Victoria University of Wellington
School of Engineering and Computer Science

SWEN221: Software Development

Assignment 5

Due: 23:59pm Sunday 11th June

1 Introduction

You are tasked with testing (and fixing) a simple implementation of the well-known *Monopoly* board game. If you have not heard of the game Monopoly before, don't worry — a simple introduction is provided below.

1.1 Monopoly Board

A Monopoly board is divided into two main areas: the border region; and, the inner region. There are 40 locations on the board, each of which is either: a *property* or a *special area*. Properties have a name, a price, a mortgage value, a rental value and can be bought or sold; the special areas are: "Go", "Free Parking", "Jail", "Goto Jail", "Chance" and "Community Chest".

A property is either: a *street*, a *railway* or a *utility*.

Each street belongs to a *colour group*. Each colour group consists of two or three consecutive streets. Each street can have up to five *houses* or one *hotel* built on them. Houses can only be built on a street when *all* the streets in its colour group are owned by the same player. A hotel can only be built on a street which has five houses and, when built, all houses on that street are removed. The rental value of a street depends upon its *base rate* and the number of houses/hotels on that street. The calculation is:

$$\text{Street rent} = \text{base rate} + 25 * (\text{number of houses}) + 200 * (\text{number of hotels})$$

If the player owns all properties in a colour group, then the *base rate* for any property in that group is doubled. The cost of building a house or hotel on a particular street depends upon the street in question. For those on the south side (e.g. Euston), the cost is \$50 per house/hotel; for those on the west side (e.g. Bow Street), the cost is \$100 per house/hotel; for those on the north (e.g. The Strand) it's \$150 per house/hotel; finally, those on the east side (e.g. Mayfair) it's \$200 per house/hotel.

There are four railways in the board: "Marylebone station", "Fenchurch Street station", "Liverpool Street station" and "King's Cross station". The rental cost for a station depends upon how many stations are owned by the player. The calculation is:

$$\text{Station rent} = \text{base rate} * \text{number of stations owned}$$

There are two utilities in the board: the "*Water Works*" and the "*Electric Company*". Again, the rental cost for these depends upon how many are owned by the player:

$$\text{Utility rent} = \text{value of last dice roll} * 4 * \text{number of utilities owned}$$

Properties can be mortgaged through the bank at any time; when this happens, the bank loans the player half the value of the property. However, rent cannot be collected on mortgaged properties. To unmortgage a property, the player must pay back the loan, plus 10% interest. A property can be mortgaged and unmortgaged as many times as one wants. However, it has to be unmortgaged before it can be mortgaged again. Unmortgaged properties may be sold at any time for the original purchase price.

An illustration of the Monopoly board is given in Figure 1. This includes the price and base rate of each property and identifies all the special zones.

Tip: you might find it useful to print Figure 1!

1.2 Playing the Game

Each player in the game has a token which is placed onto the location the player is currently visiting. There are eight tokens: the "*ScottishTerrier*", the "*Battleship*", the "*Automobile*", the "*TopHat*", the "*Thimble*", the "*Boot*", the "*WheelBarrow*" and the "*Iron*".

At beginning, all the tokens are placed at the location "GO". As the game proceeds, in each player's turn, the player rolls the dice and moves the number of locations equal to their sum in the clockwise direction.

When a player passed through the location "GO", $200 is given to that player.

Each player maintains a list of properties they own, and the amount of cash they have left. Initially, all the properties are unowned. If the person lands on an unmortgaged property owned by another, then he/she must pay its rent to the owner. If the player lands on an unowned property, they have the option to purchase it for the stated price.

NOTE: for fun, you can play the game using the simple `TextClient` interface provided.

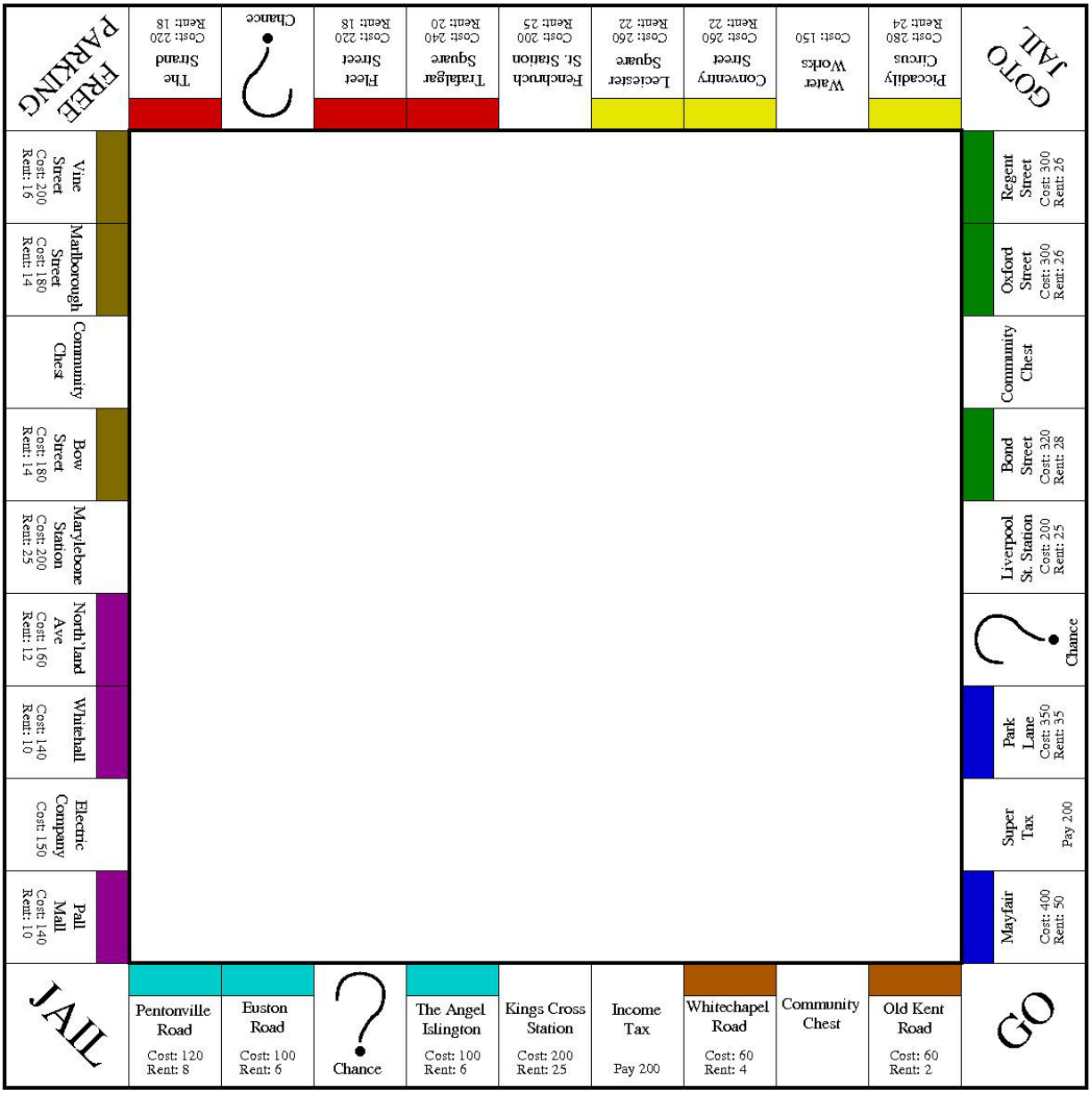
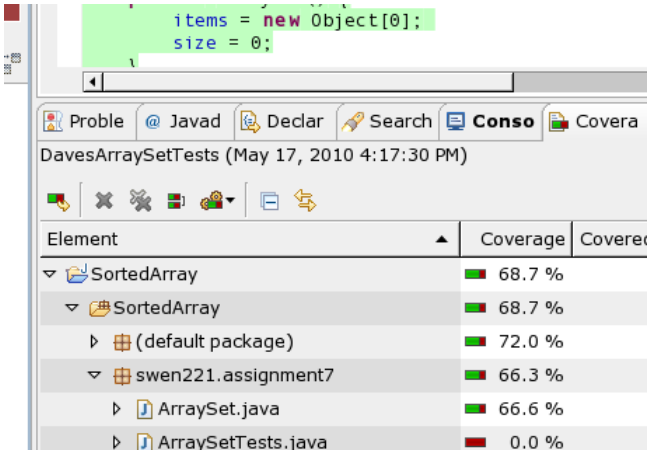


Figure 1: The Monopoly Board

What to Do

Currently, there is one test provided for the monopoly game in the file `MonopolyTests.java`. You must add tests to this file for the classes in package `swen221.monopoly`. The aim is to achieve close to 100% coverage using Emma and, in the process, identify and fix problems in the code. The following illustrates Emma being used in Eclipse:



In this example, we see that the coverage obtained for the class `ArraySet` is 66.6%.

HINT: The purpose of using Emma is to help you find problems in the code base. These problems should be relatively clear (based on the specification) once you have found them, but have been deliberately obfuscated to make it hard to find them just by looking at the code.

HINT: In this case, a good strategy for writing test cases is for each test to construct a mini-game of Monopoly and move one or more players around the board performing operations. This is because the specification given is defined in terms of *what players can do*, rather than giving specific details about individual methods.

HINT: There are *at least eight* distinct problems that have been planted in the code base. See if you can find them all!

HINT: Your submitted code will be marked against a hidden set of test cases containing more than 32 tests and which achieves 100% coverage.

HINT: The specification of Monopoly given here *differs from the standard rules of Monopoly*. You should only test against the specification given here.

Submission

Your source files should be submitted electronically via the *online submission system*, linked from the course homepage. The minimal set of required files is:

```
swen221/monopoly/Board.java
swen221/monopoly/GameOfMonopoly.java
swen221/monopoly/locations/ColourGroup.java
swen221/monopoly/locations/Location.java
```

```
swen221/monopoly/locations/Property.java
swen221/monopoly/locations/SpecialArea.java
swen221/monopoly/locations/Station.java
swen221/monopoly/locations/Street.java
swen221/monopoly/locations/Utility.java
swen221/monopoly/Player.java
swen221/monopoly/testing/MonopolyTests.java
```

You must ensure your submission meets the following requirements (which are needed for the automatic marking script):

1. **Your submission is packaged into a jar file, including the source code**. *Note, the jar file does not need to be executable.* See the following Eclipse tutorials for more on this:

   `http://ecs.victoria.ac.nz/Support/TechNoteEclipseTutorials`

2. **The names of all classes, methods and packages remain unchanged**. That is, you may add new classes and/or new methods and you may modify the body of existing methods. However, you may not change the name of any existing class, method or package. *This is to ensure the automatic marking script can test your code.*

3. **All testing mechanism supplied with the assignment remain unchanged.** Specifically, you cannot alter the way in which your code is tested as the marking script relies on this. However, this does not prohibit you from adding new tests. *This is to ensure the automatic marking script can test your code.*

4. **You have removed any debugging code that produces output, or otherwise affects the computation.** *This ensures the output seen by the automatic marking script does not include spurious information.*

Note: Failure to meet these requirements could result in your submission being reject by the submission system and/or zero marks being awarded.

Assessment

This assignment will be marked as a letter grade (A+ ... E), based primarily on the following criteria:

- **Correctness (90%)** — does submission adhere to given specification.
- **Style (10%)** — does the submitted code follow the style guide and have appropriate comments (inc. Javadoc)

As indicated above, part of the assessment for the coding assignments in SWEN 221 involves a qualitative mark for coding style. Coding style is an important aspect of coding since all code should be expected to be revisited again, i.e., it is important that you are someone else can make sense of the code it in the future and change it without accidentally introducing errors.

The qualitative marks for style are given for the following points:

- **Division of Functionality into Classes**. This refers to how *cohesive* your classes are. That is, whether a given class is responsible for single specific task (has high cohesion), or for many unrelated tasks (has low cohesion). In particular, big classes with lots of weakly related functionality should be avoided.

- **Division of Work into Methods**. This refers to how well a given task is split across methods. That is, whether a given task is broken down into many small reusable methods (good) or implemented as one large monolithic method (bad).

- **Self-Explanatory Naming**. This refers to the choice of names for the classes, fields, methods and variables in your program. First, naming should be consistent and follow the recommended Java Coding Standards (see `http://g.oswego.edu/dl/html/javaCodingStd.html`). Secondly, names of items should be descriptive and reflect their purpose in the code.

- **JavaDoc Comments**. This refers to the use of JavaDoc comments on classes, fields and methods. We expect all `public` and `protected` items to be properly documented. For example, when documenting a method, an appropriate description should be given, including despriptions for (if present) its parameters and return value. Good style dictates that `private` features are documented as well.

- **Code Comments**. This refers to the use of commenting within a given method. Comments should be used to elaborate the purpose of the code, rather than simply repeating what is evident from the code already.

- **Layout**. This refers to the consistent use of indentation and other layout conventions. Code must be properly indented and make consistent use of conventions e.g. for placing curly braces.

In addition to a mark, you should expect some written feedback, highlighting the good and bad points of your solution.