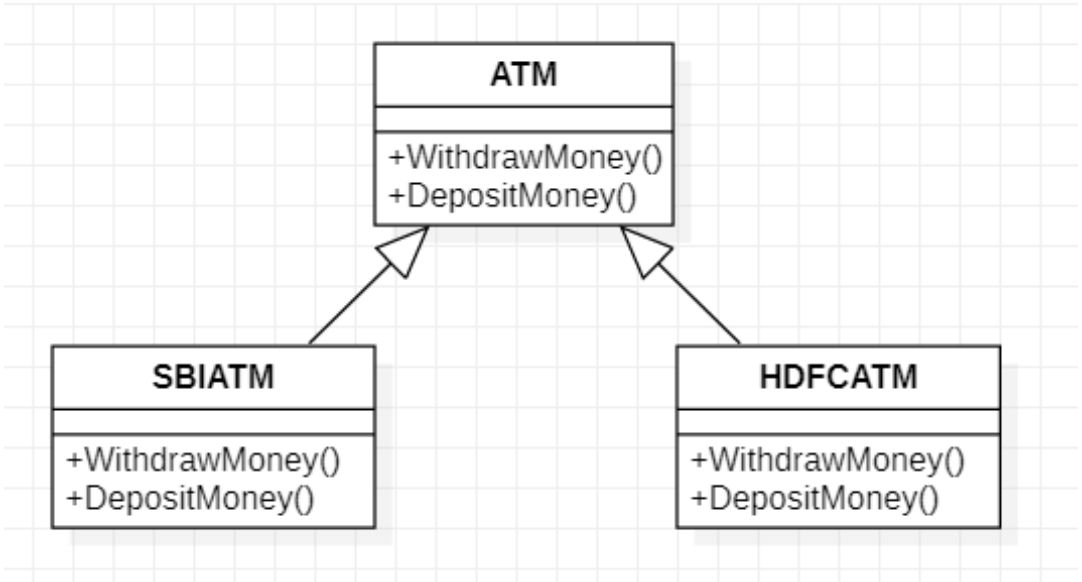


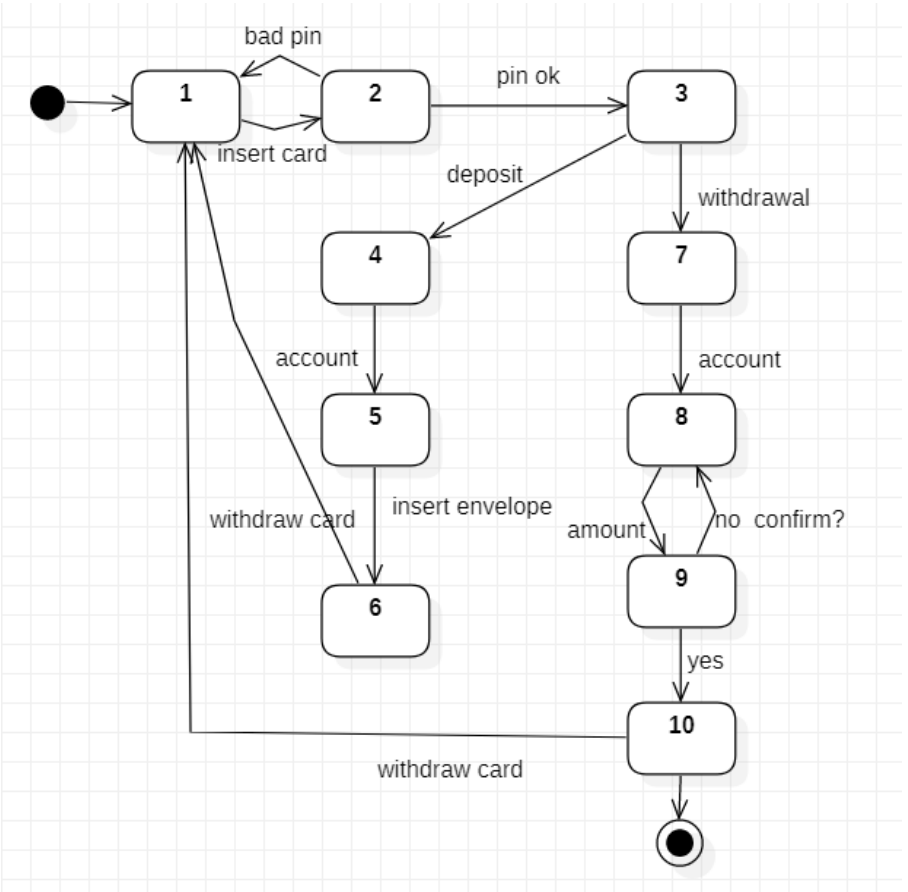
Narrow inheritance - Benefits

- It is always better to have narrow inheritance i.e., overridden smaller functions because of the following benefits:
  - Better Readability
  - Better Cohesion (SRP)
  - Better Maintainability
  - Reduces Code Duplication
  - Supports Code Extensibility
  - Enhances Flexibility

Wide and Narrow inheritance – ATM Example

WithdrawMoney and DepositMoney are huge functions which will have the following sequence of tasks to be performed





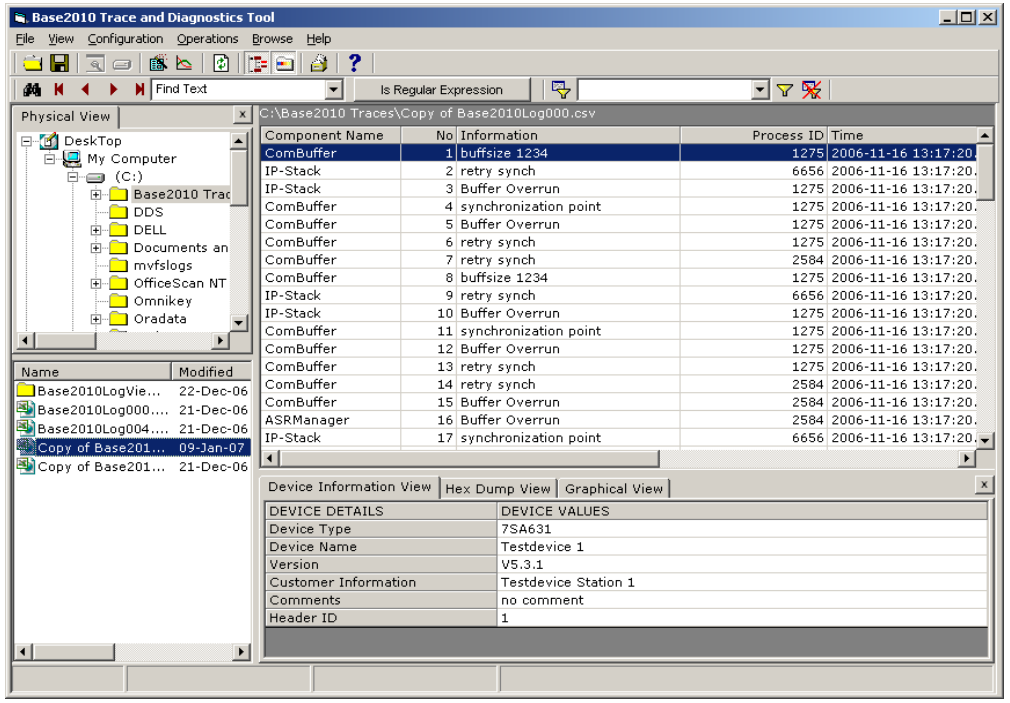
Wide and Narrow inheritance – ATM Example

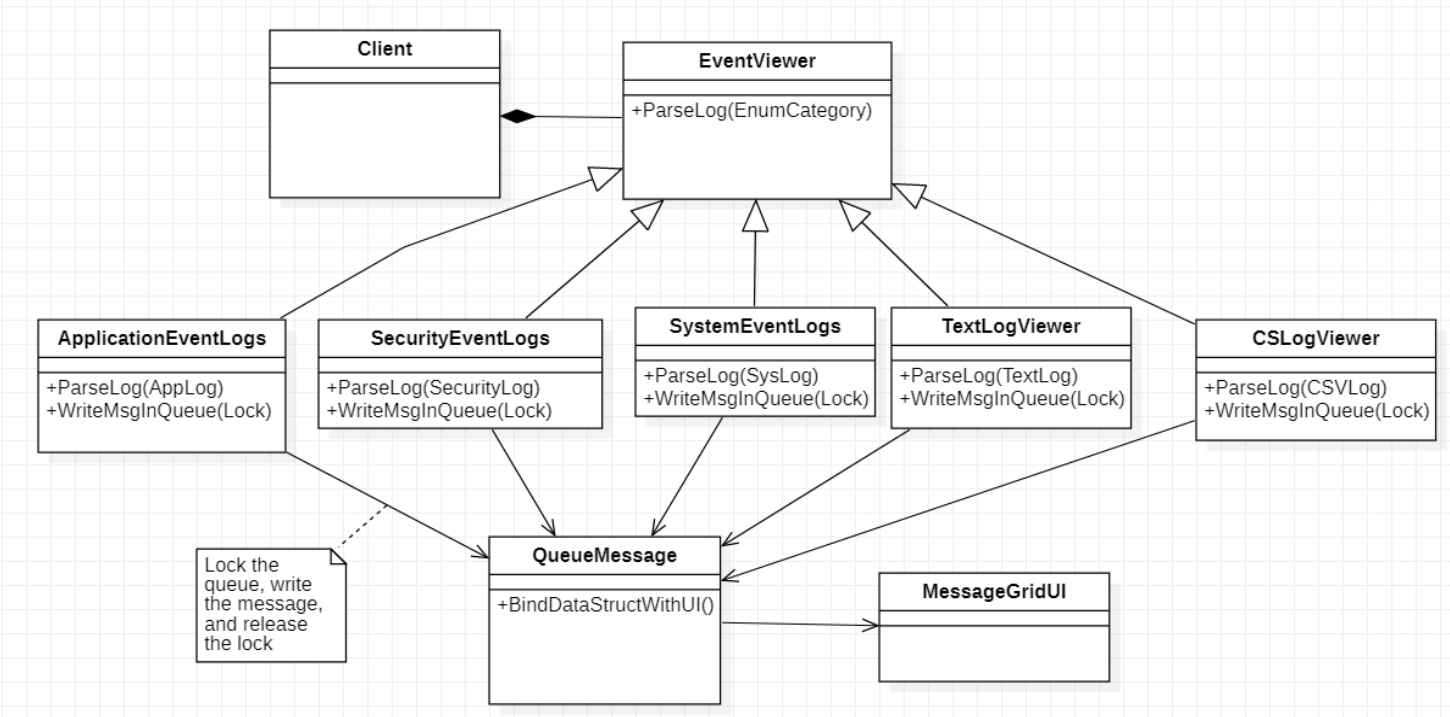
- Split WithdrawMoney and DepositMoney into small functions which has common code
- Move the common to base class.
- Keep only the varying behavior in the specific class
  - This provides the reusability of the code
  - Provides flexibility of changing the tasks (Some ATMs first allows to withdrawcard before DispensingCash)
  - Enhances Readability and Maintainability of the code
  - SRP is followed which enhances the cohesion
  - Can be extended easily with more features

Task sequence	WithdrawMoney	DepositMoney
1	CardInsertIdentification	CardInsertIdentification)
2	AskAndVerifyPin	AskAndVerifyPin
3	CashWithdraw (**child class implementation**)	CashDeposit (**child class implementation**)
4	UpdateAccount (**child class implementation**)	OpenDispenser
5	OpenDispenser	AcceptEnvelope (**child class implementation**)
6	DispenseCash (**child class implementation**)	PrintAcknowledgement (**child class implementation**)
7	PrintReceipt (**child class implementation**)	WithdrawCard
8	WithdrawCard	

Wide and Narrow inheritance – Industry Experience

LogViewer: Design before re-factoring for Narrow Inheritance





Sequence of operation of functions:

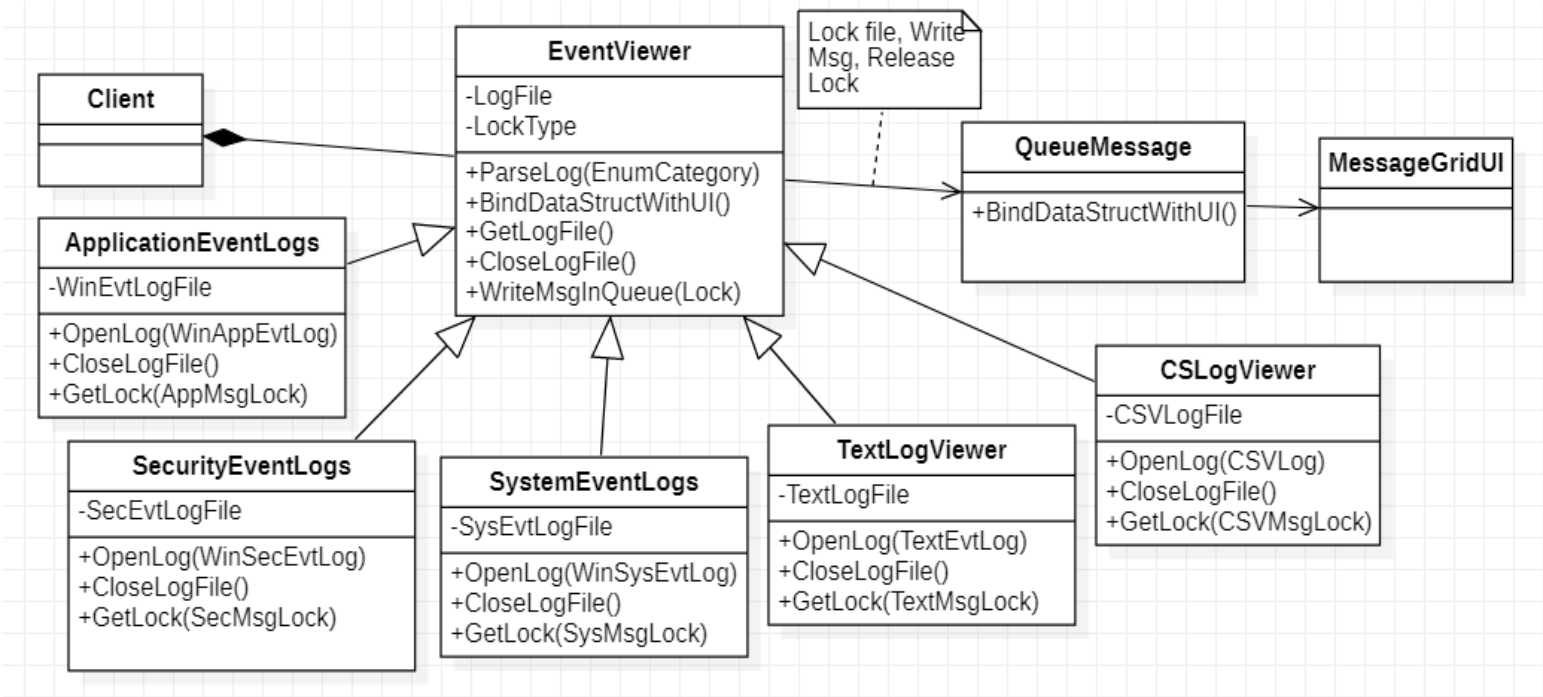
ParseLog: Open Log file ➡ Read log into a memory buffer ➡ Parse and fill the Data structure which is common across all Logs

WriteMessageInQueue: Lock the Queue ➡ Write msg in Queue ➡ Release the Queue

Wide and Narrow inheritance – Industry Experience

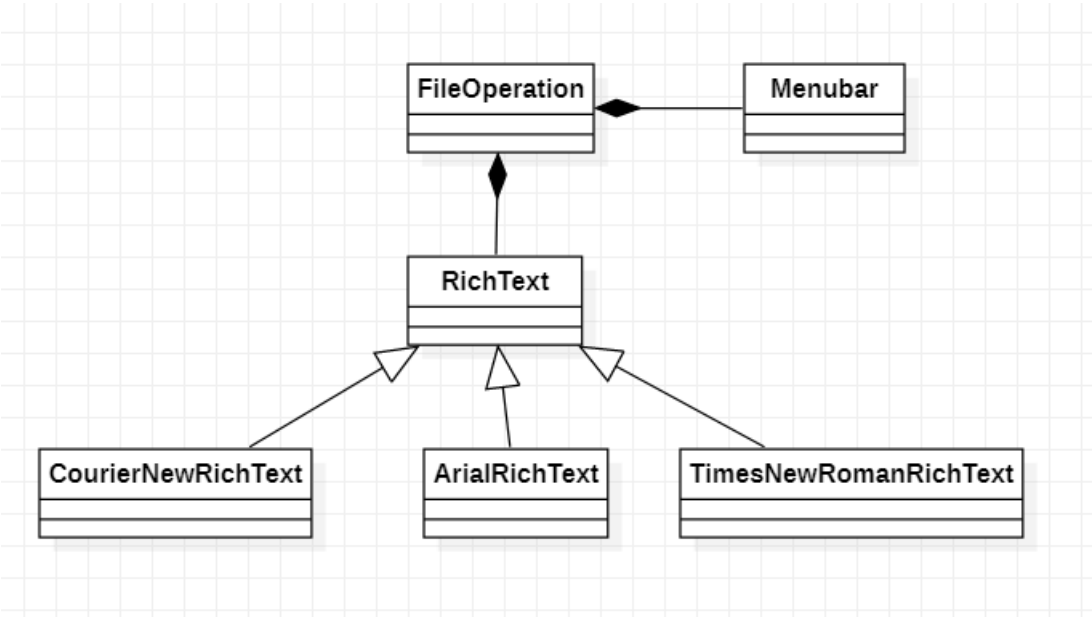
LogViewer: Design after re-factoring for Narrow Inheritance

- Common logic of parsing the buffer is moved to the base class.
- Common logic of locking, Unlocking and Writing in the Queue is abstracted to the base class.



Text Editor – Python code

- This simple TextEditor uses the following library and classes.
  - **Tkinter library**: Tkinter is the standard GUI library for Python which provides easy and powerful interfaces for creating GUI applications.
  - **Class FileOperation**: This class is responsible for all the file operations like creating new file, opening and saving files, setting titles of file etc.
  - **Class Menubar**: This class is responsible for creating the menus for file, setting up fonts and face of the text and keeps track of the text status.
  - Classes **RichText**, **CourierNewRichText**, **ArialNewRichText**, **TimesNewRomanRichText** manage the fonts



Questions?

Thank you!!