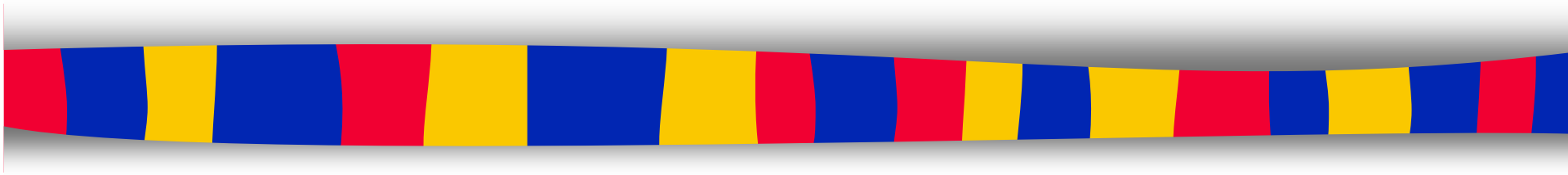
CISC 322
Software Architecture



Lecture 16:
Design Patterns 3

Emad Shihab

Material drawn from [Gamma95, Coplien95]

Slides adapted from Spiros Mancoridis and Ahmed E. Hassan

Template Pattern Intent

- Define the skeleton of an algorithm in an operation, deferring some steps to subclasses.
- The *Template Method* lets subclasses redefine certain steps of an algorithm without changing the algorithm's structure.

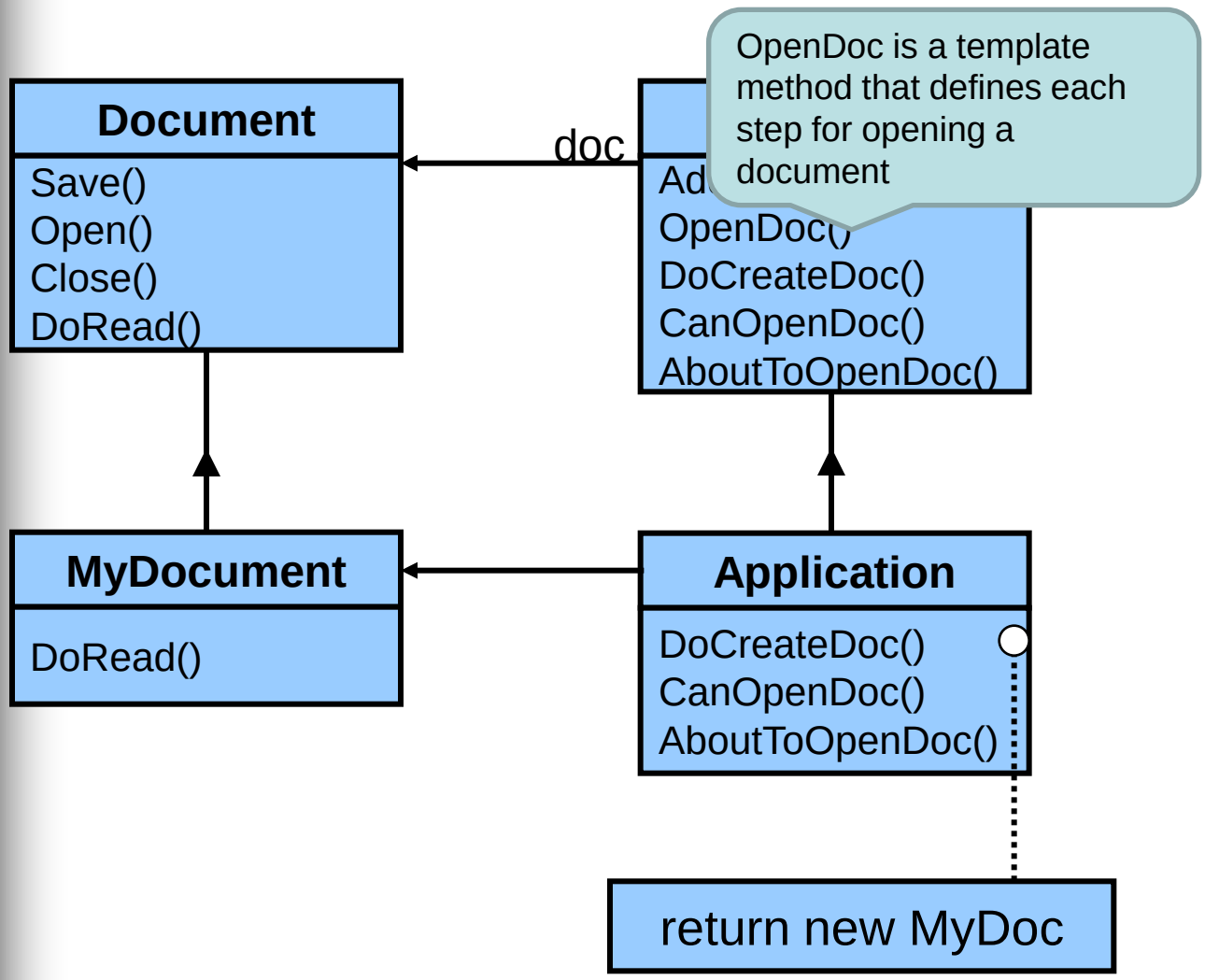
Template Pattern Motivation

- Consider an application that provides Application and Document classes
  - Application: opens existing document
  - Document: represents the information in a doc
- By defining some of the steps of an algorithm, using abstract operations, the template method fixes their ordering.

Template Pattern Motivation

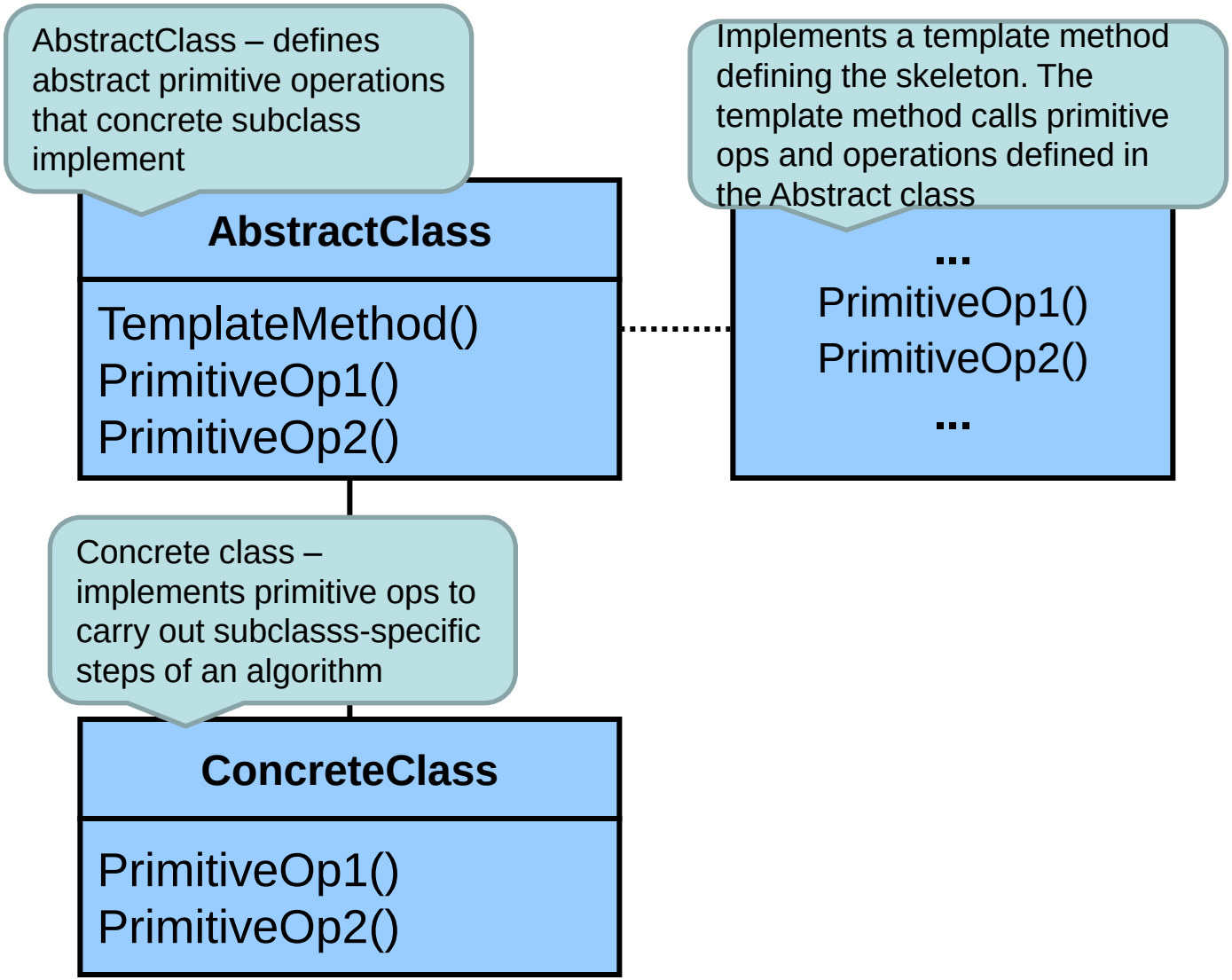
- Specific applications can subclass Application and Document to suit their specific needs
  - Drawing application: defines DrawApplication and DrawDocument sublclasees
  - Spreadsheet application: defines SpreadsheetApplication and SpreadsheetDocument sublclasees

Template Pattern Example



- CanOpenDoc() – check if doc can be opened
- DoCreateDoc() – create doc
- AboutToOpenDoc() – lets application know when a doc is about to be opened

Template Pattern Structure



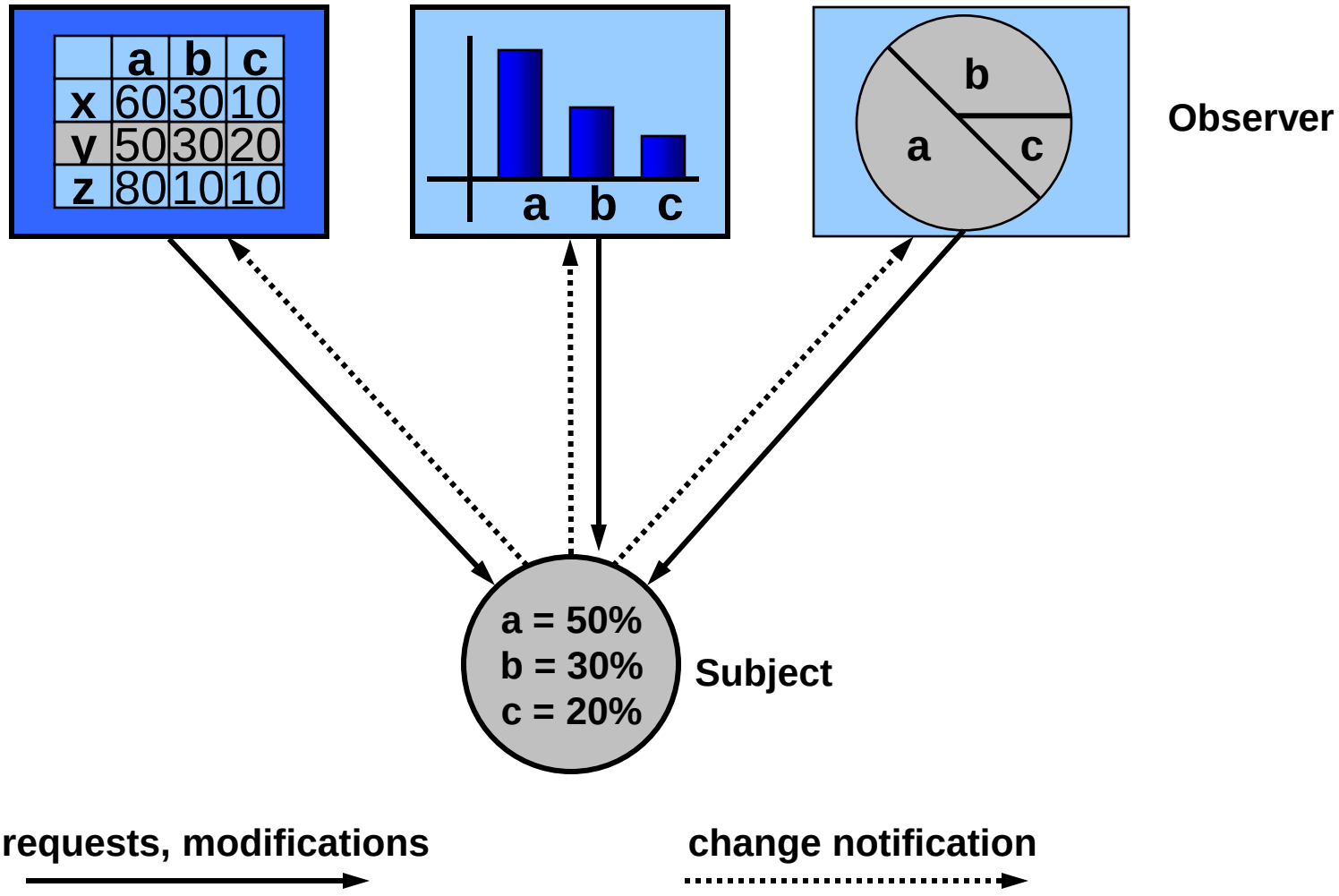
Observer Pattern Motivation

- A common side-effect of partitioning a system into a collection of cooperating classes is the need to maintain consistency between related objects.
- How can you achieve consistency?

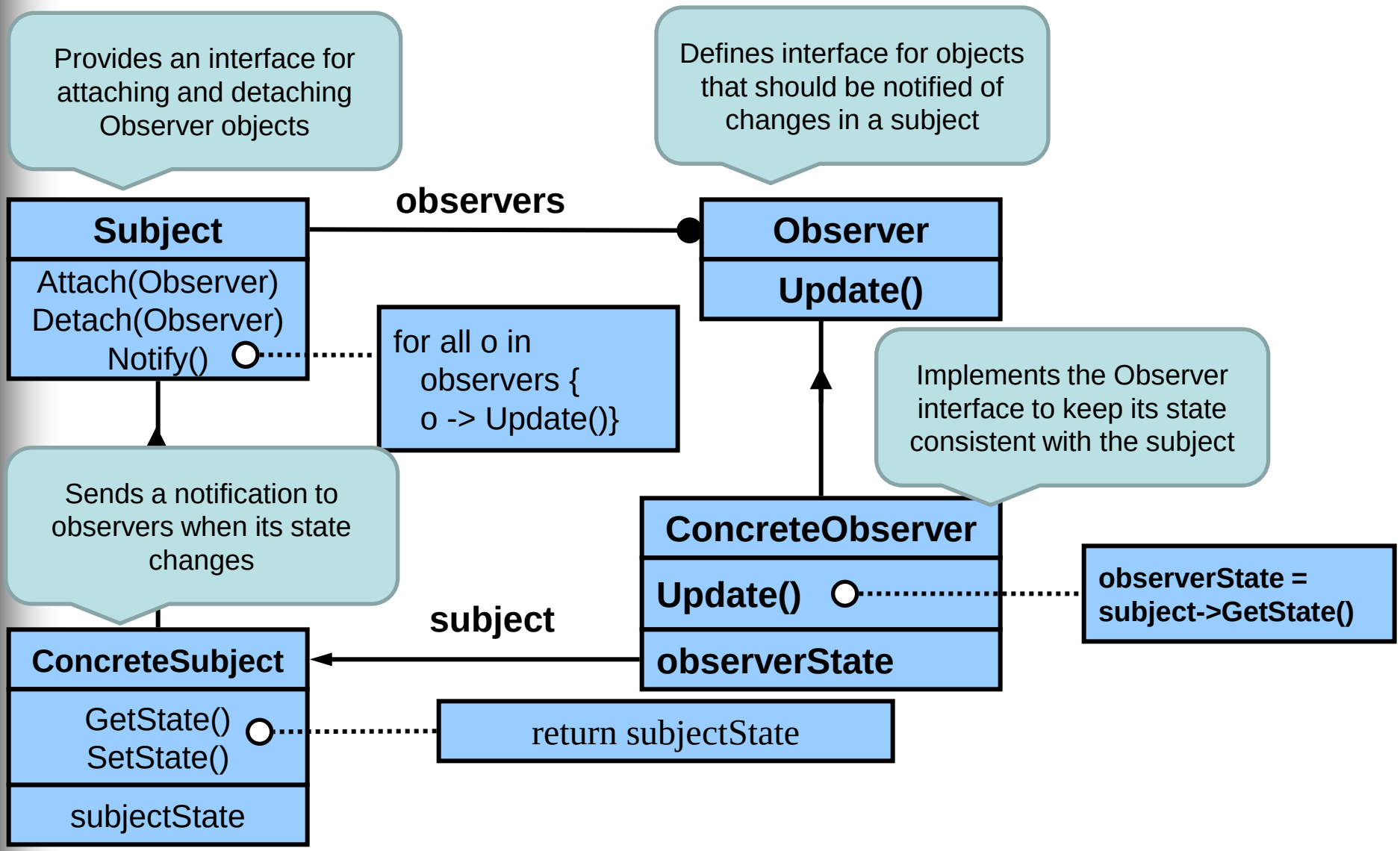
Observer Intent

- Define a one-to-many dependency between objects so that when **one object changes state**, **all its dependents** are **notified** and **updated automatically**.

Observer Pattern Example



Observer Pattern Structure



Master-Slave Pattern Motivation

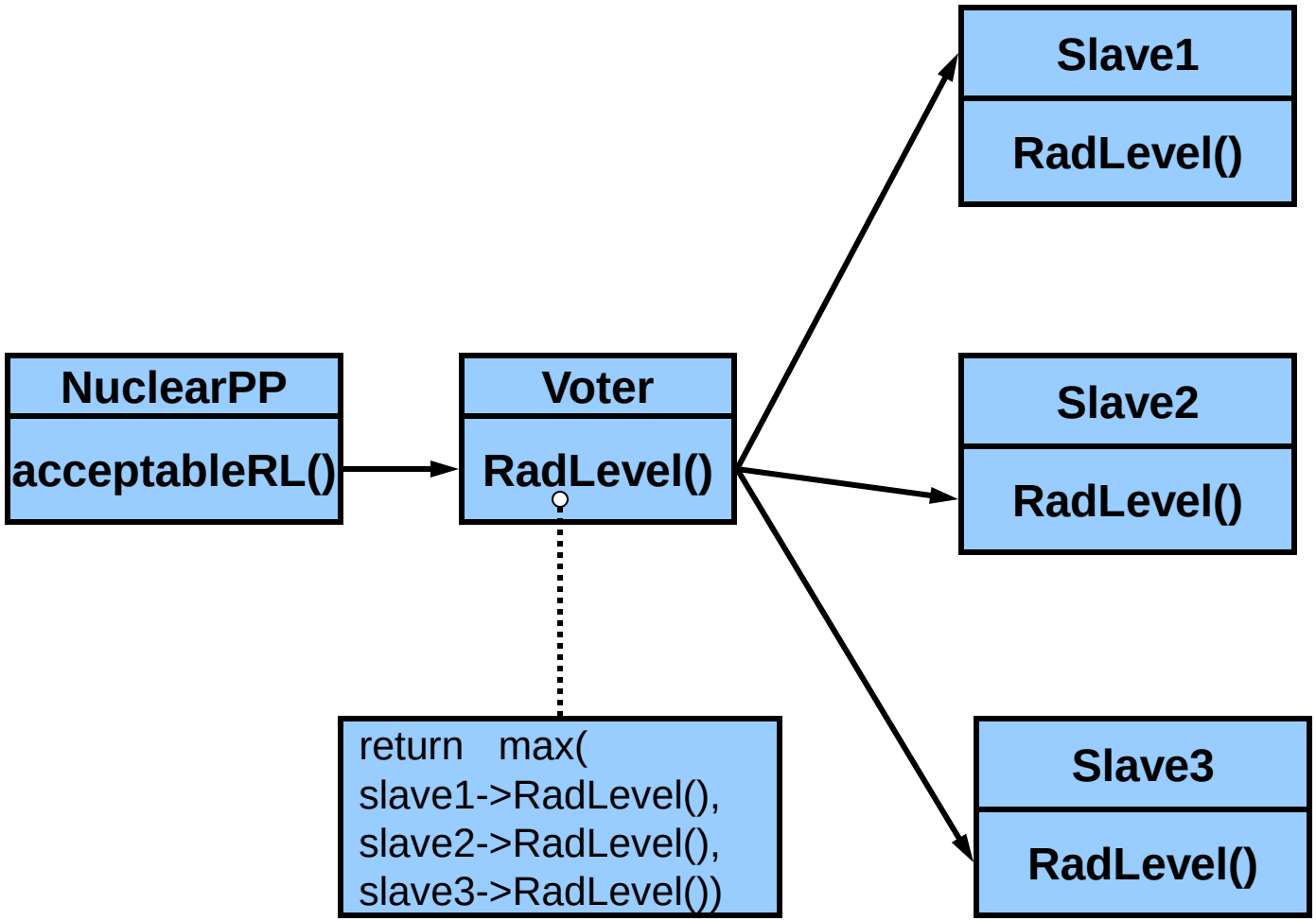
- Fault tolerance is a critical factor in many systems.
- Replication of services and delegation of the same task to several independent suppliers is a common strategy to handle such cases.

Master-Slave Pattern Intent

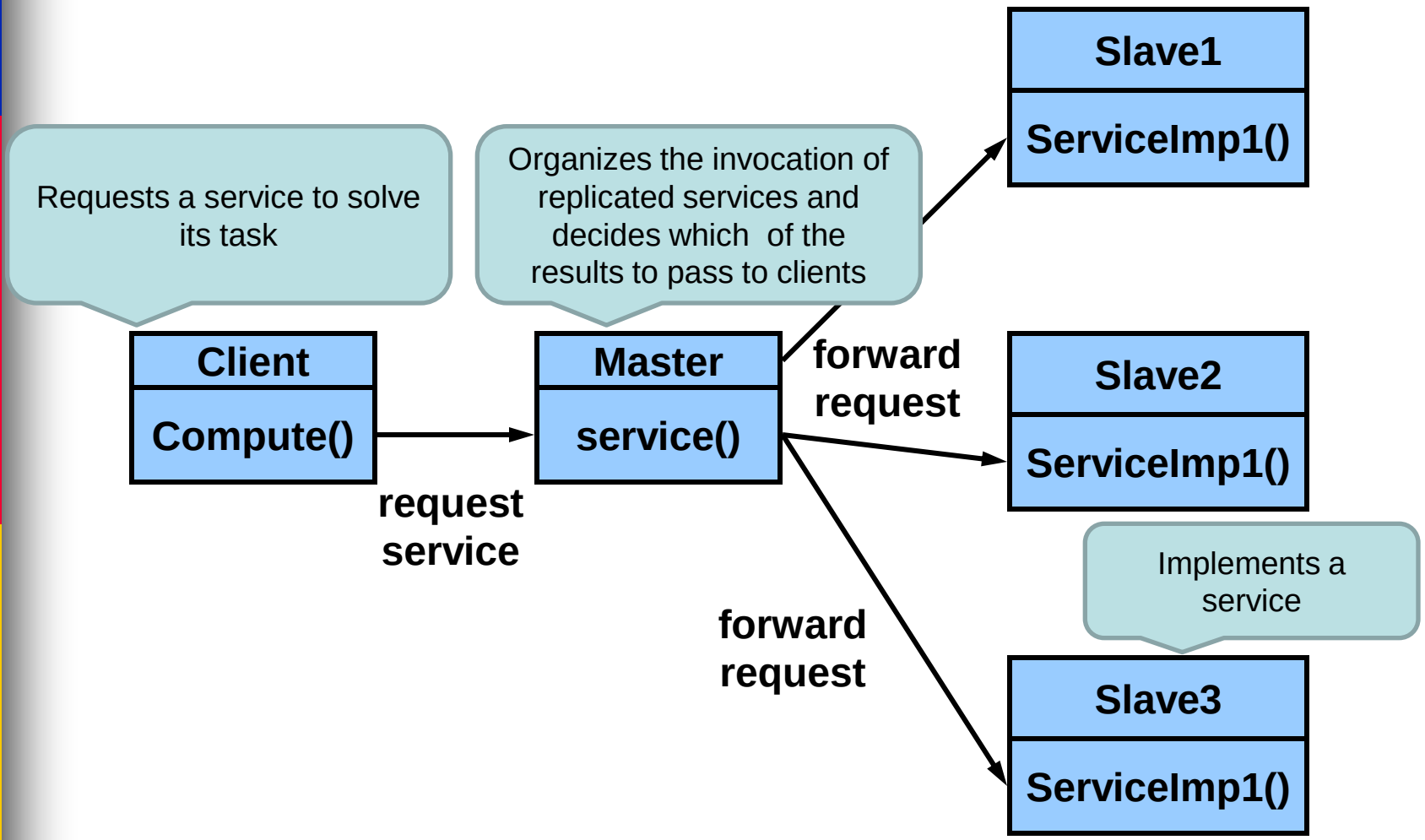
- Independent components providing the same service (slaves) are separated from a component (master) responsible for invoking them and for selecting a particular result from the results returned by the slaves.

- (Master) Handles the computation of replicated services within a software system to achieve fault tolerance and robustness.

Master-Slave Pattern Example



Master-Slave Pattern Structure



Singleton Intent

- Ensure a class only has one instance, and provide a global point of access to it

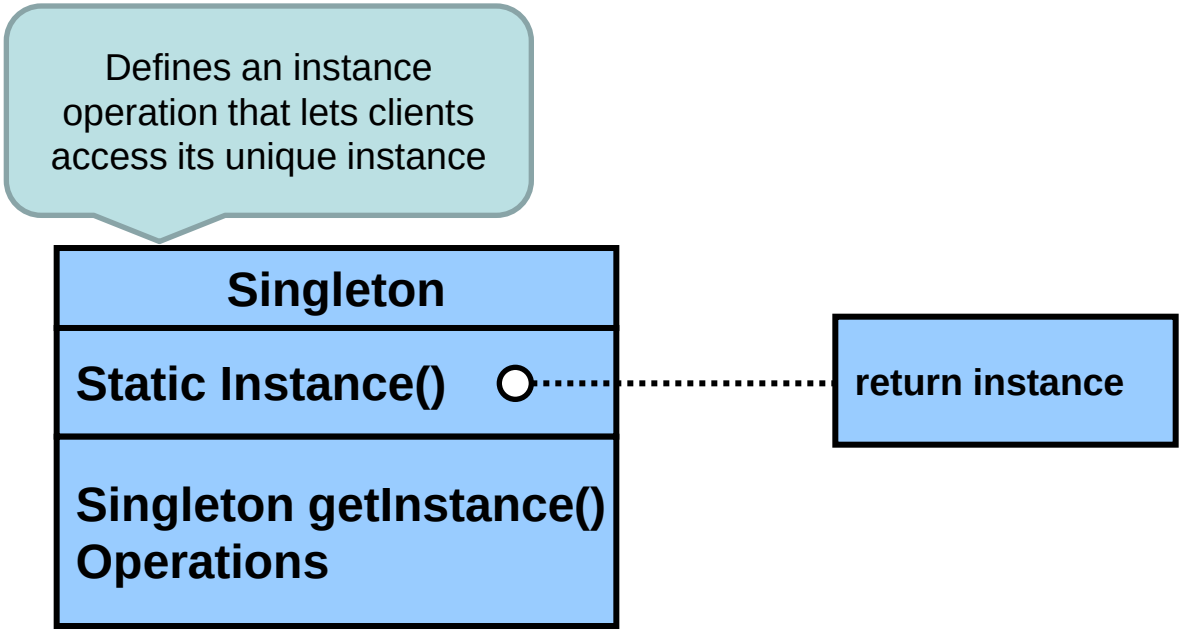
Singelton Pattern Motivation

- It is important that some classes have only one instance
  - E.g., one printer spooler, one window manager
- How to ensure that a class only has one instance?

Singelton Pattern Motivation

- Make the class itself responsible for keeping track of its sole instance
- The class can ensure that no other instance can be created and provides a way to access the instance

Singleton Pattern Structure



Singleton example

```
public class SimpleSingleton {
   private SimpleSingleton singleInstance = null;

   //Marking default constructor private
   //to avoid direct instantiation.
   private SimpleSingleton() {
   }

   //Get instance for class SimpleSingleton
   public static SimpleSingleton getInstance() {

      if(null == singleInstance) {
         singleInstance = new SimpleSingleton();
      }

      return singleInstance;
   }
}
```

http://viralpatel.net/blogs/2009/01/java-singleton-design-pattern-tutorial-example-singleton-j2ee-design-pattern.html

Schedule

NOVEMBER 2011						
SUN	MON	TUES	WED	THURS	FRI	SAT
		1 Group meeting	2 Group meeting	3	4 Group meeting	5
6	7 Group meeting	8 Presentations	9 Presentations	10	11 Presentations	12
13	14 Reports Due	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30			