SWEN30006 Software Modelling and Design
Workshop 8: GoF Patterns (Part 2)
Strategy and Composite

School of Computing and Information Systems
University of Melbourne
Semester 2, 2019

Completion: You need to demonstrate your solutions for **every exercise** to your tutor before the end of the workshop. If you do not complete the exercises in time, finish the remaining tasks by next week's workshop and present these to your tutor at the start of the workshop. Your tutor must review your work for you to be eligible to receive a mark for this workshop. See the Workshop Overview under Subject Information on the LMS for more details.

Requisite Knowledge and Tools

To complete this workshop you will need to be familiar with the following terms and concepts:

- Class Diagram
- Sequence Diagrams
- GRASP principles
- GoF Strategy and Composite

Introduction

This week we will be focusing on applying two GoF patterns, **Strategy** and **Composite**, to a particular problem. The exercises begin with designing a system using provided information, then analysing how the GoF patterns can be used to solve design problems. Finally, we will finish by implementing the system. These exercises should allow you to gain an understanding of particular design problems through on a concrete example and demonstrate how to apply design patterns to solve the problems. **These exercises are to be done individually**.

GoF Patterns: Strategy and Composite

GoF (Gang-Of-Four) patterns are a set of design patterns that provide solutions to common software design problems. GoF patterns are specializations of the GRASP building blocks, and constructed to address a specific problem. GoF patterns are a useful tool for software developers. These patterns can be considered as generalised solutions or templates that solve real-world problems. However, applying patterns on a trivial problem may result in overcomplicated implementation. Hence, it is important to understand the concept of the design patterns and their benefits when applying them. Below is a brief summary of two GoF patterns that we will be using in this week's exercises.

- **Strategy:** The strategy pattern is a behavioural pattern. It is useful when a system uses various algorithms (i.e., strategies) for a particular problem and the specific algorithm used is chosen at run-time. To achieve protected variation of GRASP principles, the Strategy pattern provides a solution by using polymorphic objects (i.e., by creating an interchangeable 'family' of algorithms). This pattern is often used in combination with the Factory pattern to determine which algorithm to use.
- **Composite:** The composite pattern is a structural pattern, which also offers protected variation. The pattern defines a manner in which to design a recursive tree structure of objects, where individual objects and groups can be accessed in the same manner.

The System Under Discussion: Monopoly Express

This week you will be creating a system for playing Monopoly Express. Monopoly Express is a board game for 2 to 4 players. To play this game, players need 1 Gameboard and 10 dice (see example below). To win the game, players have to be the first to earn $15,000. They do this by collecting the money points based on several rules. Table 1 shows the faces for each of 10 dice used in the game. Below are the instructions for Monopoly Express.

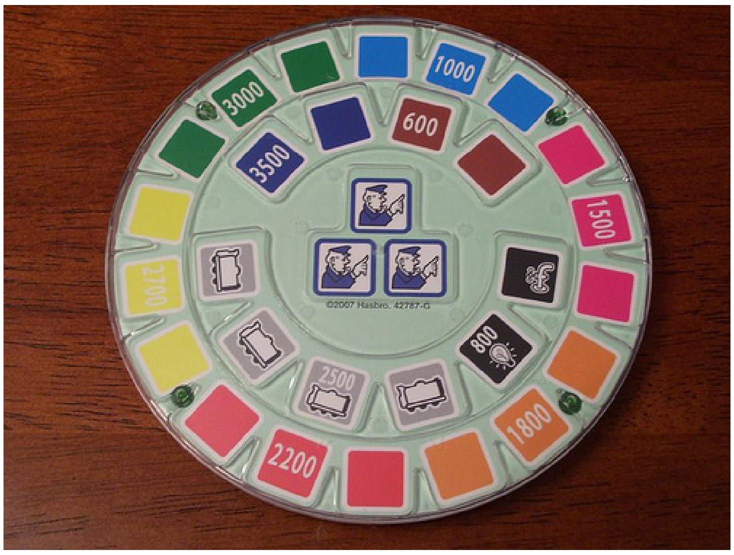


Figure 1: An example of a Monopoly Express board.

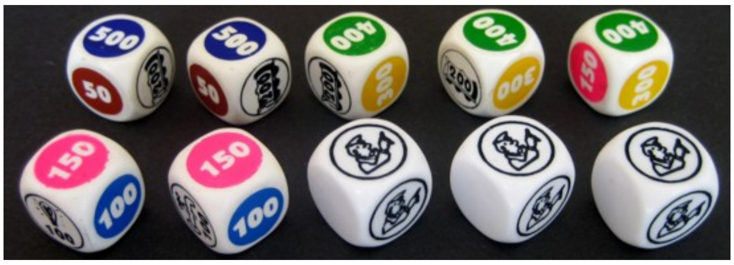


Figure 2: An example of Monopoly Express dice.

Table 1: Dice Faces. **A Java package of these dice is provided.**

Die #	Faces					
Die #1	500	200	railroad (200)	100	50	50
Die #2	500	200	railroad (200)	50	50	50
Die #3	400	300	250	200	railroad (200)	200
Die #4	400	300	250	200	railroad (200)	200
Die #5	400	300	250	200	150	150
Die #6	150	150	100	100	100	Utility (100)
Die #7	150	150	100	100	100	Utility (100)
Die #8	police man	<*blank*>	<*blank*>	police man	<*blank*>	<*blank*>
Die #9	police man	<*blank*>	<*blank*>	police man	<*blank*>	<*blank*>
Die #10	police man	<*blank*>	<*blank*>	police man	<*blank*>	<*blank*>

Table 2: Number Group and its value

Group	#Dice	Money Points
50 (Brown)	2 Dice placed on the 50 squares	600
100 (Light blue)	3 Dice placed on the 100 squares	1000
150 (Pink)	3 Dice placed on the 150 squares	1500
200 (Orange)	3 Dice placed on the 200 squares	1800
250 (Red)	3 Dice placed on the 250 squares	2200
300 (Yellow)	3 Dice placed on the 300 squares	2700
400 (Green)	3 Dice placed on the 400 squares	3000
500 (Dark blue)	2 Dice placed on the 500 squares	3500
Railroads	4 dice placed on the railroad squares	2500
Utilities	2 dice placed on the utility squares	800
Police	3 dice placed on the police squares	Your turn ends. No scoring for this turn.

Game Instructions

For each player's turn:

1. Roll all 10 dice. Look at the police dice (Die #8, #9, and #10). If you rolled a police man, place that dice on a matching square in the board. If you roll a blank for police dice, it is worth nothing
   - If all 3 of the police squares are filled with the police dice, your turn immediately ends. You do not score anything for this turn. Remove all the dice on the board.
2. Now look at the remaining dice that you rolled to see which groups you can collect (see groups in Table 2). Then decide which groups you want to collect. This will determine your rewards. For example, if you complete the 100 group, you'll earn $1000. If you rolled four railroads and decide to collect, you earn $2500. After deciding which group(s) you want to collect, place the dice on their matching squares. *Once dice are placed on the board, they cannot be moved or removed.*
3. You can roll the remaining dice (i.e., the dice that are not placed onto the squares on the board) as many times as you want, or you can decide to end your turn. If you decide to roll again, follow steps 1 through 3 for the remaining dice. *Be aware that if three police squares are filled, you do not score anything for this turn!* If you decide to end your turn, you score will be calculated as follow:
   - **Rule 1:** For each complete number group, sum the money points according to the chart in Table 2 for that group.
     Note: Rule 1 applies to every turn you play.
   - **Rule 2:** For any incomplete groups, only the group with the highest summation of the dice values is chosen. For example, two 250 dice and 1 300 dice are placed on the board, you will earn money points of 500 (250+250).
     Note: Rule 2 applies to every 2nd turn you play (e.g., the 2nd, 4th turns).

- **Rule 3:** All money points shown on the faces of dice that are placed on the board are counted towards your score.
  Note: Rule 3 applies to every 3rd turn you play (e.g., the 3rd, 6th turns).

  When multiple rules are applicable, they cannot be combined. Only the rule that has the highest score is counted toward your score.

4. After note down your money points, remove all dice from the board. You turn is over.

Your task

Design and implement the system for playing this Monopoly Express game this game with 2 players.

> **Info:** Java packages for a board and dice, and a class for user interface are provided. Figure 3 shows a Class Diagram of these packages.

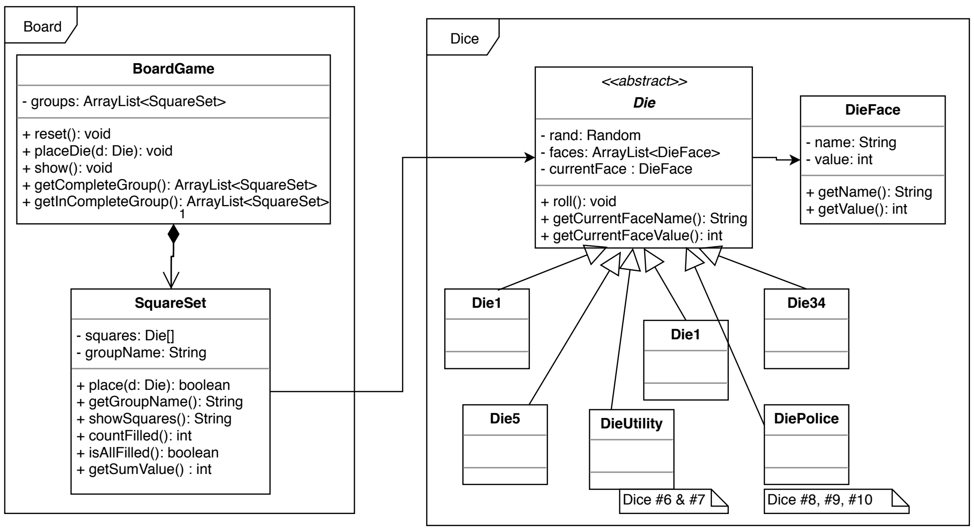


Figure 3: A Class Diagram of provided Java packages (i.e., board and dice).

Part 1 Designing the System

Before implementing the system, you need to design a system for Monopoly Express game in order to understand what data and classes are needed.

Task 1.1 Creating Domain Model

Using the description in the previous section, create a Domain Model for the problem domain as you understand it.

Task 1.2 Identifying Necessary Patterns

Based on the use cases and your Domain Model, identify how the GoF Strategy and Composite patterns could be used. Also consider any other applicable patterns.

Discuss with your neighbour and justify why should we apply these patterns to those objects.

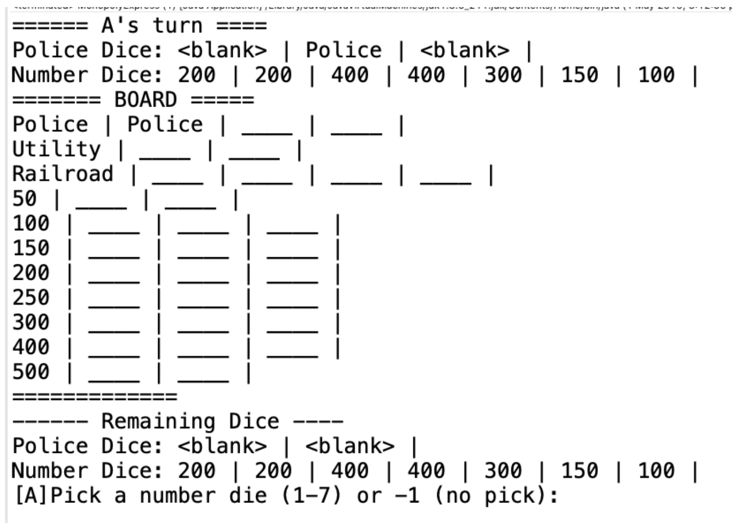
Task 1.3 Creating a Design Model

Draw a Class Diagrams for the system. Include all objects that are part of a pattern, and any directly related objects.

Part 2 Implementing the System

Using your design from Part 1, implement Monopoly Express in Java. You should create the appropriate classes. The partial user interface and the packages of board and dice are provided in `MonopolyExpress.zip` (excluding a die cup class). Figure 4 shows an example of the user interface of the game.

It is required to have a consistency between your design and implementation. By now, you should also be aware of GRASP principles and ensure that your design and implementation do not violate them.



```
====== A's turn ====
Police Dice: <blank> | Police | <blank> |
Number Dice: 200 | 200 | 400 | 400 | 300 | 150 | 100 |
======= BOARD =====
Police | Police | ____ | ____ |
Utility | ____ | ____ |
Railroad | ____ | ____ | ____ | ____ | ____ |
50 | ____ | ____ |
100 | ____ | ____ | ____ |
150 | ____ | ____ | ____ |
200 | ____ | ____ | ____ |
250 | ____ | ____ | ____ |
300 | ____ | ____ | ____ |
400 | ____ | ____ | ____ |
500 | ____ | ____ |
=============
------ Remaining Dice ----
Police Dice: <blank> | <blank> |
Number Dice: 200 | 200 | 400 | 400 | 300 | 150 | 100 |
[A]Pick a number die (1-7) or -1 (no pick):
```

Figure 4: An example of the user interface for Monopoly Express