CSCI3180 – Principles of Programming Languages – Spring 2014

Assignment 2 — Your First Date with Ruby

Deadline: Mar 9, 2014 (Sunday) 23:59

1 Introduction

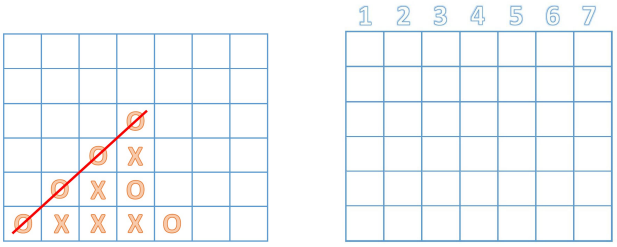
The purpose of this assignment is to offer you a first experience of Ruby, which is a pure object-oriented programming language. The main focuses are Dynamic Typing and Duck Typing. You will meet other special and important features of Ruby in a subsequent assignment.

The assignment consists of two parts. First, you need to implement the rule of a famous game called "Connect 4" in Ruby. Also, you need to design a simple computer player of the game which can at least generate valid moves. Of course, you can earn bonus points by implementing sophisticated AI. Second, a Ruby implementation of a University Administration System is given. You need to understand the Ruby program and re-implement it in Java. In the process, you will learn to appreciate the flexibility of Duck Typing.

2 Task 1: Connect-4

This is a small programming exercise for you to get familiar with Ruby, which is a dynamically typed language. In this task, you have to *strictly follow* our proposed OO design and the game rules for "Connect-4" stated in this section.

Figure 1: Connect-4



(a) Player O wins the game (b) Index of each column

Figure 2: Connect-4 Game Board

2.1 Description

The Connect-4 (Figure 1) game is a two-player game in which each player is associated with a symbol to stand for his/her discs and then takes turns dropping the discs from the top into one of the seven columns, each with six rows. Discs stack on top of others. In Figure 2(a), we mark the two types of discs by "O" and "X" respectively. The pieces fall straight down, occupying the next available space within a column. The objective of the game is to connect four of one's own discs of the same symbol vertically, horizontally, or diagonally before your opponent. If no one win the game until all cells are filled, the game ends in a tie. Figure 2(b) shows a game won by Player O.

A column is *unfilled* if it contains less than 6 discs. A player's move is *valid* if the specified column exists and is unfilled.

2.2 Types of Players

The two players, Player X and Player O, can be controlled by either human or computer. Users can choose the types (human or computer) of Player X and Player O before the game starts. Player O starts first.

Human-Controlled Player

A human-controlled player needs a human user to provide instruction for each move. In each turn, the symbol of the human-controlled player will be put in an unfilled column as specified by the human player. The input is an integer i from 1 to 7 (as shown in Figure 2(b)).

Computer-Controlled Player

A computer-controlled player just randomly chooses an unfilled column to make a valid move.

2.3 Input/Output Specification

The input/output specification is detailed in this section.

2.3.1 Input Specification

In this exercise, you are required to use command line to get input information. Since this is a simple program, there are just two operations requiring user-input.

Types of Players

Users can choose the types of Player X and Player O to be either human- or computer-controlled. You have to use command line to request for the types of the two players before starting the game (as shown in Figure 3).

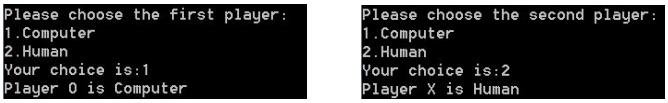


Figure 3: Players' types

Human-controlled Player's Move

Human-controlled players require user-inputs to determine their next moves. When it is human-controlled player's turn during the game, your program should request for the column of the next move. The input should be an integer from 1 to 7. Any invalid input or moves should be forbidden and the user is requested for input again until a valid move is given.

2.3.2 Output Specification

All the output should be printed to the command line window. Whenever there are changes made in the game board, your program should print out the updated game board. When the game is finished, a message that indicates the result of the game should be printed. The winning connected four pieces should also be highlighted in reverse video. If there are more than one "connect-4", you can show either one of them or all of them.

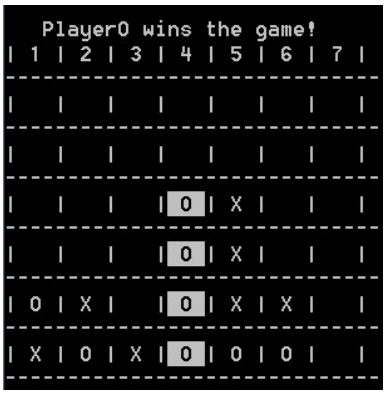


Figure 4: Output shown in the command line window

2.4 Ruby Classes

Please follow the classes `Connect_Four`, `Player`, `Human` and `Computer` defined below in your Ruby implementation. You are free to add other variables, methods or classes. You should run your program using the statement: `Connect_Four.new.startGame.`

1. **Class** `Connect_Four`
   A game object which holds the game board, coordinates players' turns and controls game logics. You have to implement it with the following components:

   - **Instance Variable(s)**

`@gameBoard`
- This is a two-dimensional array representing the game board.

`@player1`
- This is a variable representing Player O.

`@player2`
- This is a variable representing Player X.

`@turn`
- This is a variable indicating the player that should play in the current turn.

- **Instance Method(s)**

  `initialize`
  - Initialize the Connect_Four game.

  `startGame`
  - Start a new game and play until winning/losing or draw.

  `printGameBoard`
  - Print out the game board in the command line window.

2. **Class** `Player`
   An abstract super class representing a player object. You have to implement it with the following components:

   - **Instance Variable(s)**

     `@playerSymbol`
     - This is a variable indicating the symbol of the player (X or O).

   - **Instance Method(s)**

     `initialize: playerSymbol`
     - Initialize the player with its symbol X or O.

     `nextColumn`
     - An abstract method to be implemented in subclasses, which returns the column of next move.

3. **Classes** `Human` and `Computer`
   Classes extending the super class `Player` to represent a human- and a computer-controlled player object respectively.

2.5 Bonus

Random moves are so naive. We encourage you to make a more clever AI. If your AI has a 50% odds against the original AI, you will earn an extra 15 points on this assignment. If your AI is the most brilliant one in the class, you will earn an extra 30 points. We will arrange for an automated tournament environment to test your implementations. Please indicate in your assignment submission if you want to join the tournament.

2.5.1 Tournament

In order to make tournament more convenient, students who will join this tournament should *strictly follow* the standard of input and output below.

The AI should be built in the instance method: `nextColumn`

1. **Input of the method**
   A two-dimensional array: `gameBoard`.
2. **Output of the method**
   The index of the column: `column (1 to 7)`
   (Invalid output will get penalty points.)

3 Task 2: University Adminstration System

This task is to implement a University Administration System in Java. A Ruby program of the system is given. You need to understand its behavior and re-implement it with Java. After finishing this task, you will have experienced a special feature of Ruby called Duck Typing. And you will see a difference between Ruby and Java, the latter of which does not support Duck Typing.

3.1 Duck Typing

The following synopsis of Duck Typing is summarized from:

```
http://en.wikipedia.org/wiki/Duck_typing
http://www.sitepoint.com/making-ruby-quack-why-we-love-duck-typing
```

In standard statically-typed object-oriented languages, objects' classes (which determine an object characteristics and behavior) are essentially interpreted as the objects' types. Duck Typing is a new typing paradigm in dynamically-typed (late binding) languages that allows us to dissociate typing from objects's characteristics. It can be summarized by the following motto by the late poet James Whitcombe Riley:

> When I see a bird that walks like a duck and swims like a duck and quacks like a duck,
> I call that bird a duck.

The basic premise of Duck Typing is simple. If an entity looks, walks, and quacks like a duck, for all intents and purposes it is fair to assume that one is dealing with a member of the species anas platyrhynchos. In practical Ruby terms, this means that it is possible to try calling any method on any object, regardless of its class.

An important advantage of Duck Typing is that we can enjoy polymorphism without inheritance. An immediate consequence is that we can write more generic codes, which are cleaner and more precise.

3.2 Background

After the successful and brilliant CUSIS system, CUHK is considering to build an advanced information system to take care of various administrative needs. The implementation of the Chinese University Administrative System (CUAS) is delegated to Prof. Jimmy Lee of the Department of Computer Science and Engineering. It is a complicated system that consists of sub-systems which manage information of different units in CUHK. The information involves usage of different kinds of users which include students, staff and visitors.

The system is so complex that Jimmy has no idea on how to design it. Unfortunately, the deadline gets closer and closer. Fortunately, with the consent of AUHK, Jimmy's friend at AUHK offers him to use the source code of a similar system which is implemented in Ruby and used at AUHK. Unfortunately, server of CUHK supports only Java. You are Jimmy's excellent students who are willing to help Jimmy "plagiarize" and replicate the system in Java.

3.3 Task Description

The system you need to "plagiarize" and re-implement is a simplified version of CUAS. It involves information of units such as libraries and canteens, and users who are professors, students and visitors. Users will have different characteristics and behaviors in different university units. For

instance, professors and students can borrow and return books from the libraries. Departments pay salary to professors. Students receive scholarship from their departments. Professors, students and visitors can buy lunches at the canteen. Professors can eat and drink in the staff club.

Task Requirement

The requirement is simple. Please replicate all the behavior of the given Ruby Program using Java. All features of Java except "reflection" can be used. You will be also evaluated on your programming style.

4 Report

Your simple report should answer the following questions within TWO A4 page:

1. Using your codes for Task 1 and Task 2, discuss features of Ruby that are different from other common object-oriented programming languages.
2. Using codes for Task 2, compare polymorphism obtained with Duck Typing and with inheritance.
3. Using codes for Task 2, give two scenarios in which the Ruby implementation is better than the Java implementation. Given reasons.

5 Submission Guidelines

Please read the guidelines CAREFULLY. If you fail to meet the deadline because of submission problem on your side, marks will still be deducted. So please start your work early!

1. In the following, **SUPPOSE**

   your name is *Chan Tai Man*,
   your student ID is *1155234567*,
   your username is *tmchan*, and
   your email address is *tmchan@cse.cuhk.edu.hk*.

2. In your source files, insert the following header. REMEMBER to insert the header according to the comment syntaxes of Ruby and Java.

```
/*
 * CSCI3180 Principles of Programming Languages
 *
 * --- Declaration ---
 *
 * I declare that the assignment here submitted is original except for source
 * material explicitly acknowledged.  I also acknowledge that I am aware of
 * University policy and regulations on honesty in academic work, and of the
 * disciplinary guidelines and procedures applicable to breaches of such policy
 * and regulations, as contained in the website
 * http://www.cuhk.edu.hk/policy/academichonesty/
 *
 * Assignment 2
 * Name : Chan Tai Man
 * Student ID : 1155234567
 * Email Addr : tmchan@cse.cuhk.edu.hk
 */
```

The sample file header is available at

```
http://www.cse.cuhk.edu.hk/~csci3180/resource/header.txt
```

3. `Gzip` the `tarred` file to `username.tar.gz` by

```
gzip tmchan.tar
```

4. `Uuencode` the `gzipped` file and send it to the course account with the email title "HW2 *studentID yourName*" by

```
uuencode tmchan.tar.gz tmchan.tar.gz \
| mailx -s "HW2 1155234567 Chan Tai Man" csci3180@cse.cuhk.edu.hk
```

5. Please submit your assignment using your Unix accounts.

6. An acknowledgement email will be sent to you if your assignment is received. **DO NOT** delete or modify the acknowledgement email. You should contact your TAs for help if you do not receive the acknowledgement email within 5 minutes after your submission. **DO NOT** re-submit just because you do not receive the acknowledgement email.

7. You can check your submission status at

   `http://www.cse.cuhk.edu.hk/~csci3180/submit/hw2.html`.

8. You can re-submit your assignment, but we will only grade the latest submission.

9. Enjoy your work :>