# PROJECT

Handout: **Tuesday, 11 October 2016**
Due: **Monday, 4 December 2016**

**This is a team project. Each team may have 2 to 3 members. Please team up on Blackboard\Groups before Oct. 14, 2016. Afterwards, we will randomly group the ones with no team.**

## 1 OVERVIEW

The goal of this project is to develop a board game named Monopoly[1].

## 2 PROJECT GRADING

You can earn in total at most 100 points in the project. The points are composed as follows:

- Requirements coverage (4 points for each requirement, in total 40 points)
- Code quality (10 points for Object-Oriented Design and 10 for code inspection results, in total 20 points)
- Statement coverage of unit tests (15 points)
- Presentation (10 points)
- Final report (15 points)
- Bonus points (10 points for each bonus feature in Section 6, in total 20 points)

## 3 CODE INSPECTION

The inspection tool[2] of IntelliJ IDEA checks your program to identify potential problems in the source code. We have prepared a set of rules to be used in grading (included in the same archive as this description). Import the rule set file into your IntelliJ IDEA IDE and check your code against the rules.

The inspection tool is able to check for many potential problems, but only a few of them are considered errors by the provided rule set. 2 Points will be deducted for each error in your code reported by the tool.

## 4 STATEMENT COVERAGE OF UNIT TESTS

Statement coverage[3] is a measure used in software testing to describe the percentage of statements that has been tested: in general, the higher the coverage, the more reliable the test results. In this project we will use the built-in tool[4] of IntelliJ IDEA to collect statement coverage information on your tests.

Your will get 15 base points for statement coverage between 95% and 100%, and 14 base points for coverage between 90% and 94%, and so on. You will get 0 base points for statement coverage below 25%.

Your final grade will be calculated as your base points multiplied by the percentage of your requirement coverage. For example, if you only implement 60% of the requirements and you achieved 95% statement coverage of your code, then your points for this part will be 9 = 15 * 60%.

## 5 GENERAL NOTES

- JDK 1.8 will be used in grading your project. Make sure you use the same version of JDK for your development.
- Your code for implementing the regular features should not rely on any library that is not part of the standard API of Java SE 1.8. You may use third party libraries to implement the bonus features.
- The game should handle incorrect user input graciously. For example, when a user input is invalid, the game should not crash.
- You may refer to the RabbitRace class from Lab 1 for implementing randomness (e.g. in rolling a dice).
- QuickHelp.pdf includes a simple introduction to the sample project structure (SampleProject.zip) and how to use the related tools in IntelliJ IDEA.

1. https://en.wikipedia.org/wiki/Monopoly_(game)
2. https://www.jetbrains.com/help/idea/2016.2/code-inspection.html
3. http://en.wikipedia.org/wiki/Code_coverage
4. https://www.jetbrains.com/help/idea/2016.2/code-coverage.html

# 6 PROJECT DETAILS

The game comes with a board divided into 20 squares (Figure 1), a pair of four-sided (tetrahedral) dice, and can accommodate two to six players.

It works as follows:

- Players have money and can own property. Each player starts with HKD 1500 and no property.
- All players start from the first square ("**Go**").
- One at a time, players take a turn: roll the dice and advance their respective tokens clockwise on the board. After reaching square 20 a token moves to square 1 again.
- Certain squares take effect on the player (see below), when his token passes or lands on the square. In particular, it can change the player's amount of money.
- If after taking a turn a player has a negative amount of money, he retires from the game. All his property becomes unowned.
- A round consists of all players taking their turns once.
- The game ends either if there is only one player left or after 100 rounds. The winner is the player with the most money after the end of the game. Ties (multiple winners) are possible.

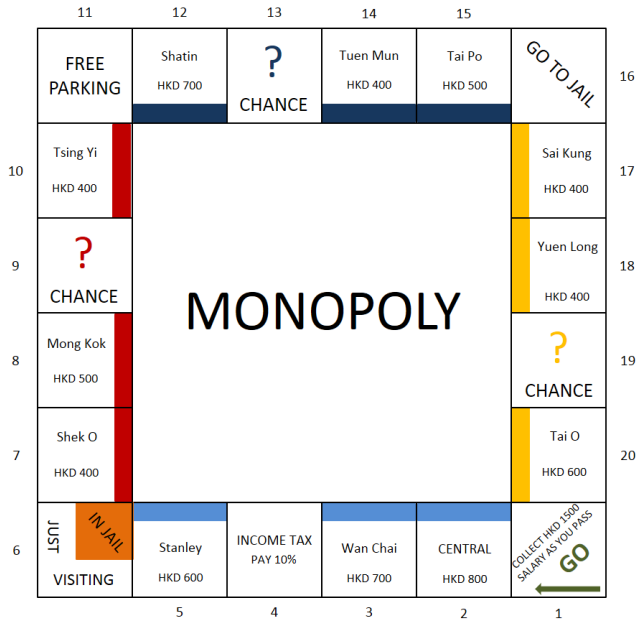


**Figure 1. Monopoly board**

There are the following kinds of squares on the board:

- **Property squares (marked by a colored stripe).** They contain the name and the price of the property and can be owned by players. If a player lands on an unowned property, he can choose to buy it for the written price or do nothing. If a player lands on a property owned by another player, he has to pay a rent (rent amounts are listed in Table 1).
- **Go.** Every time a player passes through (not necessarily lands on) this square he gets HKD 1500 salary.
- **Chance.** If a player lands on one of these squares he either gains a random amount (multiple of 10) up to HKD 200 or loses a random amount (multiple of 10) up to HKD 300.
- **Income tax.** If a player lands on this square he pays 10% of his money (rounded down to a multiple of 10) as tax.
- **Free parking.** This square has no effect.
- **Go to Jail.** If a player lands on this square he immediately goes to the "**In Jail**" part of the "**In Jail/Just Visiting**" square.
- **In Jail/Just Visiting.** If a player lands on this square he is "**Just Visiting**": the square has no effect. However, if the player got here by landing on "**Go to Jail**", he is in Jail and cannot make a move. A player gets out of Jail by either throwing doubles[1] on any of his next three turns (if he succeeds in doing this he immediately moves forward by the number of spaces shown by his doubles throw) or paying a fine of HKD 50 before he rolls the dice on either of his next two turns. If the player does not throw doubles by his third turn he must

**Table 1: Properties**

| Position | Name | Price | Rent |
|---|---|---|---|
| 2 | Central | 800 | 90 |
| 3 | Wan Chai | 700 | 65 |
| 5 | Stanley | 600 | 60 |
| 7 | Shek O | 400 | 10 |
| 8 | Mong Kok | 500 | 40 |
| 10 | Tsing Yi | 400 | 15 |
| 12 | Shatin | 700 | 75 |
| 14 | Tuen Mun | 400 | 20 |
| 15 | Tai Po | 500 | 25 |
| 17 | Sai Kung | 400 | 10 |
| 18 | Yuen Long | 400 | 25 |
| 20 | Tai O | 600 | 25 |

[1] When both dice come out the same face up.

pay the HKD 50 fine. He then gets out of Jail and immediately moves forward the number of spaces shown by his throw.

### Requirements

Req-1. The game must support a command line user interface.
Req-2. The game must support both human players and computer players.
Req-3. At each step of a human player, the game must ask for a command from the player. Supported commands must include: **continue**, **report**, **auto**, and **retire**.
Req-4. Upon receiving the command **continue**, the game must let the player take his/her turn.
Req-5. Upon receiving the command **report**, the game must print out the information of each square on the board and each player's location on the board.
Req-6. Upon receiving the command **auto**, the game must use a computer player to takes over the player's place. A computer player always **continue**s until the end of the game and makes decisions in Req-8 randomly.
Req-7. Upon receiving the command **retire**, the game must make the player retire (i.e., the player leaves the game with all his/her property becoming unowned).
Req-8. When a human player can decide whether to buy a property or to pay the fine to get out of Jail, the game must ask for the player's input.

Other details of the game can be decided based on the best judgement of each team.

### Bonus Features

- Graphical user interface.
- Online game.

## 7 WHAT TO HAND IN

Hand in the code of your classes, the tests, and the final report.

**If you use other IDEs for your development, make sure all your classes and tests are put into an IntelliJ IDEA project that is ready to be tested and executed.**

**Failing to do this or submitting code with compilation errors will lead to severe penalty (deduction of 50 points)!!**

# 8 PROJECT REPORT TEMPLATE

## Project Report

Group XYZ
COMP2021 Object Oriented Programming Fall 2016

Name1
Name2
Date

1. Introduction

*This document describes the design and implementation of the Monopoly game of group XYZ. The project is part of the course COMP2021 Object-Oriented Programming at PolyU. The following sections describe the requirements that were implemented and the design decisions taken. The last section describes the available commands in the game.*

2. The Monopoly Game

*Give a short description of what Monopoly is.*

2.1 Requirements

*Describe in this part which requirements (and possibly bonus requirements) you have implemented. Give a short description (1-2 sentences) of each requirement. List the software elements (classes and or functions) that are mainly involved in implementing each requirement.*

*Req-x: Description*

*Software elements*

2.2 Design

*Give an overview of the design of the game and describe in general terms how the implementation works. You can mention design patterns used, class diagrams, or anything else that helps understand the implementation.*

3. Quick Start Guide

*Describe here all the commands available.*