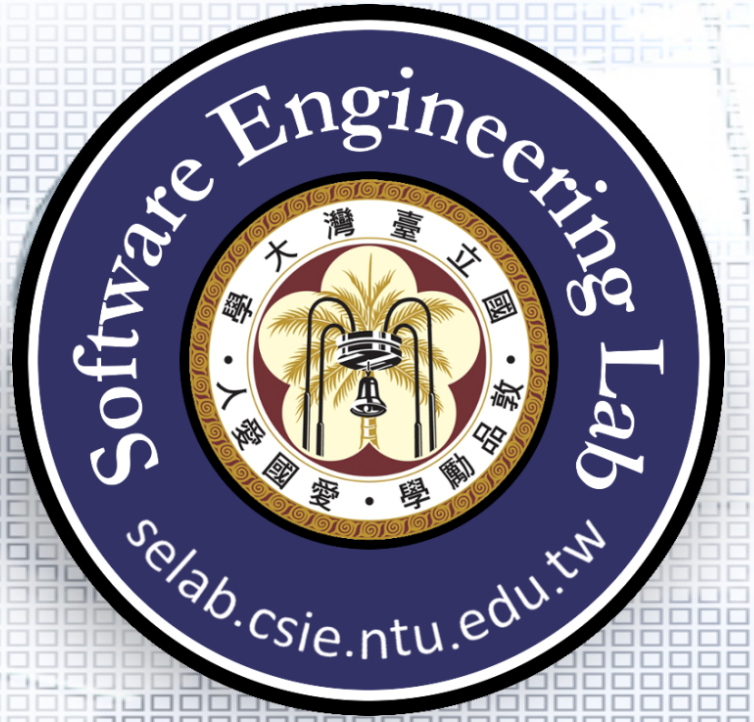
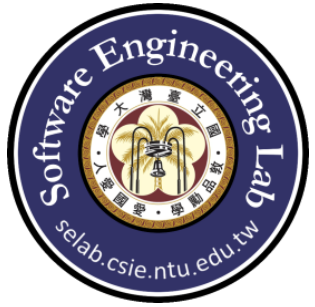


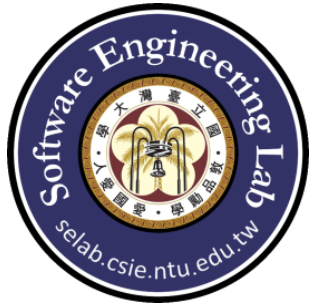
Façade Pattern

Prof. Jonathan Lee（李允中）
Department of CSIE
National Taiwan University



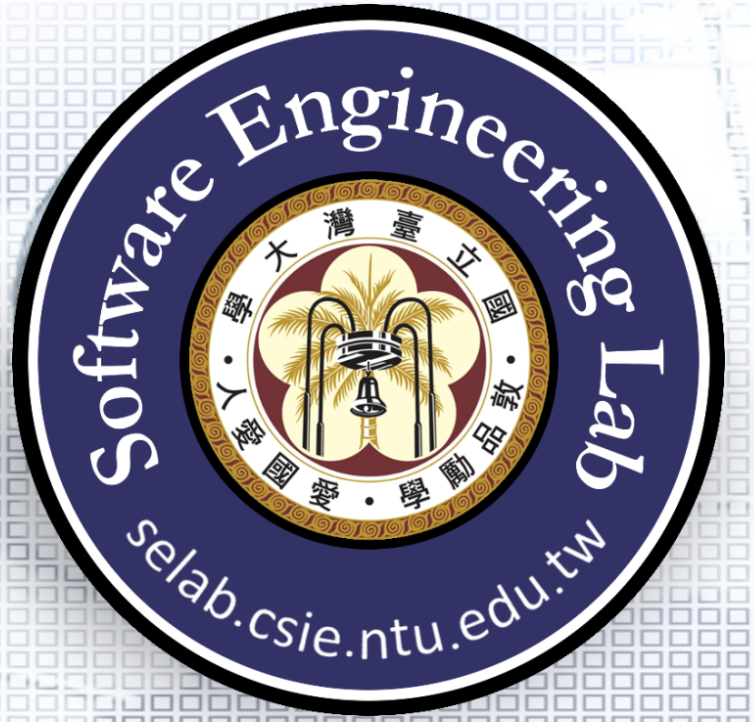
Design Aspect of Facade

Interface to a subsystem



Outline

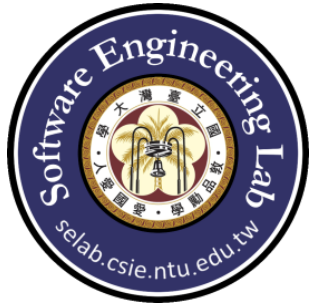
- A Programming Environment Requirements Statements
- Initial Design and Its Problems
- Design Process
- Refactored Design after Design Process
- Recurrent Problems
- Intent
- Façade Pattern Structure
- Home Sweet Home Theater: Another Example



A Programming Environment

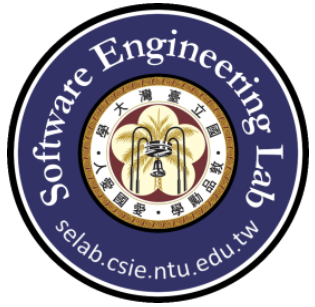
Prof. Jonathan Lee (李允中)

Department of Computer Science and Information Engineering
National Taiwan University



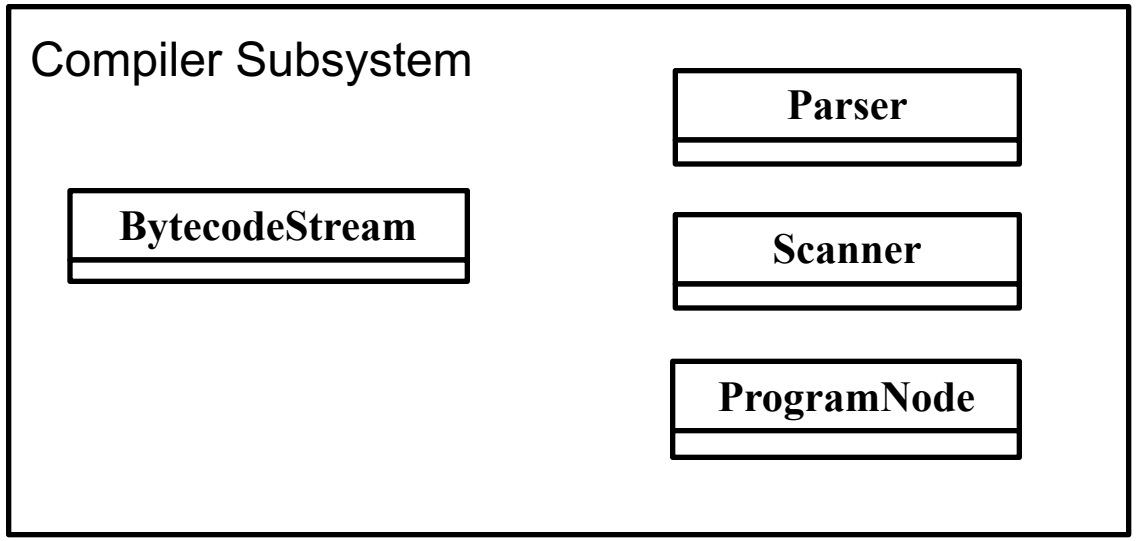
Requirements Statement

- A compiler subsystem contains classes such as Scanner, Parser, ProgramNode, and BytecodeStream.
- The client classes need to use Scanner, Parser, ProgramNode, and BytecodeStream to compile some code.



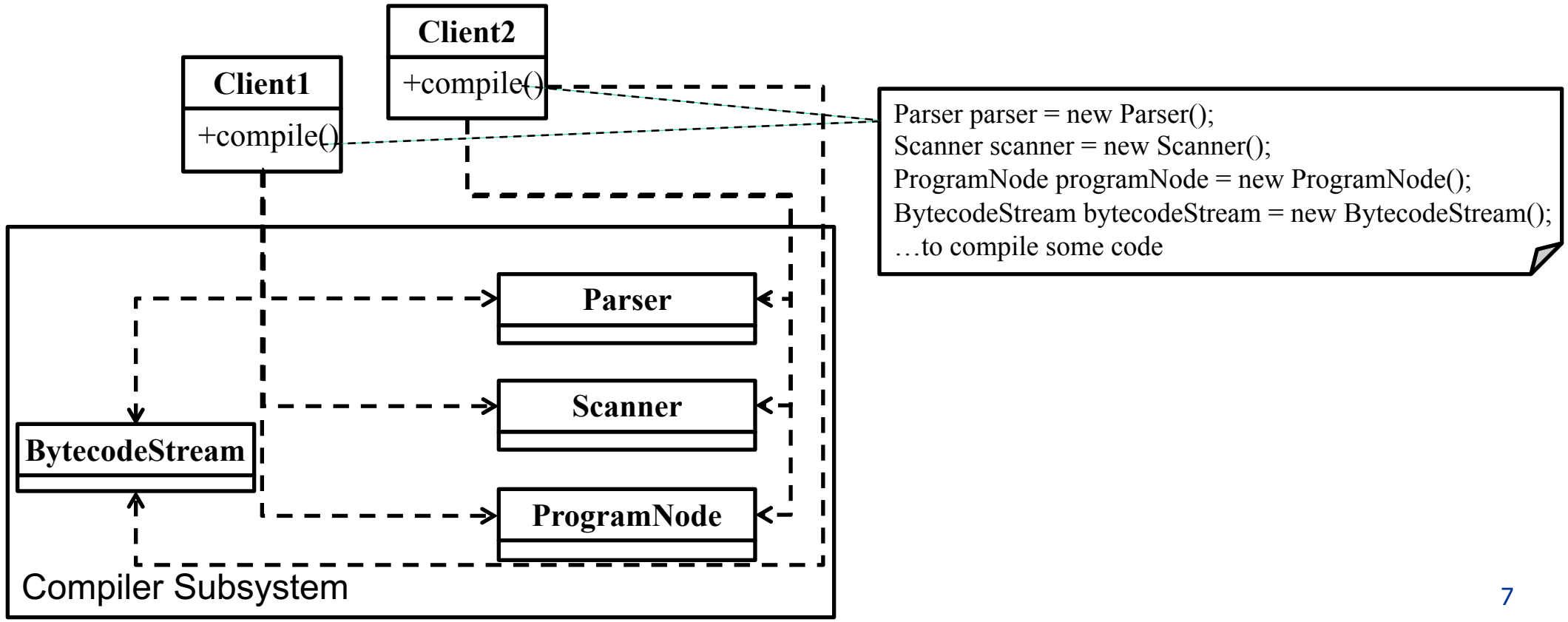
Requirements Statements$_1$

- A compiler subsystem contains classes such as Scanner, Parser, ProgramNode, and BytecodeStream.

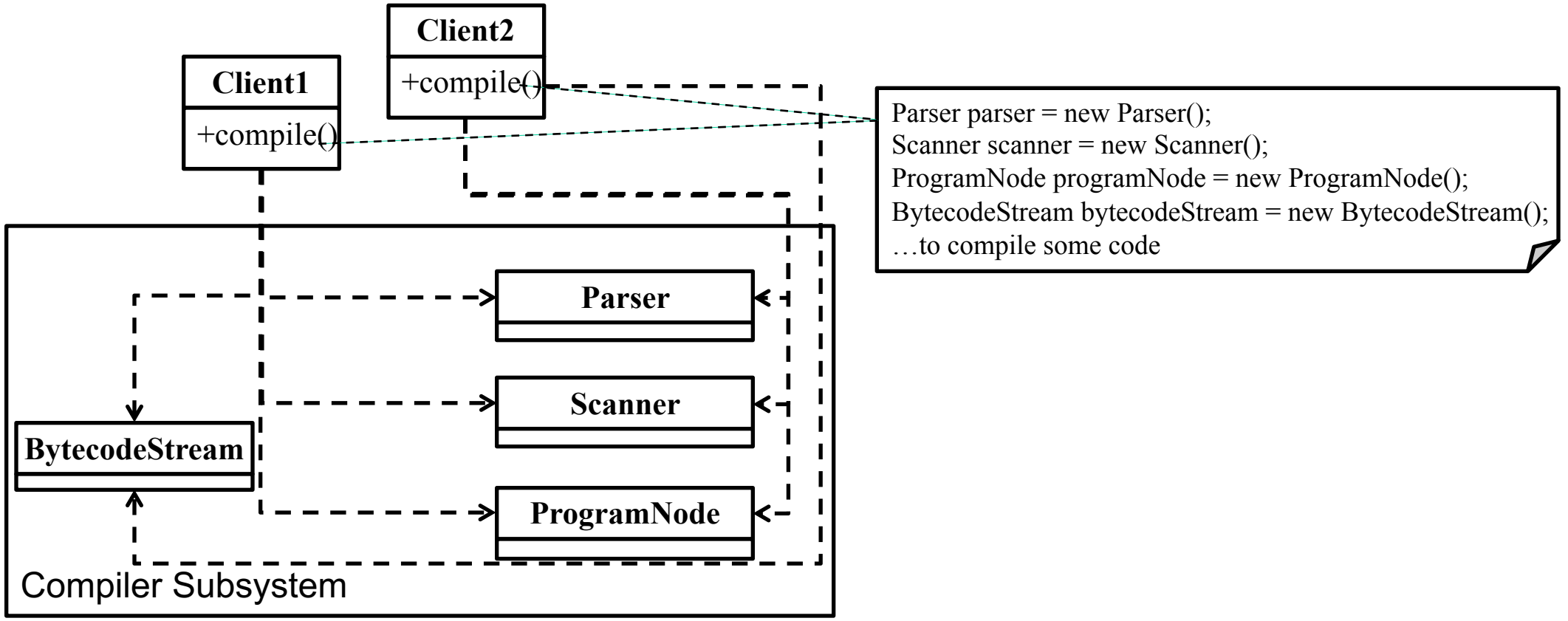


Requirements Statements$_2$

- The client classes need to use Scanner, Parser, ProgramNode, and BytecodeStream to compile some code.



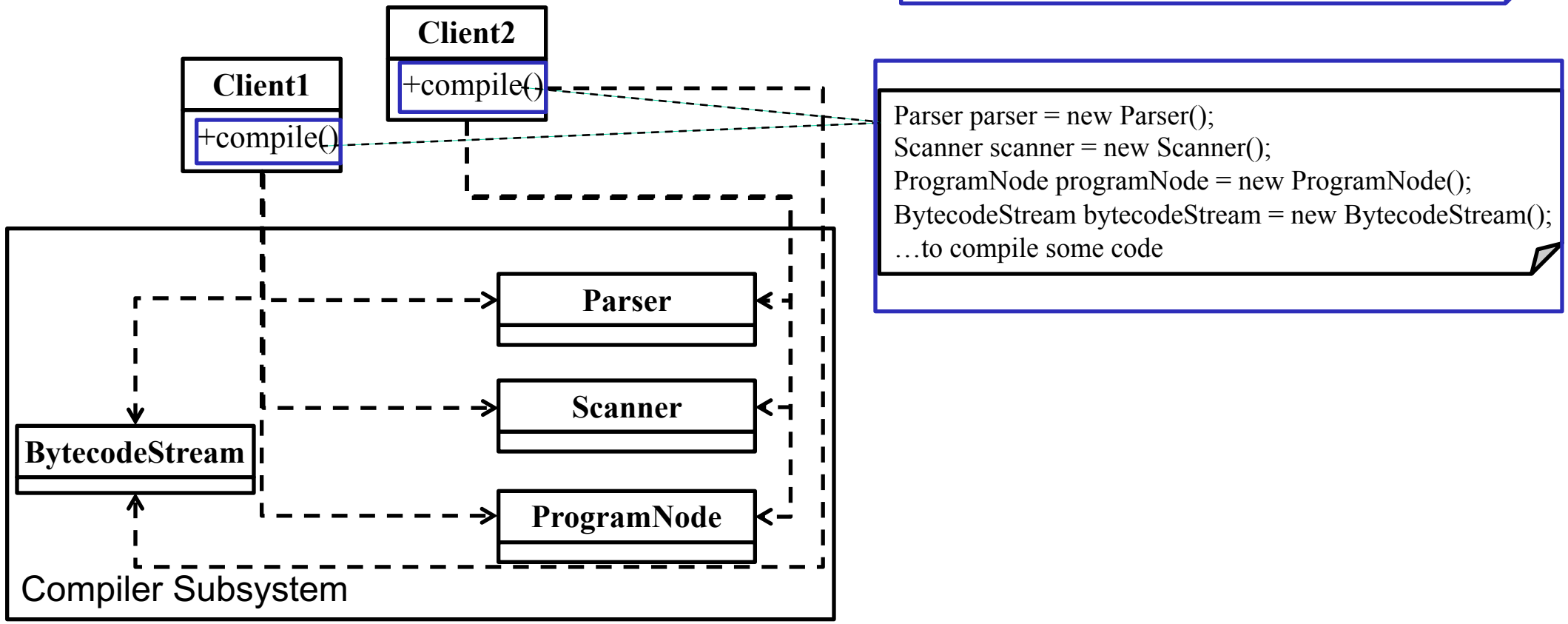
Initial Design



Problems with Initial Design

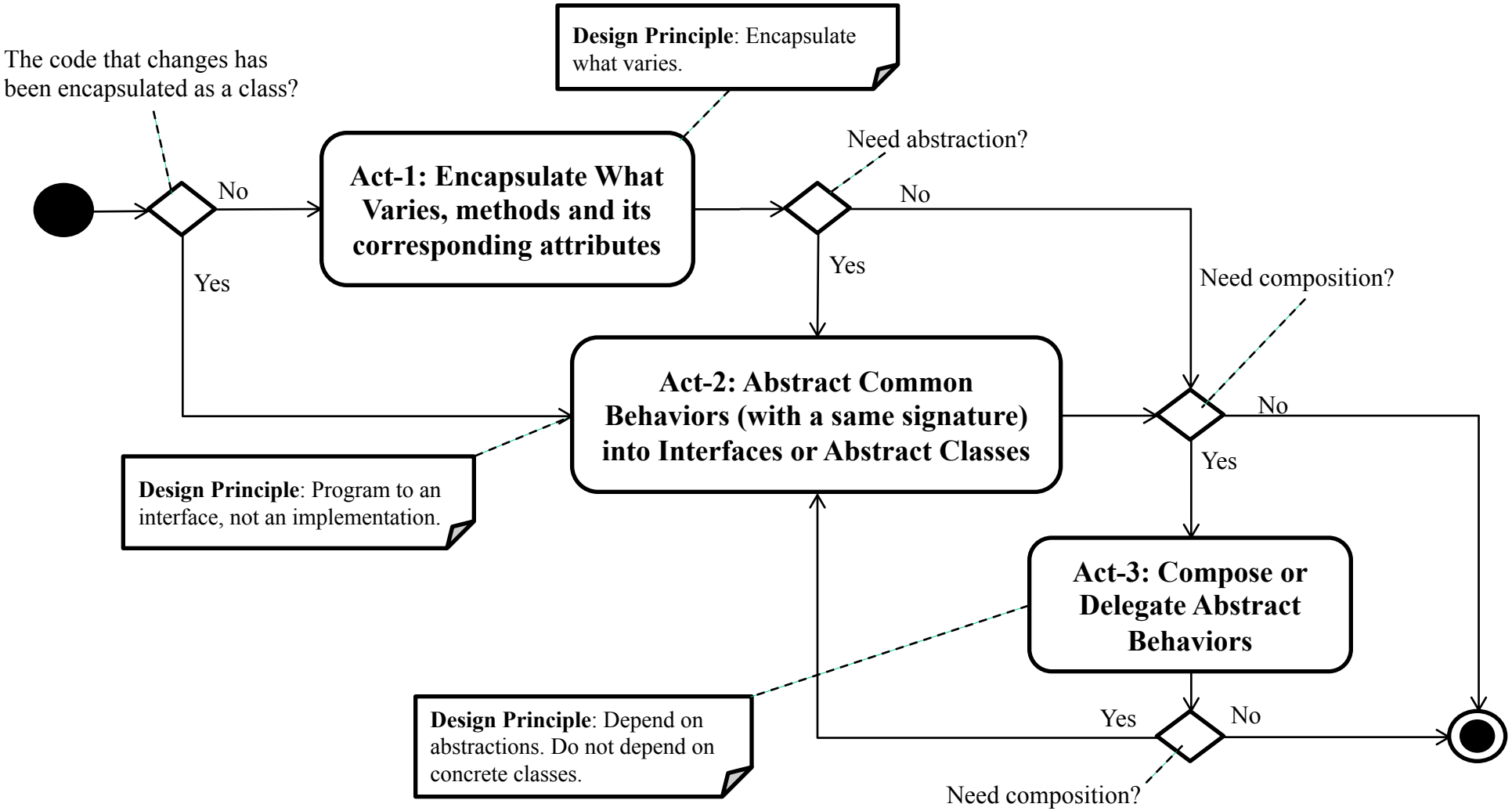
Problem1: When more and more clients want to use this compiler subsystem, it will cause a lot of duplicate code.

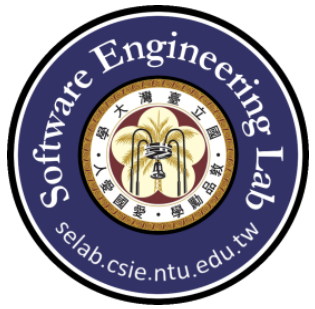
Problem2: If compiler subsystem changes, all the clients will be modified.



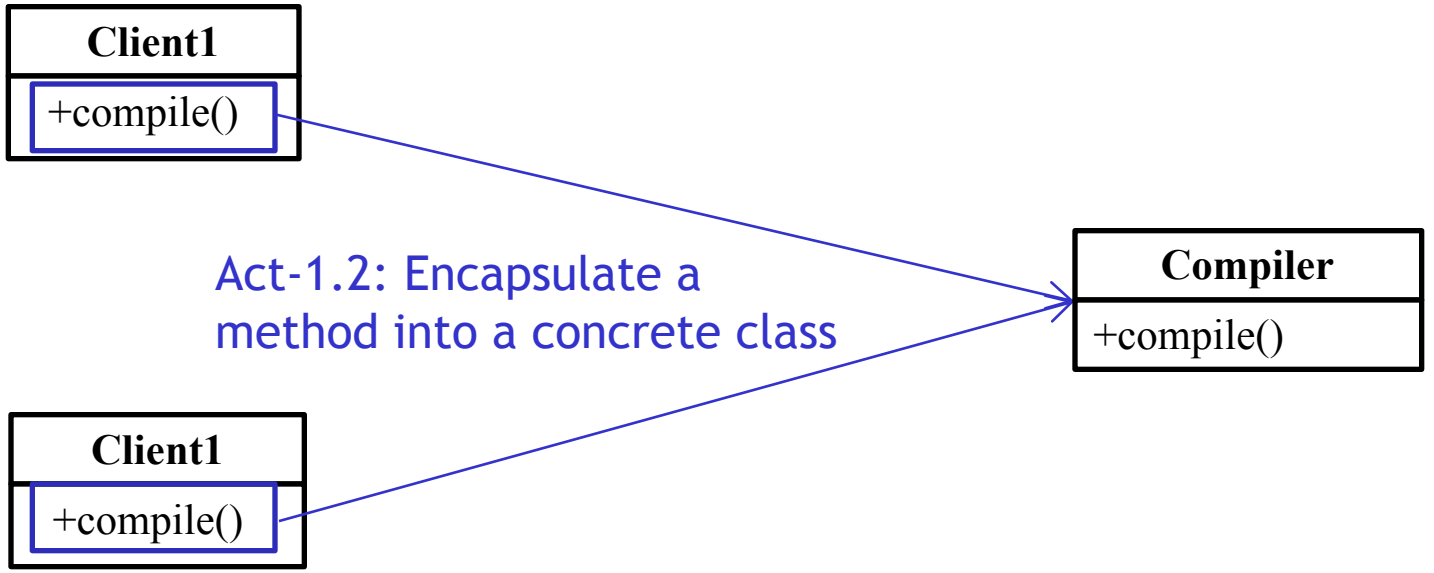

```
Parser parser = new Parser();
Scanner scanner = new Scanner();
ProgramNode programNode = new ProgramNode();
BytecodeStream bytecodeStream = new BytecodeStream();
…to compile some code
```

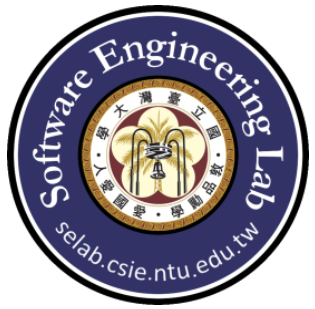
Design Process for Change





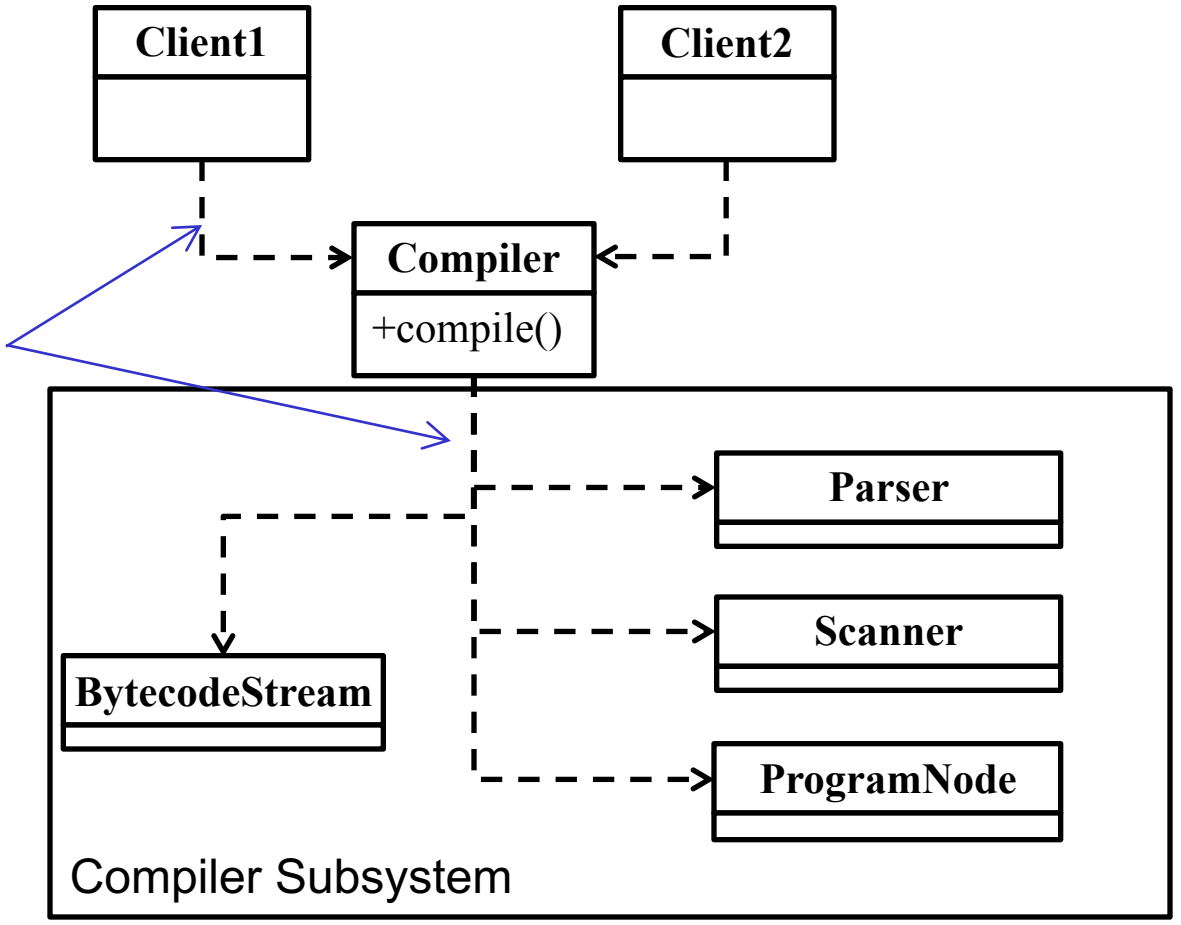
Act-1: Encapsulate What Varies

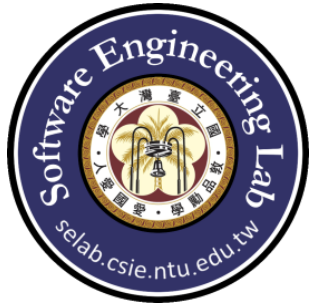




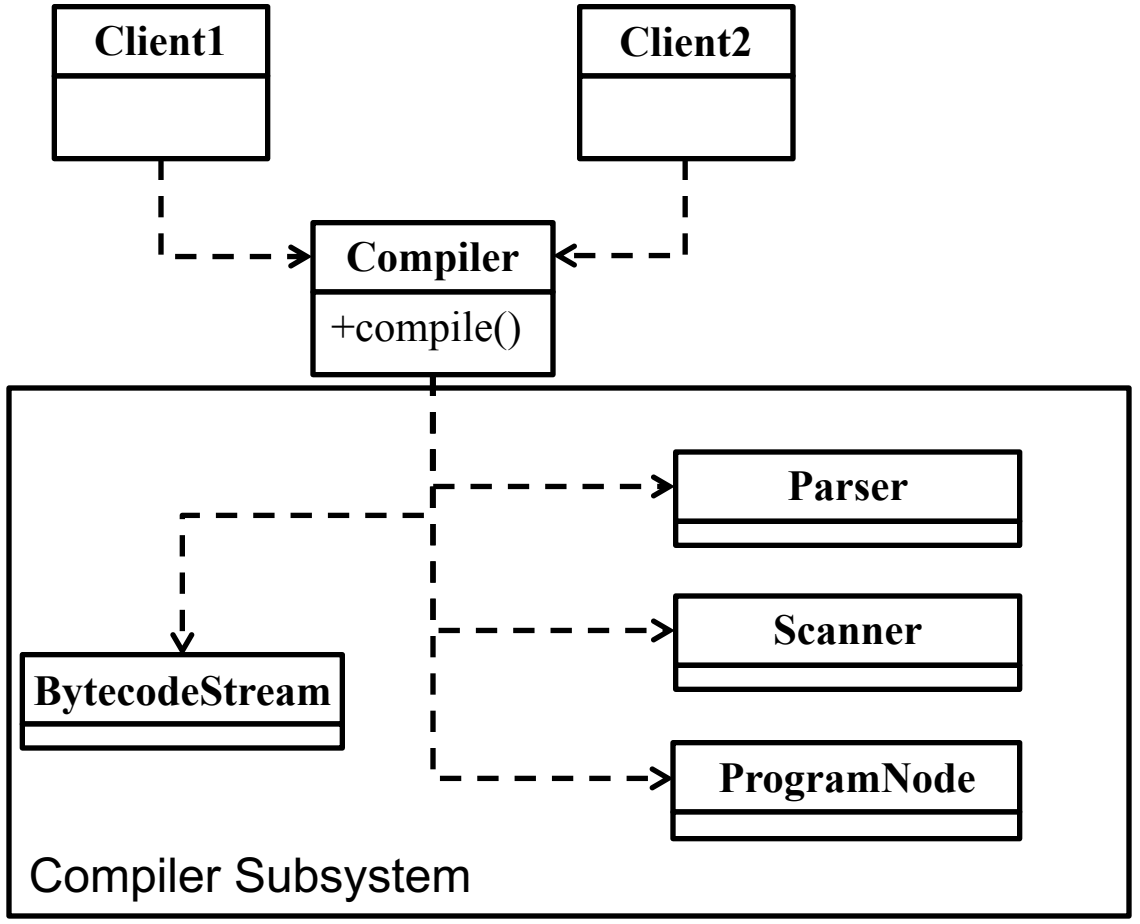
Act-3: Compose Abstract Behaviors

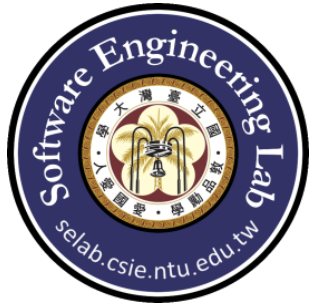
Act-3.4: Delegate behavior to a method of a concrete class





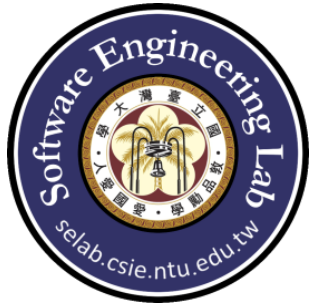
Refactored Design after Design Process





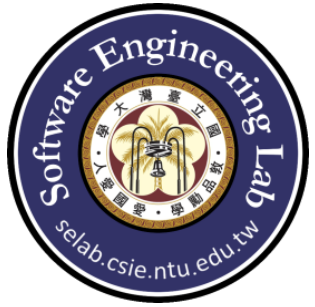
Recurrent Problem

- A common design goal is to minimize the communication and dependencies between subsystems.
  - One way to achieve this goal is to introduce a façade object that provides a single, simplified interface to the more general facilities of a subsystem.



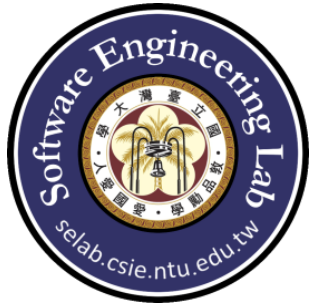
Compiler

```
public class Compiler {
    public void compile(){
        Parser parser = new Parser();
        Scanner scanner = new Scanner();
        ProgramNode programNode = new ProgramNode();
        BytecodeStream bytecodeStream = new BytecodeStream();
        parser.parse();
        scanner.scan();
        programNode.construct();
        bytecodeStream.output();
    }
}
```



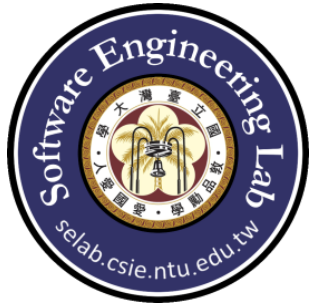
Parser

```
public class Parser {
    public void parse() { System.out.println("Parsing.....");  }
}
```



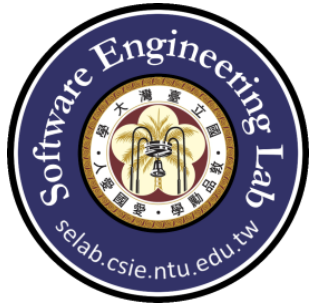
Scanner

```
public class Scanner {
    public void scan() { System.out.println("Scanning.....");}
}
```



ProgramNode

```
public class ProgramNode {
    public void construct() { System.out.println("Construct Program Node"); }
}
```



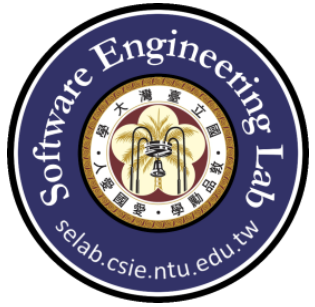
BytecodeStream

```
public class BytecodeStream {
    public void output() { System.out.println("Output bytecode....."); }
}
```

Test case output

Sample.out

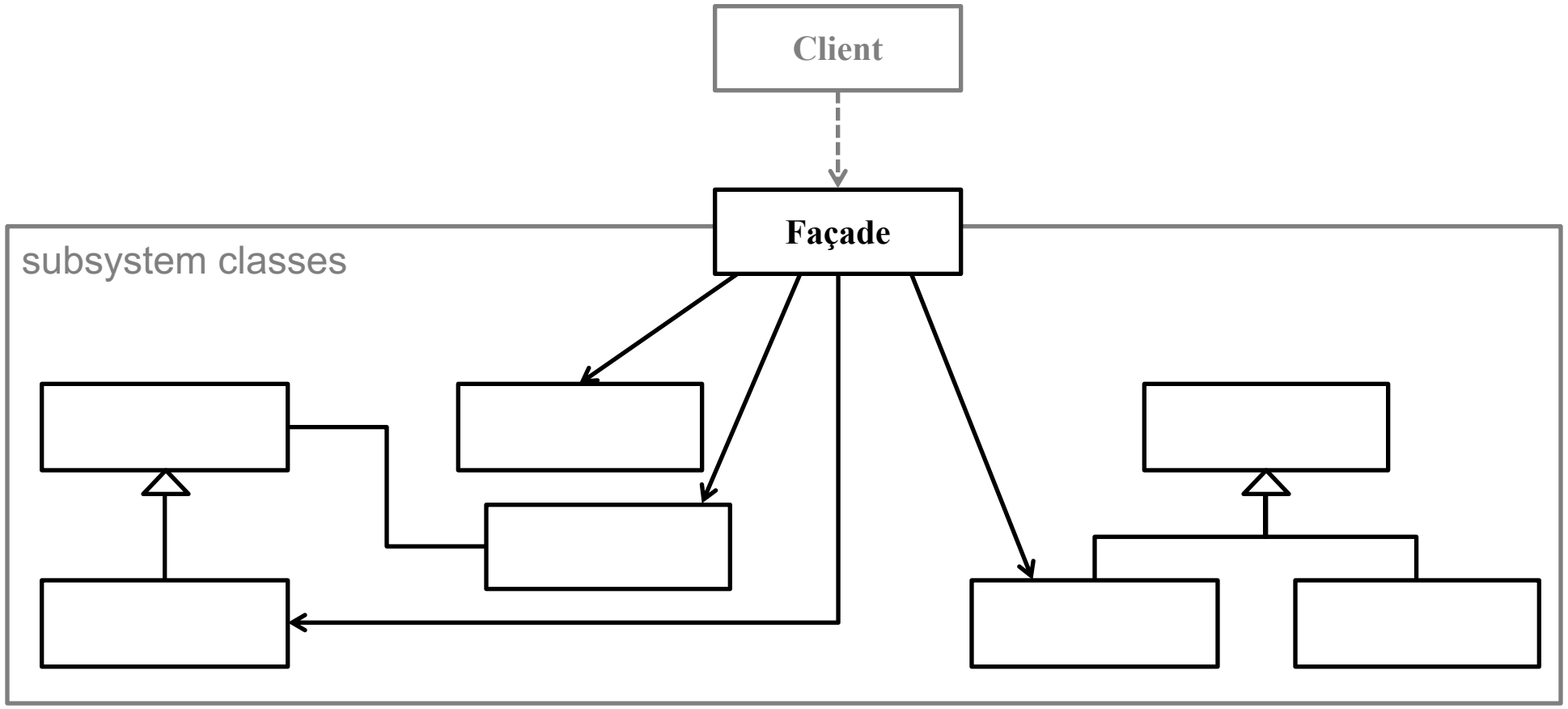
```
1  Parsing.....
2  Scanning.....
3  Construct Program Node
4  Output bytecode.....
```

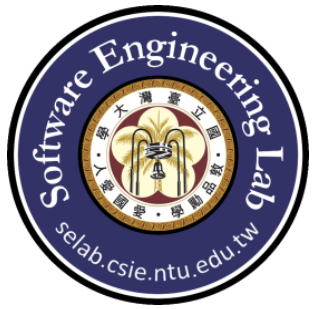


Intent

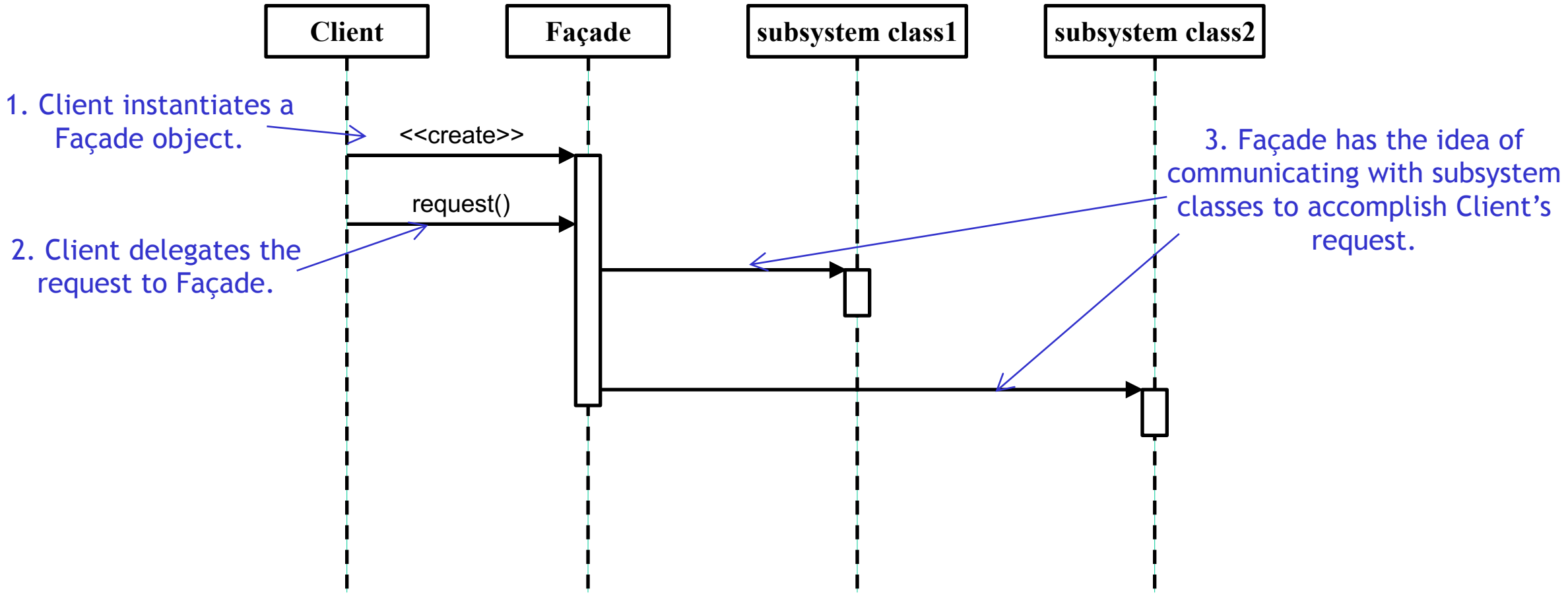
- Provide a unified interface to a set of interfaces in a subsystem. Façade defines a higher-level interface that makes the subsystem easier to use.

Façade Structure$_1$



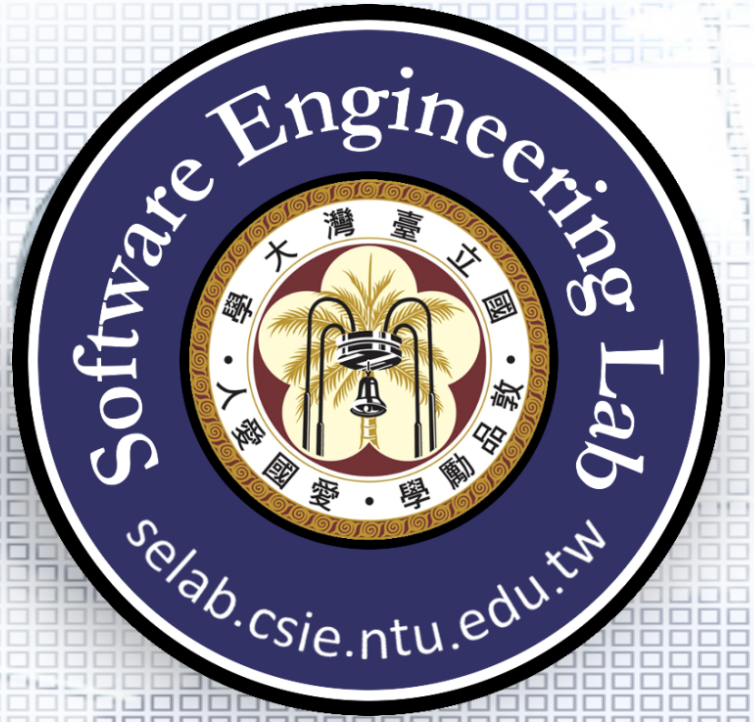


Façade Structure$_2$



Façade Structure$_3$

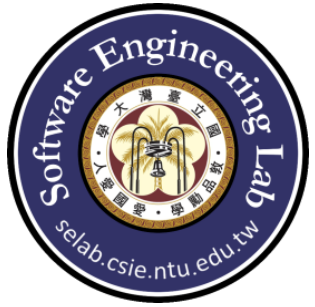
	Instantiation	Use	Termination
Façade	Client	Client	Don't Care
subsystem classes	Don't Care	Façade	Don't Care



Home Sweet Home Theater

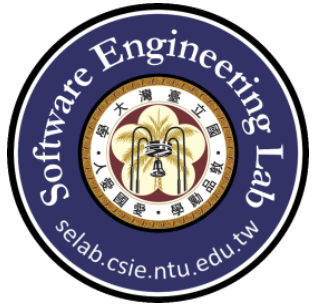
Prof. Jonathan Lee (李允中)

Department of Computer Science and Information Engineering
National Taiwan University



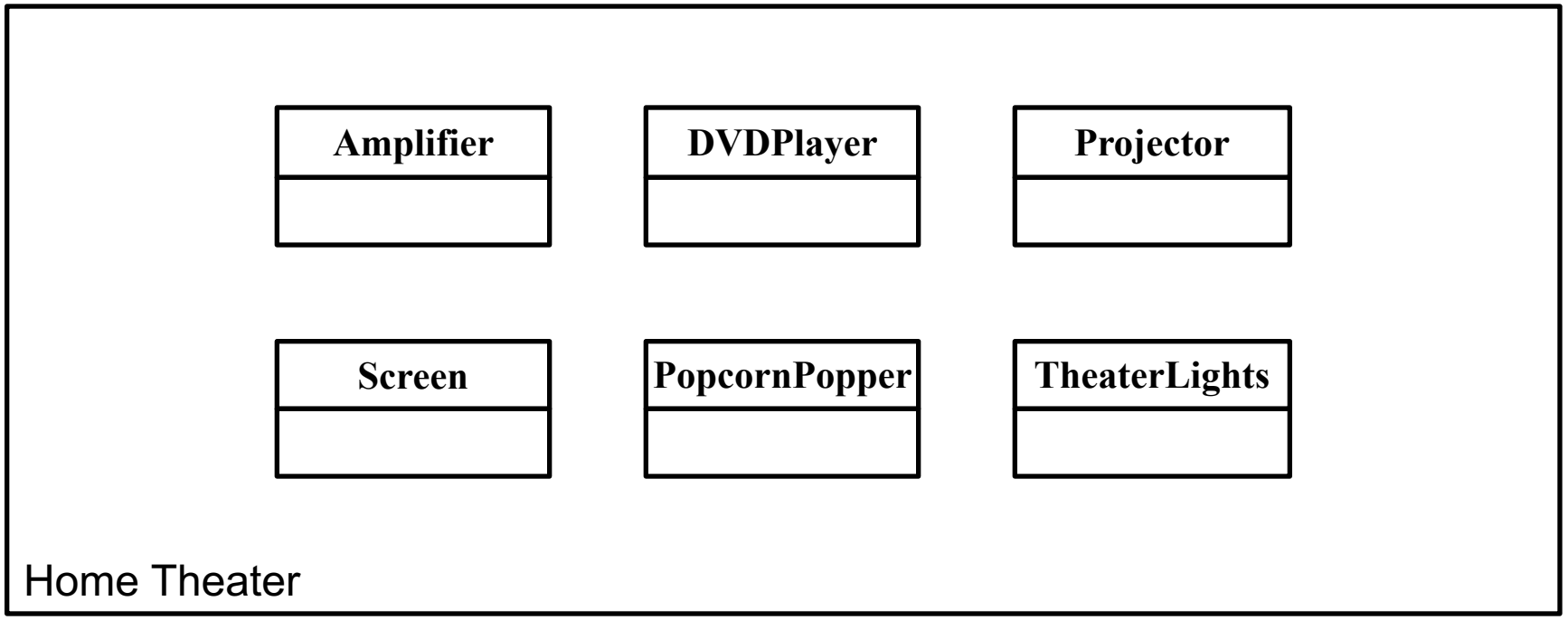
Requirements Statements

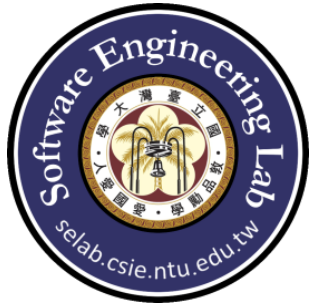
- A Home Theater consists of an amplifier, a DVD player, a projector, a screen, a popcorn popper, and theater lights.
- A user can watch a movie through the following process:
  1. Turn on the popcorn popper
  2. Start the popper popping
  3. Dim the lights
  4. Put the screen down
  5. Turn the projector on
  6. Turn the sound amplifier on
  7. Turn the DVD player on
  8. Start the DVD player playing



Requirements Statements$_1$

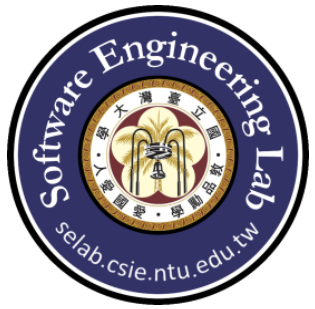
- A Home Theater consists of an amplifier, a DVD player, a projector, a screen, a popcorn popper, and theater lights.



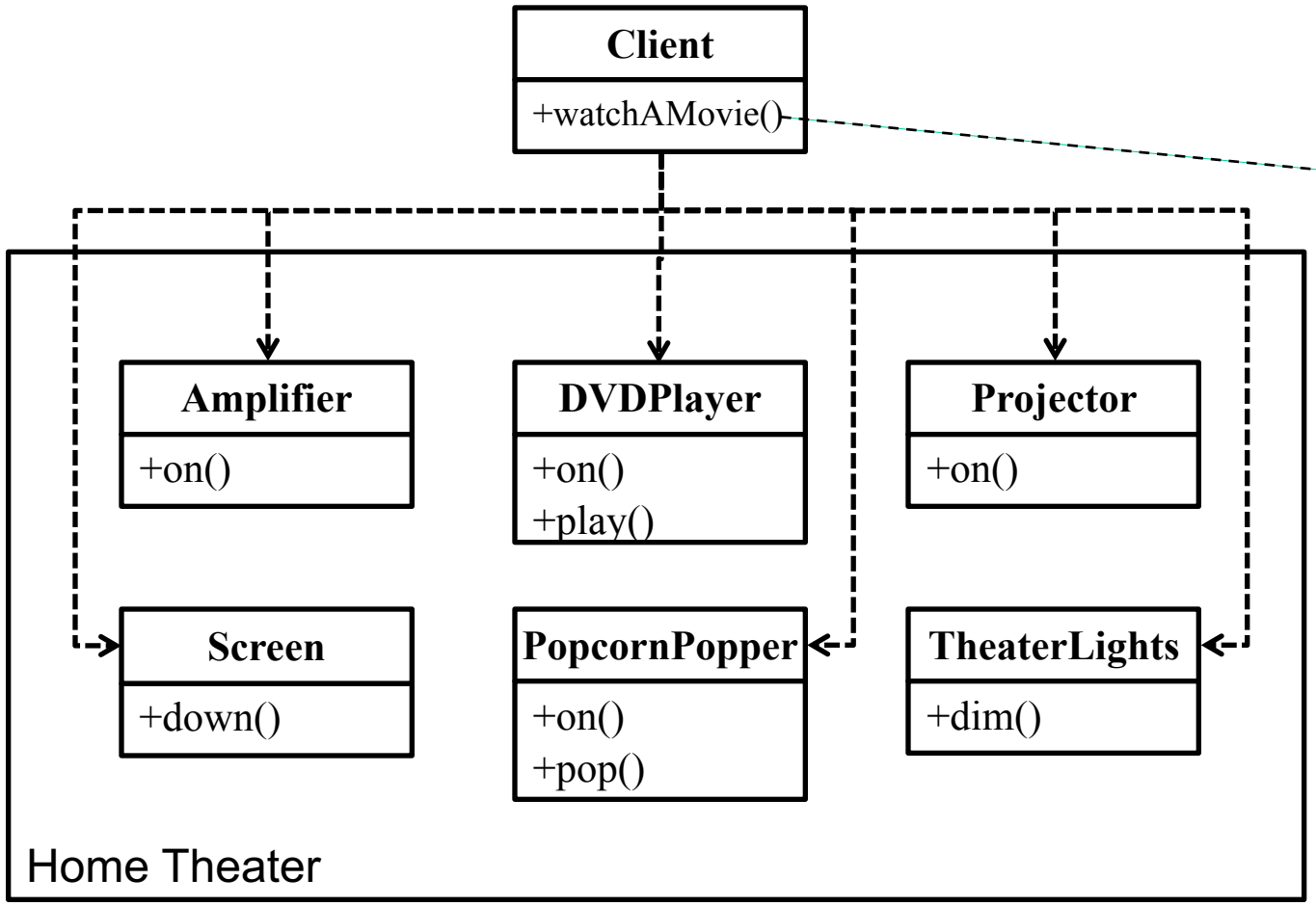


Requirements Statements$_2$

- A user can watch a movie through the following process:
  1. Turn on the popcorn popper
  2. Start the popper popping
  3. Dim the lights
  4. Put the screen down
  5. Turn the projector on
  6. Turn the sound amplifier on
  7. Turn the DVD player on
  8. Start the DVD player playing

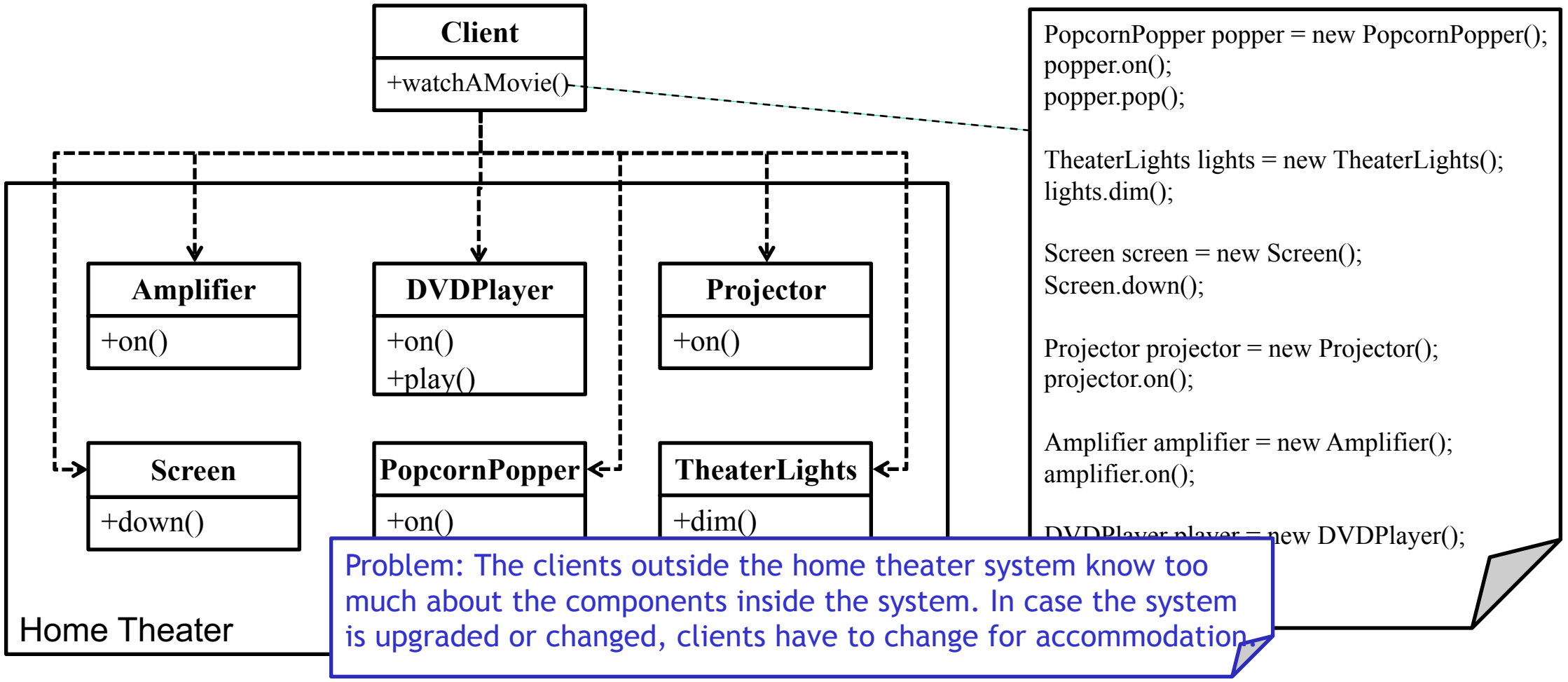


Initial Design

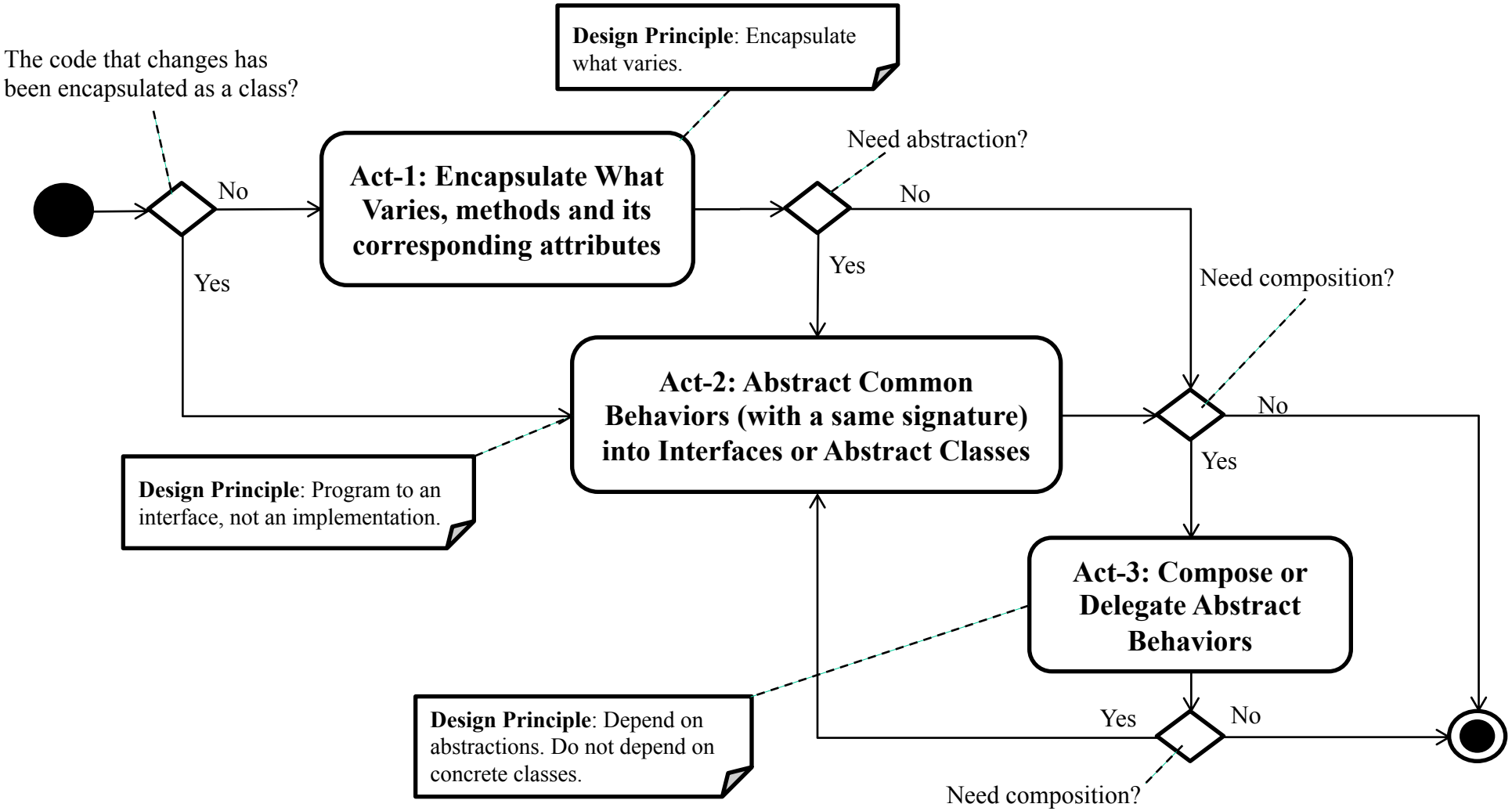


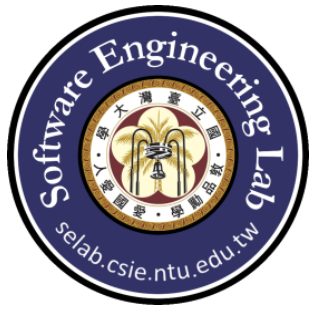

```
PopcornPopper popper = new PopcornPopper();
popper.on();
popper.pop();

TheaterLights lights = new TheaterLights();
lights.dim();

Screen screen = new Screen();
Screen.down();

Projector projector = new Projector();
projector.on();

Amplifier amplifier = new Amplifier();
amplifier.on();

DVDPlayer player = new DVDPlayer();
player.on();
player.play();
```

Problems with Initial Design



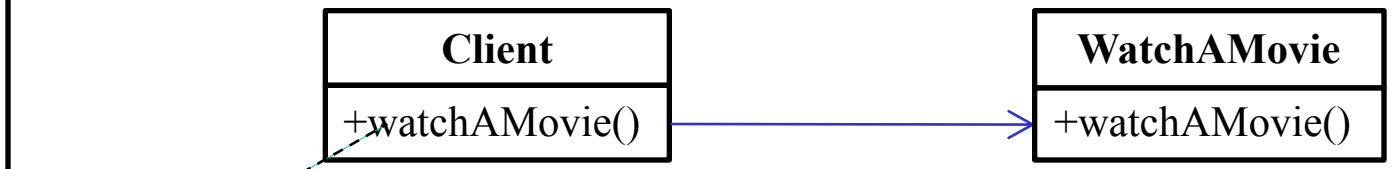
Design Process for Change





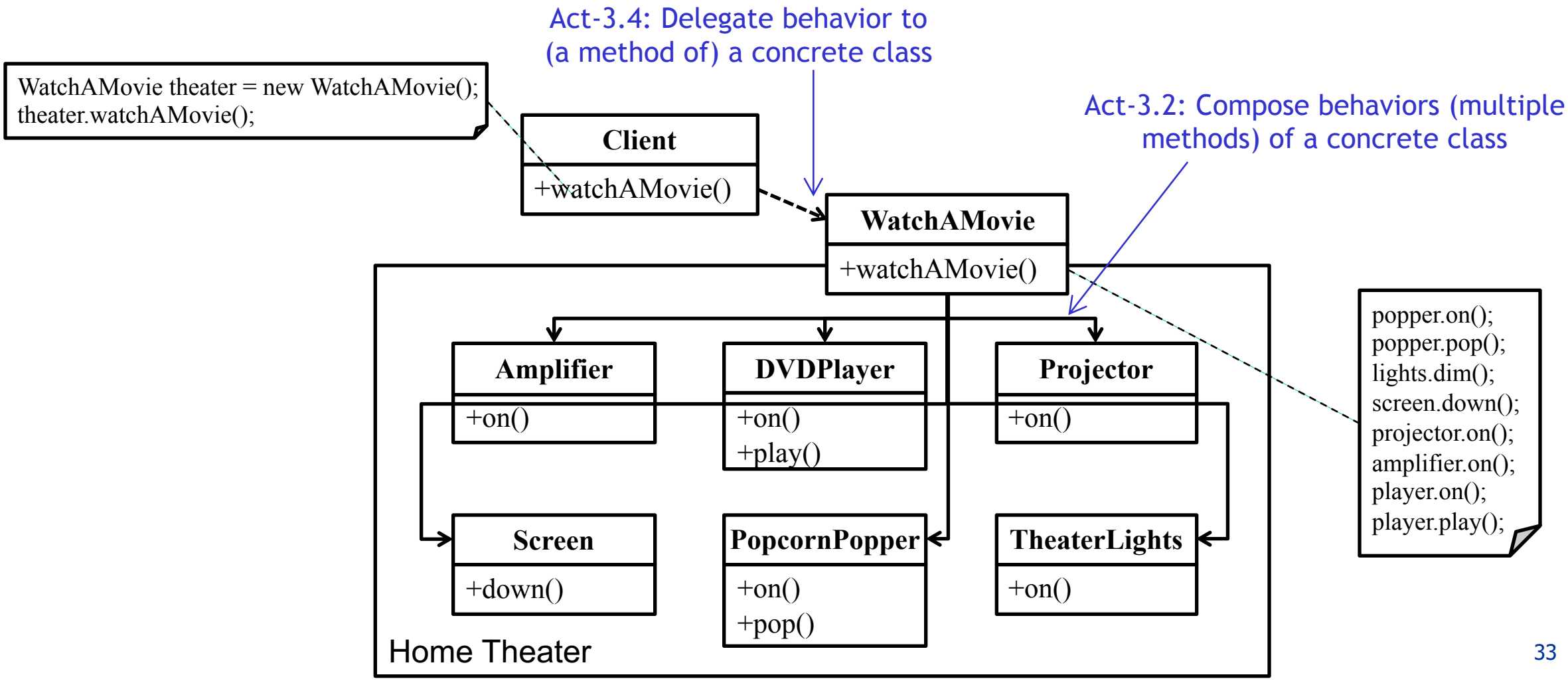
Act-1: Encapsulate What Varies

```
PopcornPopper popper = new PopcornPopper();
popper.on();
popper.pop();

TheaterLights lights = new TheaterLights();
lights.dim();

Screen screen = new Screen();
Screen.down();

Projector projector = new Projector();
projector.on();

Amplifier amplifier = new Amplifier();
amplifier.on();

DVDPlayer player = new DVDPlayer();
player.on();
player.play();
```



Act-1.2: Encapsulate a method into a concrete class

Act-3: Compose Behaviors



Refactored Design after Design Process