Design Patterns & Software Architecture
Template Method

dr. Joost Schalken-Pinkster

Windesheim University of Applied Science

The Netherlands

The contents of these course slides is (in great part) based on:
Chris Loftus, *Course on Design Patterns & Software Architecture for NEU*. Aberystwyth University, 2013.
Jeroen Weber & Christian Köppe, *Course on Patterns and Frameworks*. Hogeschool Utrecht, 2013.
Leo Pruijt, *Course on Software Architecture*. Hogeschool Utrecht, 2010-2013.

Session overview

- Template Method

Template method design pattern

Let’s find a design pattern

Will now present, on the board,
and using Eclipse,
a solution that utilises
the template method design pattern…

Case: Database tool design Requirements

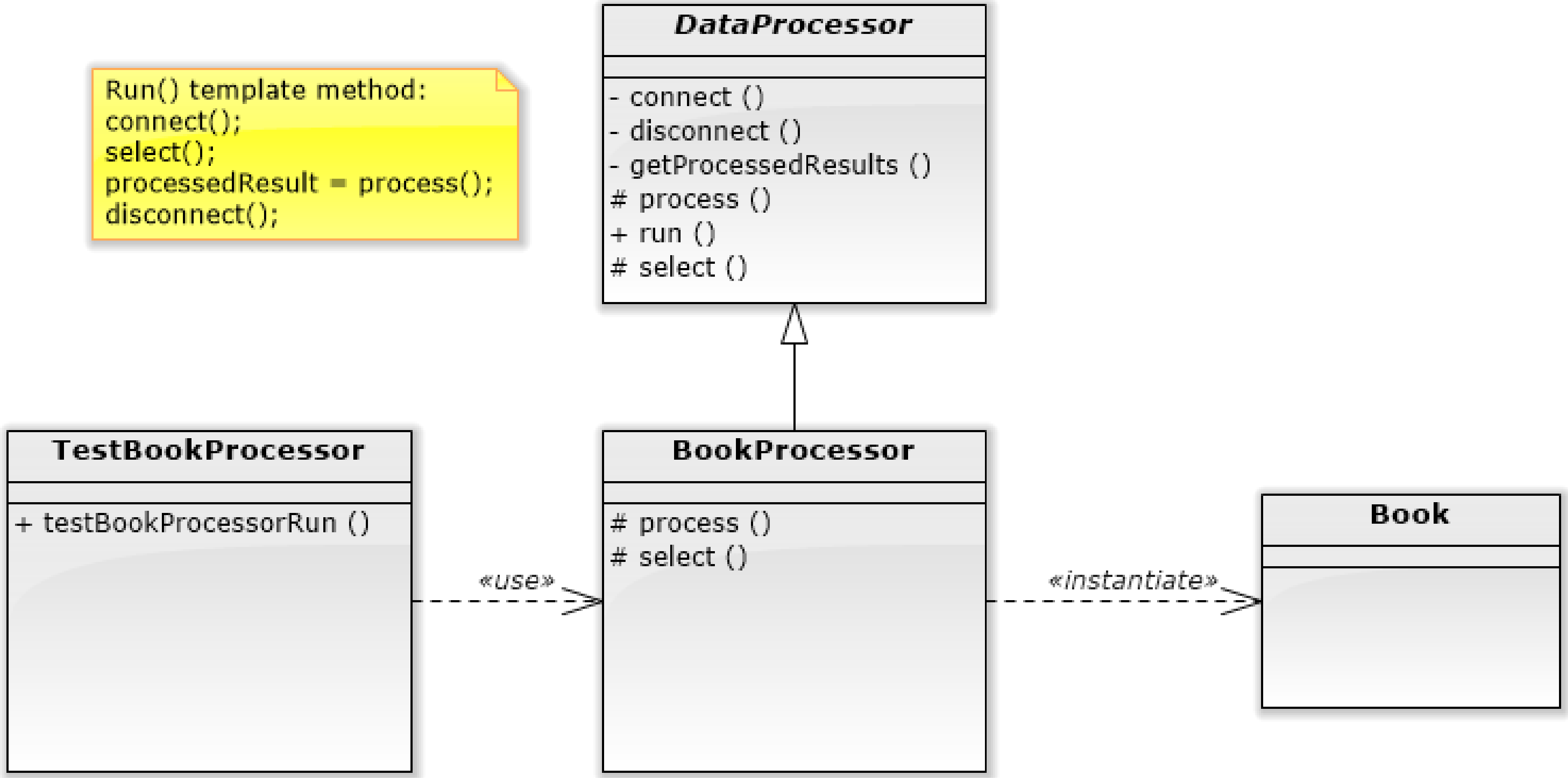
You have been asked by your manager to provide classes that allow access to various relational tables such that there is a way to select data from the tables and then process that data and possibly return a result list of the data:

1. Provide an abstract way to connect to the database
2. Be able to plug in various classes that work with different relational tables
3. All these classes should have the same operations to select and then process the selected data
4. It must be possible for the client code to obtain a list of processed data

Template Method pattern definition

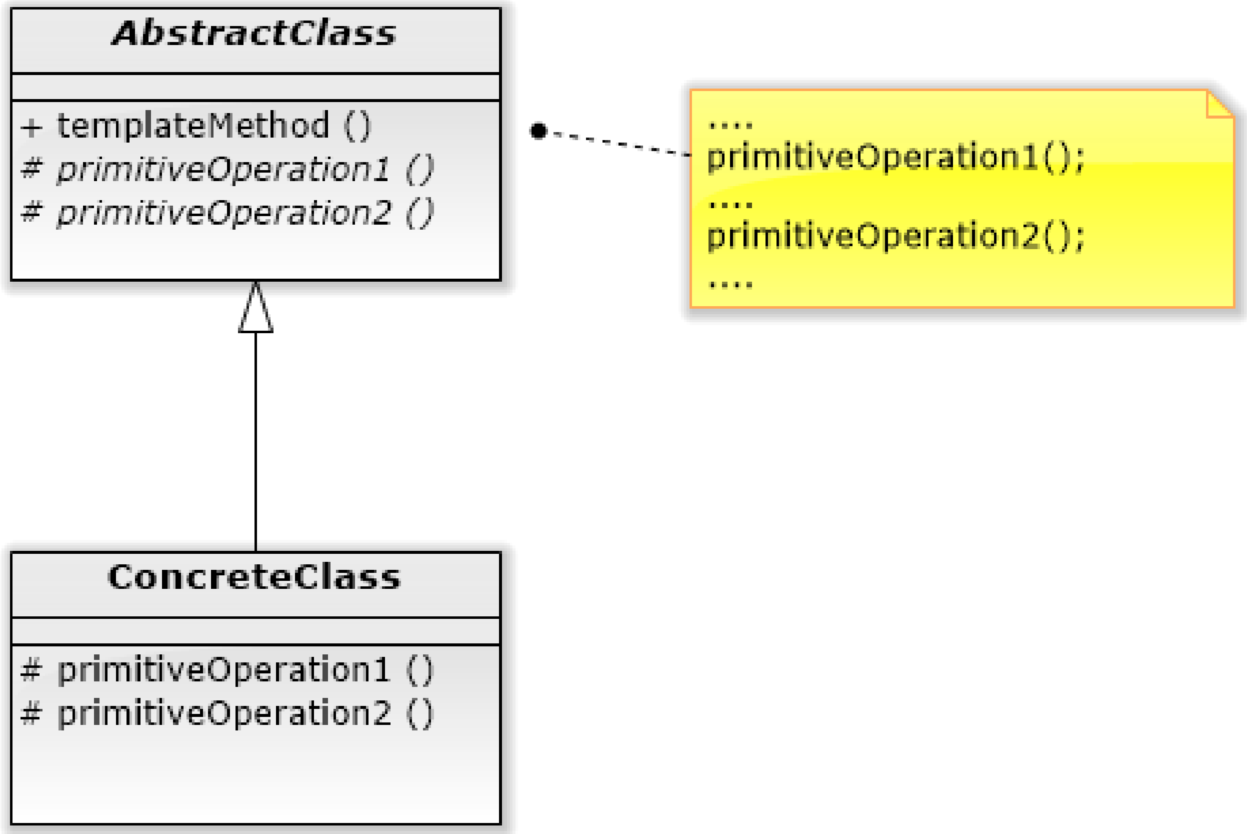
- **Intent**: Define the skeleton of an algorithm in an operation, deferring some steps to subclasses. Template Method lets subclasses redefine certain steps of an algorithm without changing the algorithm's structure.
- **Motivation**: A data-processing data access object is needed…

Motivation


Run() template method:
connect();
select();
processedResult = process();
disconnect();
DataProcessor
- connect ()
- disconnect ()
- getProcessedResults ()
# process ()
+ run ()
# select ()
TestBookProcessor
+ testBookProcessorRun ()
«use»
BookProcessor
# process ()
# select ()
«instantiate»
Book

- **Applicability**: Use template method when:
  - To implement the invariant parts of an algorithm once and let subclasses implement the behaviour that can vary.
  - When common behaviour in subclasses should be factored out into a super-class…
  - To control subclass extensions…

- **Structure:**



AbstractClass
+ templateMethod ()
# primitiveOperation1 ()
# primitiveOperation2 ()
....
primitiveOperation1();
....
primitiveOperation2();
....
ConcreteClass
# primitiveOperation1 ()
# primitiveOperation2 ()

- **Participants:**
  - AbstractClass (DataProcessor)
    - defines abstract primitive operations that concrete subclasses define to implement steps of an algorithm.
    - implements a template method defining the skeleton of an algorithm. The template method calls primitive operations as well as operations defined in AbstractClass or those of other objects.
  - ConcreteClass (BookProcessor)
    - implements the primitive operations to carry out subclass-specificsteps of the algorithm.

- **Consequences**: The template method pattern:
  - Is good for code reuse…
  - Supports Inversion of Control principle (also called the Hollywood principle) …
  - Template methods call the following kinds of operation:
    - AbstractClass methods…
    - Primitive abstract operations…
    - Factory methods…
    - Hook operations…

- **Implementation**:
  - Using Java access modifiers etc to control visibility and extendability (i.e. final)…
  - Minimize the number of primitive operations…
  - Naming conventions…

Design Principle: Don’t call us, we’ll call you!

<u>Holywood Principle:</u>

Don’t call us, we’ll call you!

E. Freeman, E. Freeman, B. Bates, K. Sierra, Head First Design Patterns. O'Reilly, 2004. p. 299

Related design patterns

- Template Method sounds a bit like the Strategy design pattern. How do they differ?...

Classroom discussion….

Reading

For this lesson please read:

- Chapter 8 (Encapsulating Algorithms) of Head First Design Patterns.