Build Management & Strategy

9-14-2011
Opening Discussion

- Do you have any questions about the quiz?
- Presentations should go beyond the book. Assume everyone has done the reading.
To make life easy you want to use a build manager.

This lets you specify different things that can be done and how to do them.

Helps make sure you don't get lazy and skip steps.
Options

- make
  - The original and still broadly in use with C/C++.
- Ant
  - Created specifically for Java.
- Maven
  - Newer than Ant and gaining traction.
- SBT
  - Created for use with Scala.
- Many others.
Build Description

- make
  - Target: dependencies
    - Commands tabbed in
    - Uses an XML document.
    - `<project name="..." default="..." basedir="...">
      <property ...
      <path ...>...
      <target ...>...
    </project>`
You can follow the standard steps of TDD with build tools.

Make a test list and check things off.

The testing is running the build tool to see if everything works.
One significant recommendation from the book is to put tests in a separate tree from the other source.

The primary reason for this is that some tools, like javadoc, should not be run on test code.
These are approaches to solving problems that come up repeatedly in programming.

Fundamental rule: Abstract that which varies.
Motivating the Strategy Pattern

- Book uses the pay station with different towns.
- My simulations use different population types, different particle types, different force types.

Solutions

- Copy code
- Parameter and switch
- Inheritance with separate subtypes
- Composition of object that encapsulates rules
Methods of Change

- Change by Modification
  - To make a change in behavior you have to change the existing code.
- Change by Addition
  - To make a change in behavior you add additional code.
Delegation

- Giving parts of a task off to some other object/type.
- The new object/type has responsibility for handling a small piece of the whole problem.
The compositional approach is often called the Strategy Pattern.

The idea is that you want to abstract the strategy taken for some part of the problem.

That part of the problem is delegated to another object whose exact type can be varied easily without changing other parts of the code.
Book uses interface and different implementations for different strategies. This is the Java style.

In C++ this can be done with templates instead.

This is part of the flexibility of patterns.