Training : Parking Lot - Template Method

This page last changed on Dec 07, 2006 by dswood.

Template Method

Session Objectives	Session Overview
By the end of this session students should ... • Recognize duplication between subclasses and remove it with a Template Method • Understand and be able to implement a Template Method	**Activity** • Lab - Template Method • Define Pattern - Template Method

Session Notes

Lab - Template Method

Ask the students to find the duplication around the park() method. They should find that both ParkingLot.park() and Attendant.park() throw an exception if they can't park and notify their listeners after parking.

Ask the students to remove the duplication.

They should move the park() method up to ParkingFacility and introduce a template method:

Error formatting macro: code: Traceback (innermost last): File "<string>", line 1, in ? ImportError: no module named pygments

public void park(Object car)

Unknown macro: { if (! canPark()) throw new CannotParkCarException(); basicPark(car); notifyFacilityChanged(); }

Define Pattern - Template Method

Problem

- Duplication between methods on different subclasses of a common base class
- Most of the logic is the same, with only small differences

Pattern/Solution

- Pull the duplicated code into a method on the common base class
- Have the method on the base class call an abstract method, which is implemented differently on the two objects

Benefits/Liabilities

- Do not introduce inheritance just to use template method. If you have 2 separate classes with similar logic use a Strategy instead

Refactoring steps

- Make each method look as similar as possible
- Extract unique logic of each method into a separate method - make it abstract on superclass
- Pull up common code to superclass and call the abstract method