Discovering Design Patterns

The goal of this recitation is to detect commonalities in solutions to common problems and discover design patterns from those solutions. In the first part of this recitation you will compare two designs and describe an abstract pattern from them. In the second part of this recitation you will apply that pattern to solve other design problems.

Discovering a design pattern

Consider the following design problems:

> **Alarm Clock:** When an alarm clock goes off in a computer system, several actions can be taken; the user can add and remove these actions in the running system. Examples of actions are: (1) the user is visually notified that the alarm has gone off, (2) a sound is played, and (3) the computer is scanned for viruses.

> **Customer Management:** You wish to maintain a customer list in a business sales application. Multiple independent tasks are executed when a new customer is added. First, the system automatically generates and prints a welcome letter for the customer. Second, the customer list in the user interface is updated to reflect the change.

Task 1: With a partner, discuss how you would implement solutions for the above problems. Your solutions should be extensible (i.e. you should be able to add alarm actions or customer management tasks with minimal code changes) and separate concerns to maximize cohesion. We recommend that you draw a rough UML diagram to explain your designs.

Task 2: In the `alarmclock` and `customerlist` packages we have provided example implementations for both systems. We also describe our design for the customer management system in a UML diagram on the back of this page. Explore the code and UML diagram to understand how our solutions solve the design problems. With a partner, discuss the extensibility and cohesion of the sample solutions. Explicitly describe the changes you would need to add an alarm action or a customer management task to the respective systems.

Task 3: Describe the commonalities of these solutions as a design pattern. To do this you will need to infer which parts of the solutions are specific to the given problems and which parts are more general as a reusable design idea, and describe the commonalities of the solutions in a general UML class diagram. Choose suitable abstract names for the key components of the pattern, and name the pattern itself. Finally, identify when the pattern is applicable and discuss the consequences, both positive and negative, of applying the pattern.

Using your new design pattern

Consider the following scenario:

> **Logging:** When writing large applications it is useful to have a dedicated system for displaying and storing log messages. There are many different ways of handling log messages. For example, we might write all debug log messages to the console with the prefix `[Debug]`; we might also write all messages to a log file or send an email to the system administrator with the error message, or take multiple of those actions.

Task: In the `loggingsystem` package we provide a very simple logger implementation that has two hard-coded logging mechanisms. Rewrite the code to use the discovered pattern such that different loggers could be easily configured and added.

A UML class diagram for the `CustomerList` system