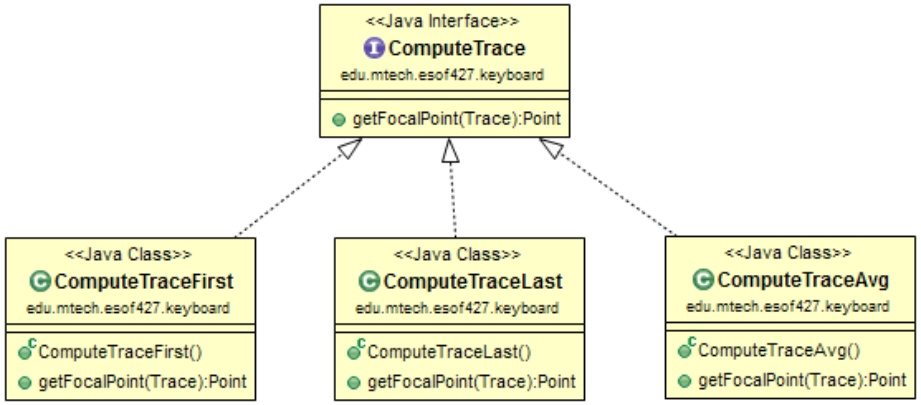
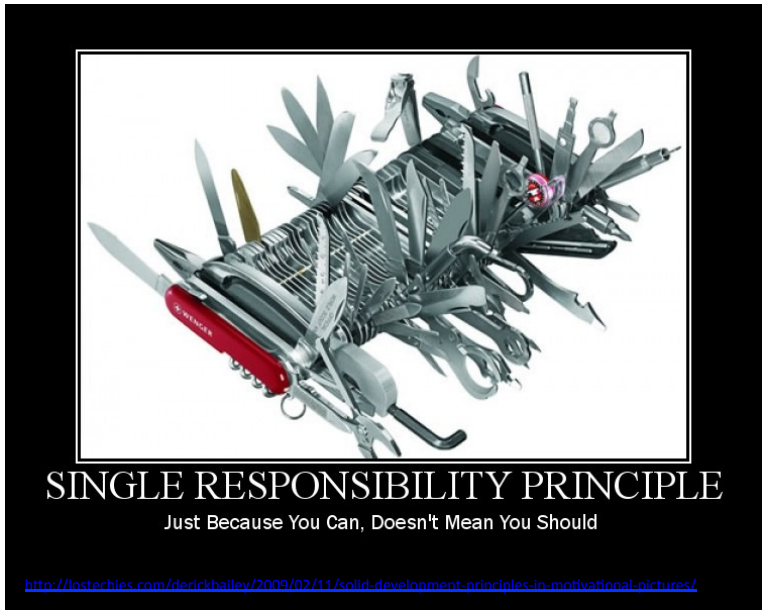
Software Design and Architecture






```java
@Test
public void testGetBoundingBox()
{
    Rectangle rect = new Rectangle(new Point(5.0, 5.0), 1.0, 1.0);

    Rectangle box = rect.getBoundingBox();
    assertEquals(rect.getLeft(), box.getLeft(), EPSILON);
    assertEquals(rect.getRight(), box.getRight(), EPSILON);
    assertEquals(rect.getTop(), box.getTop(), EPSILON);
    assertEquals(rect.getBottom(), box.getBottom(), EPSILON);
}
```

OOP and OOD

- How to build complex software?
  - Structured programming
    - gotos considered harmful
  - Object oriented programming (OOP)
- Language support alone isn't enough
  - (or even strictly necessary)
  - How we use the tools matters!
- Objected oriented design (OOD)
  - How do we design software that is easy to modify, extend and maintain?

Course topics

- OOP and OOD
- SOLID
- UML
- Design patterns
- Refactoring
- Testing
- C#
- ???

SOLID

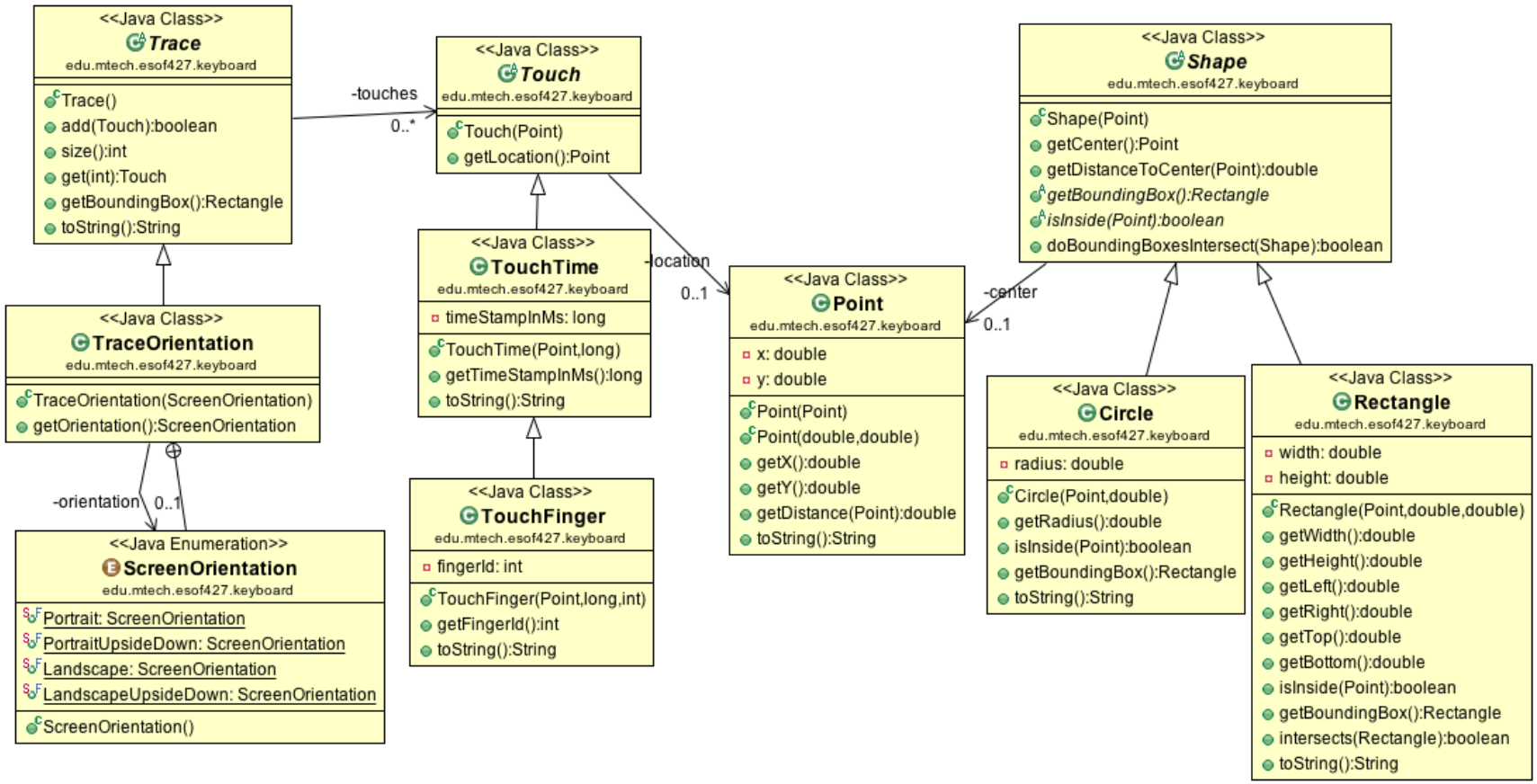
- How to manage dependencies in code?
  - The first five principles - SOLID

SRP	Single responsibility principle
OCP	Open/closed principle
LSP	Liskov substitution principle
ISP	Interface segregation principle
DIP	Dependency inversion principle

"Uncle Bob"

UML

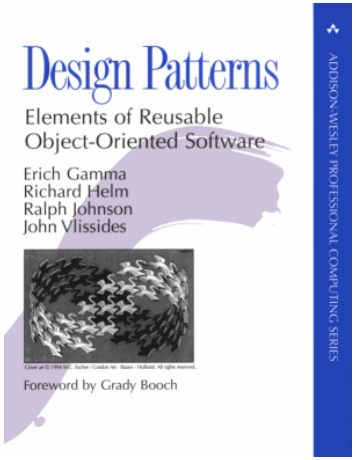
- Unified Modeling Language
  - Class diagrams

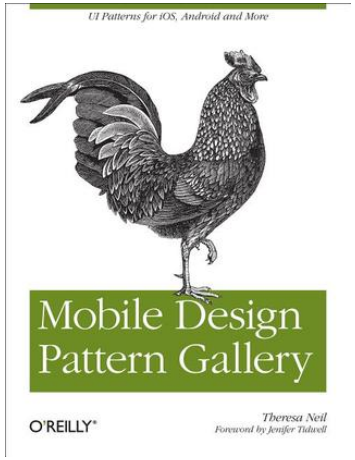


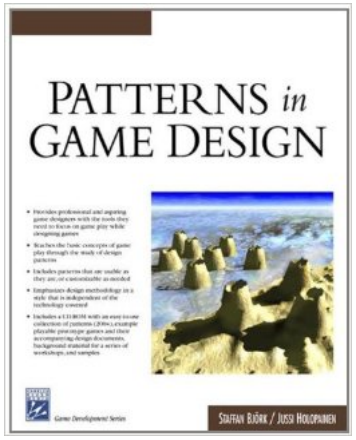
Design patterns

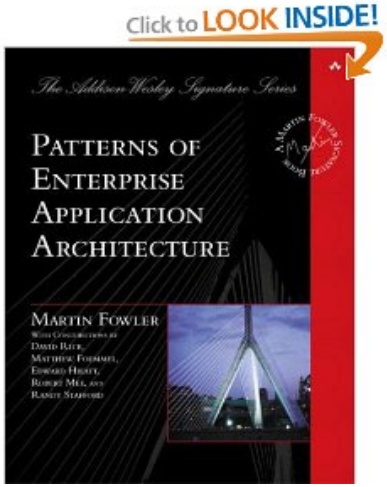
"A pattern is a solution to a problem in a context"

- Context, situation in which the pattern applies. Should be a *recurring* situation.
- Problem, goal you are trying to achieve. Also any constraints that occur in the context.
- Solution, a general design that anyone can apply resolving the goal and constraints.









Refactoring

- Refactoring:
  - Existing code base has gotten “stinky”
    - e.g. lots of repeated code
    - e.g. breaks in many places due to a single change
  - Improve maintainability
  - Make it easier and safer to handle future changes
    - Under some expected “axis of change”
  - Often towards specific design patterns

Testing

- Unit tests
  - Using JUnit
- Code coverage
  - Line/branch coverage using Eclipse plugin
- Test driven development

```
@Test
public void testGetBoundingBox()
{
    Rectangle rect = new Rectangle(new Point(5.0, 5.0), 1.0, 1.0);

    Rectangle box = rect.getBoundingBox();
    assertEquals(rect.getLeft(), box.getLeft(), EPSILON);
    assertEquals(rect.getRight(), box.getRight(), EPSILON);
    assertEquals(rect.getTop(), box.getTop(), EPSILON);
    assertEquals(rect.getBottom(), box.getBottom(), EPSILON);
}
```

C#

- Assume you know or can learn basics
  - e.g. loops, conditionals
- GUI frameworks
  - WPF vs. Windows Forms
- OOP features
- Parallel processing and concurrency
  - Threads, thread pools
  - Asynchronous patterns
  - Parallel programming
- Project using advanced C# and design patterns