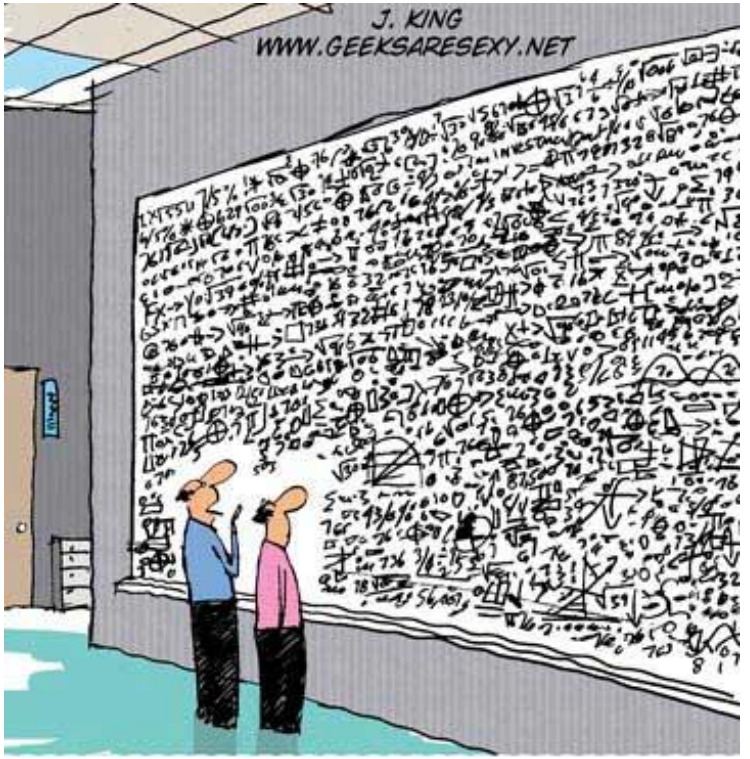
Separation of Concerns

Eduardo Figueiredo

http://www.dcc.ufmg.br/~figueiredo

Motivation

- We develop software with several quality attributes in mind
  - Reusable
  - Flexible
  - Reliable, etc.
- Software complexity always grows
  - New challenges for software development



"...And that, in simple terms, is what's wrong with your software design."

Separation of Concerns

In order to master complexity, one has to deal with one important issue (or concern) at a time.

Dijkstra, 1976

- Development techniques have been proposed in the last 40 years to support better separation of concerns

Structured Programming

- Functions and Procedures
  - Abstractions to modularize actions
- Data
  - Input or output of actions
- Actions and data are designed and implemented apart
  - High level programming

OO Programming

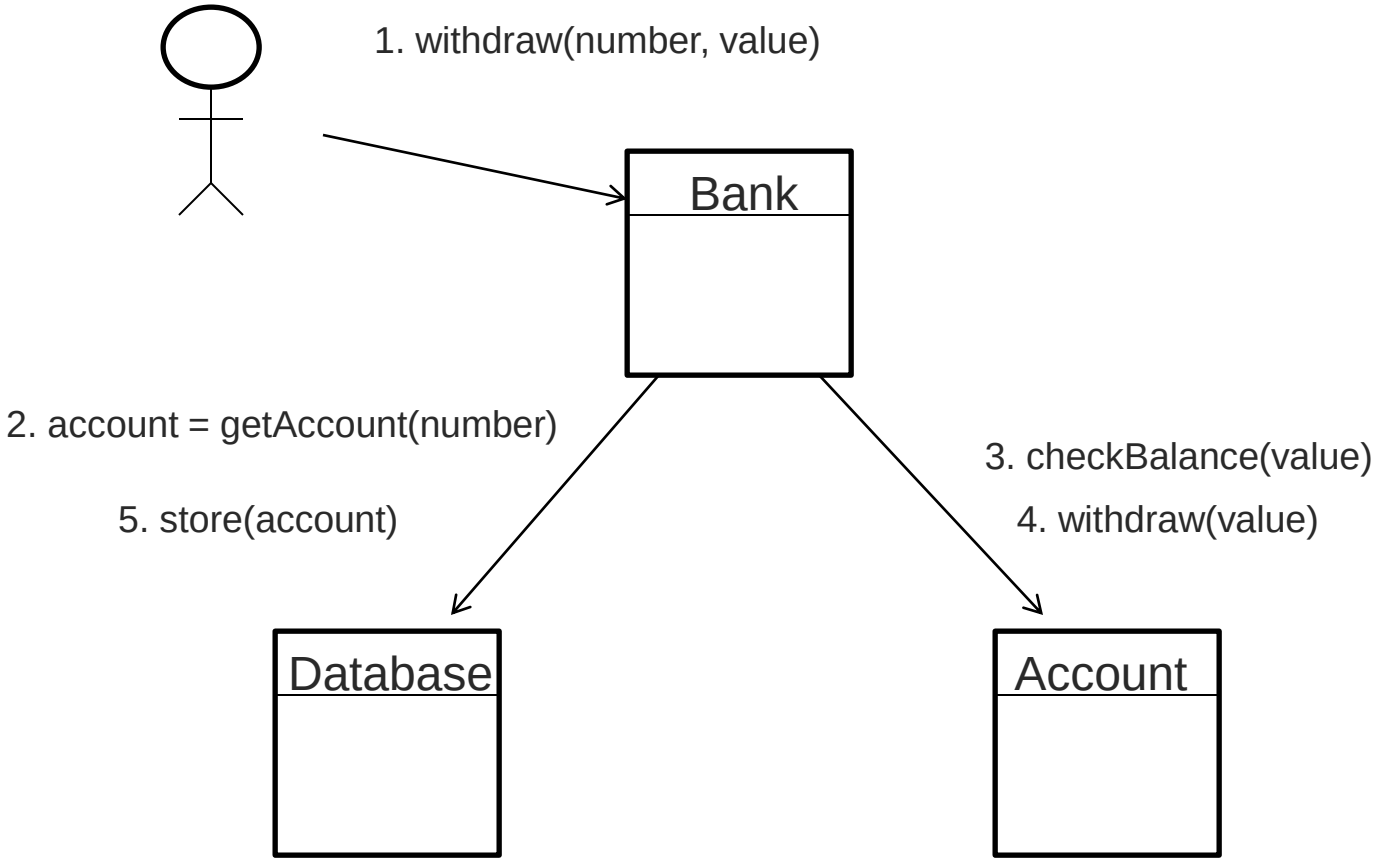
- Higher level abstractions (closer to human reasoning)
  - Objects, not functions
- Objects put together data and actions in a single entity
- An OO program is composed of several objects talking to each other

Examples of Objetos

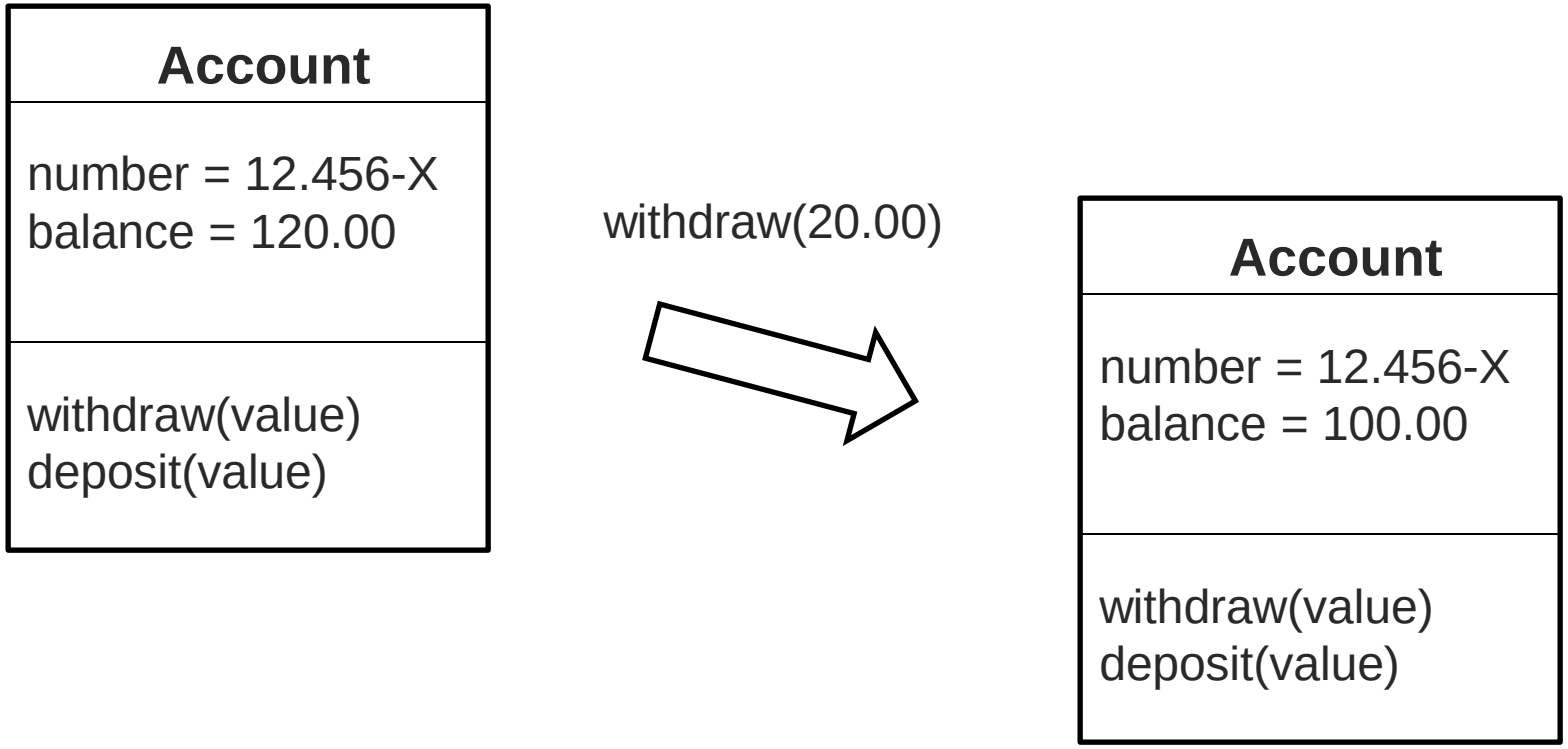
- Bank
  - Financial entity

- Account
  - An ordinary client account in the bank
- Database (of accounts)
  - Archive to store accounts

Dynamics of Withdraw



Example: Account Object



States of the object Account

Class Account in Java

```
public class Account {

  private String number;
  private double balance;

  public void withdraw(double value) {
    if (getBalance() >= value)
      setBalance(getBalance() - value);
    else { /*erro!*/ }
  }

  public void credit(double value) {
    setBalance(getBalance() + value);
  }

  ...
}
```

Issues with OOP

- Software complexity needs to be addressed (beyond OOP)
- Quality attributes are not properly supported by OOP
- Separation of concerns
  - OOP supports separation of functional requirements
  - OOP does not care about separation of non-functional requirements

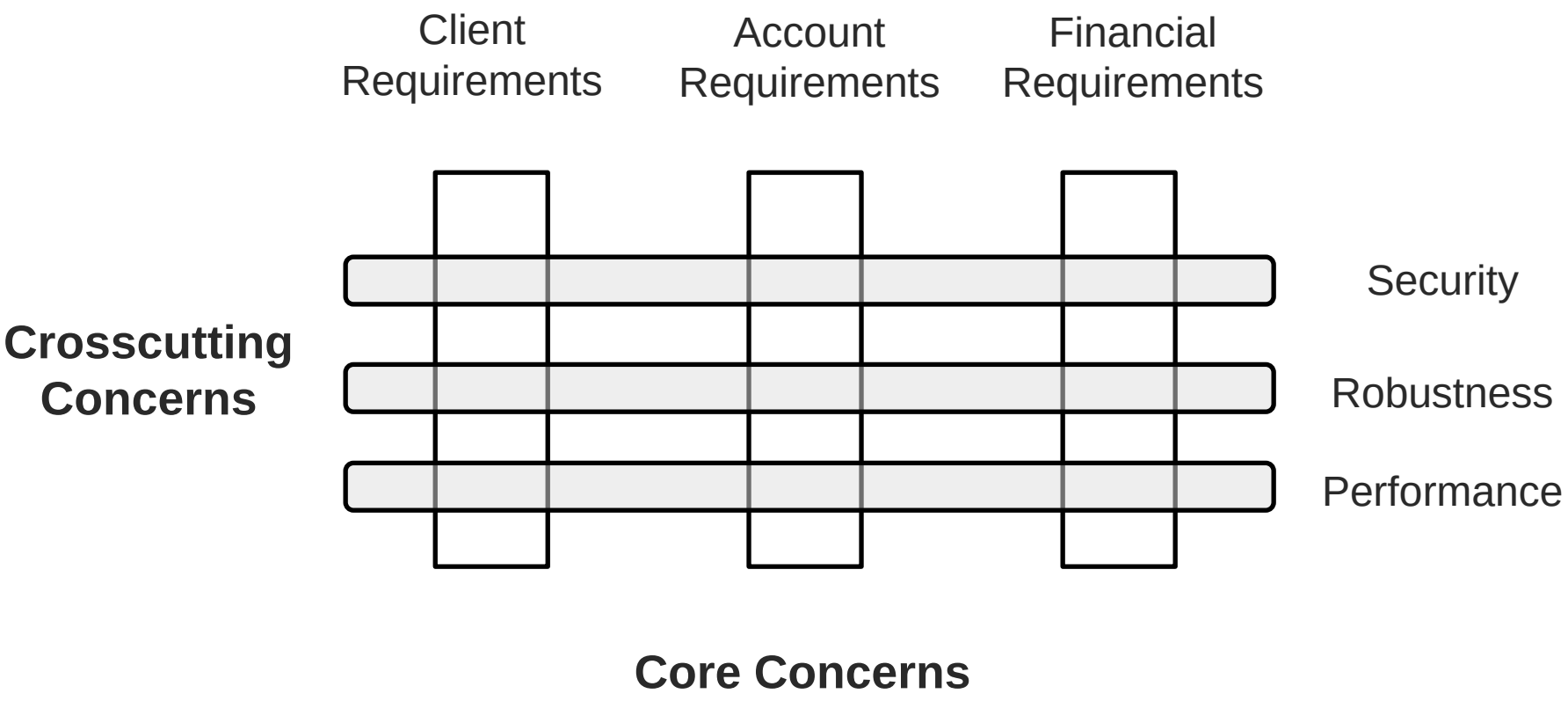
Classification of Concerns

- Core concerns
  - Functional concerns
- Crosscutting concerns
  - Quality of service
  - Organizational concerns
  - Global system concerns

Crosscutting Concerns

- Concerns implemented by several modules of the system and mixed up with other concerns
- Two typical problems
  - Scattering
  - Tangling
- It is hard to change, remove or evolve a crosscutting concern

Representation



Example of Logging in Java

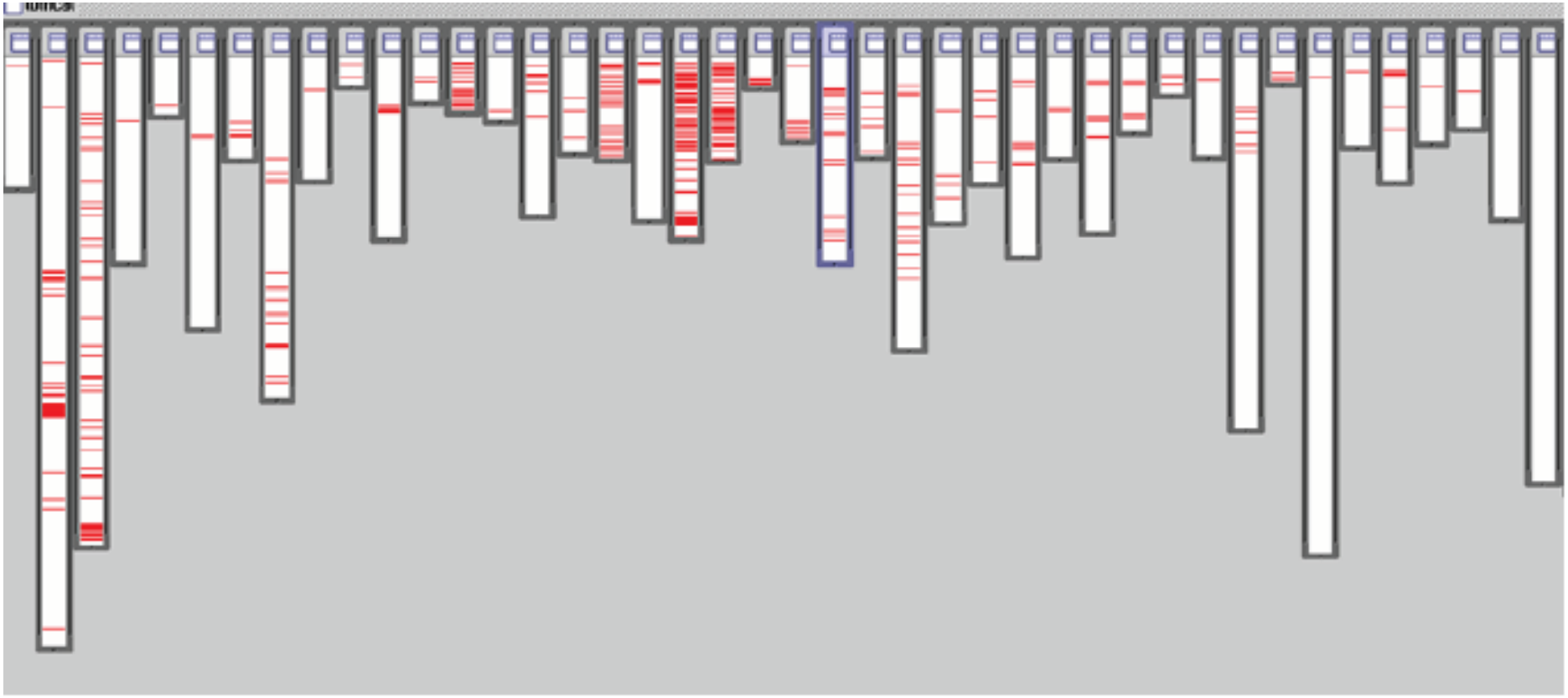
```
public class Account {

  private String number;
  private double balance;

  public void withdraw(double value) {
    if (getBalance() >= value) {
      setBalance(getBalance() - value);
      System.out.println("Success withdrawn!");
    }
  }
  ...
}
```

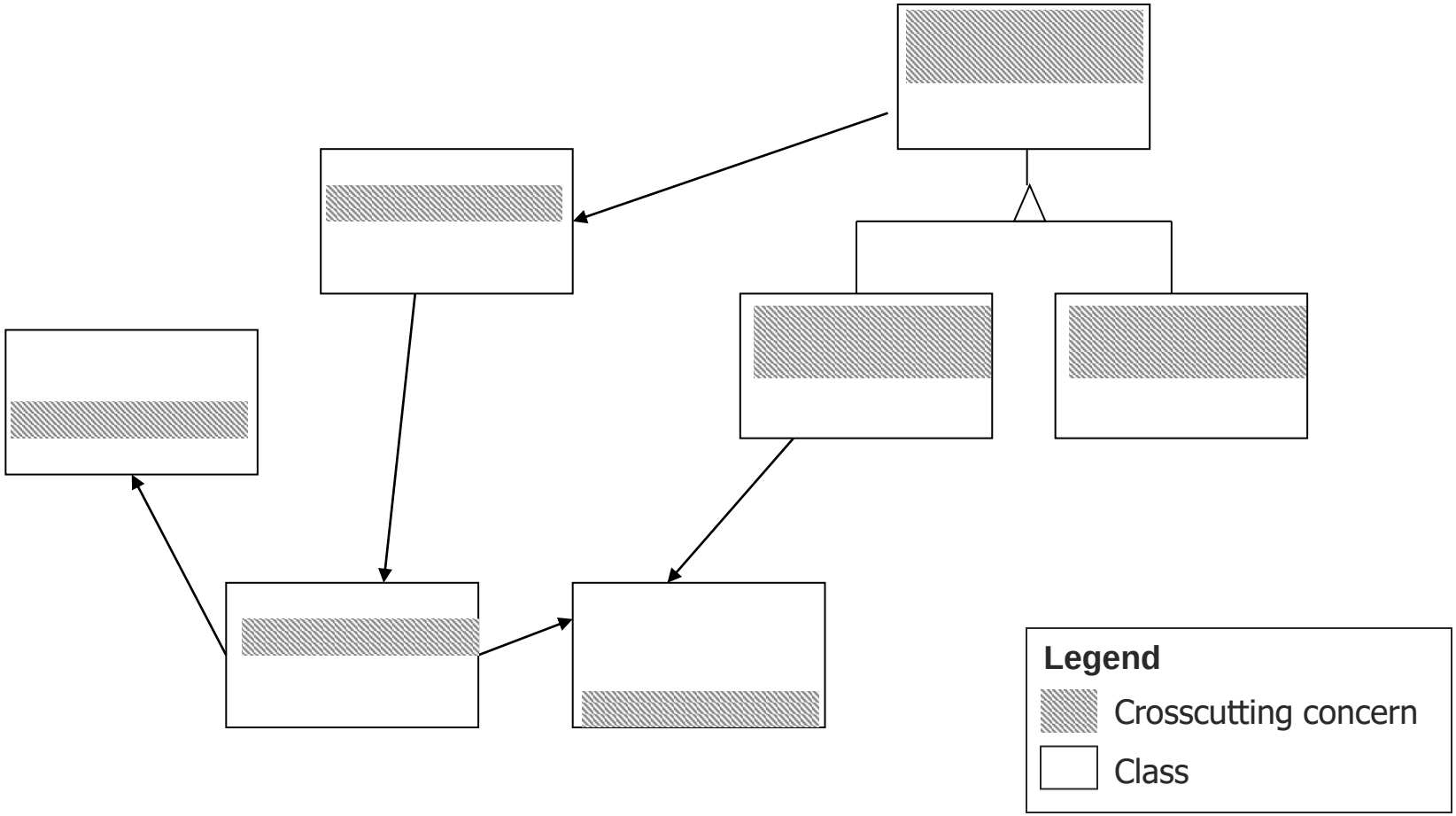
The red code implements Logging. It is scattered across several parts of the system code.

Logging in Tomcat

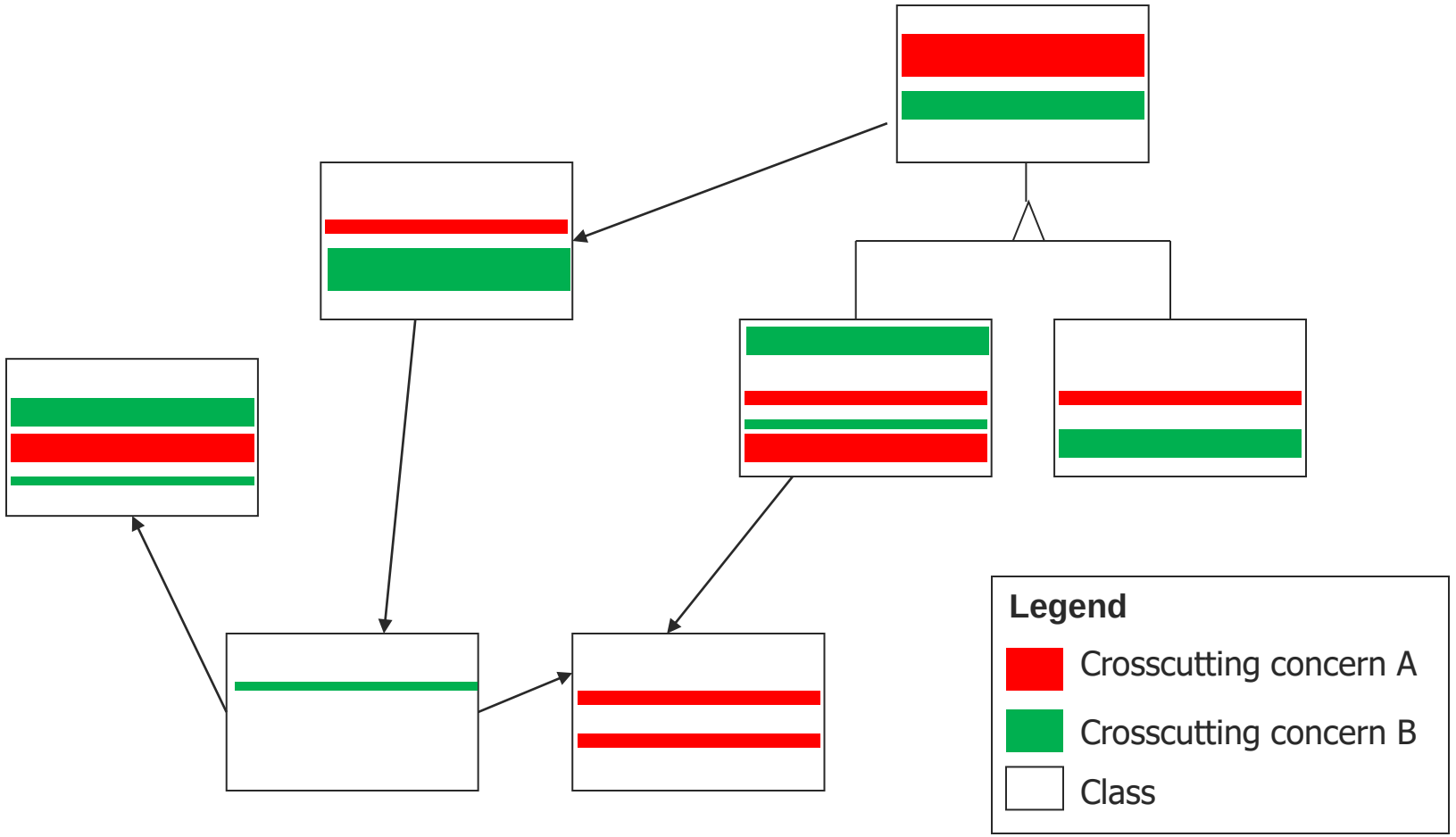


The red code implements Logging. It is scattered across several parts of the system code.

Scattering



Tangling


Legend
Crosscutting concern A
Crosscutting concern B
Class

Bibliography

- Ian Sommerville. **Software Engineering**, 9th Edition. Addison-Wesley, 2010.
  - Section 21.1 Separation of Concerns