CSCI3180 – Principles of Programming Languages – Spring 2018

Assignment 2 — Your First Date with Python

Designed by: Zhizhen Ye

Deadline: Mar 11, 2018 (Sunday) 23:59

1 Introduction

The purpose of this assignment is to offer you a first experience with Python, which supports the object-oriented programming paradigm. Our main focuses are Dynamic Typing and Duck Typing. Please use **Python 2.7** to finish this assignment.

The assignment consists of 4 tasks.

1. You need to implement the rules of a famous game called Gomoku in Python.
2. You are asked to demonstrate the advantages/disadvantages of dynamic typing through some example code.
3. We give you the JAVA source code of a game called "survival game". You need to re-implement the game in Python.
4. Additional features and mechanisms are introduced to the "survival game". You need to implement the enhanced game in both JAVA and Python.

After completing the 4 tasks, you need to write a report elaborating on dynamic typing and duck typing.

IMPORTANT: all your codes will be graded on the Linux machines in the Department. You are welcome to develop your codes on any platform, but please remember to test them on Department's machines.

2 Task 1: Gomoku

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

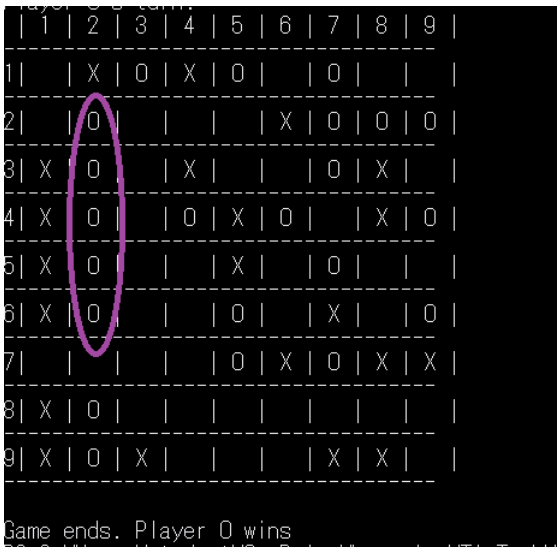

Figure 1: Gomoku

2.1 Description

Gomoku is a two-player game on a 9×9 gameboard. Each player is associated with a symbol, 'O' or 'X'. Players place their respective symbols into empty cells of the board in turn. The objective of a player is to connect five consecutive symbols of his own vertically, horizontally, or diagonally before the opponent does. If no one win the game when all cells are filled, the game ends in a tie.

In Figure 1, for example, five cells surrounded by the oval are connected. Player O wins the game.

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 always 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 cell as specified by the human player. The input is a pair of integers, each ranging from 1 to 9.

Computer-Controlled Player

A computer-controlled player just randomly chooses an unfilled cell 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.

Type 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. See an example screen shot in Figure 2.

```
Please choose player 1 (O):
1. Human
2. Computer Player
Your choice is: 1
Player O is Human.

Please choose player 2 (X):
1. Human
2. Computer Player
Your choice is: 2
Player X is Computer.
```

Figure 2: Players' types

Human-controlled Player's Moves

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 position of the next move. The input should be a pair of integers, each from 1 to 9, separated by a single space. Any

invalid inputs or moves should be forbidden and the user is requested for input again until a valid move is given. See Figure 3 for an example.

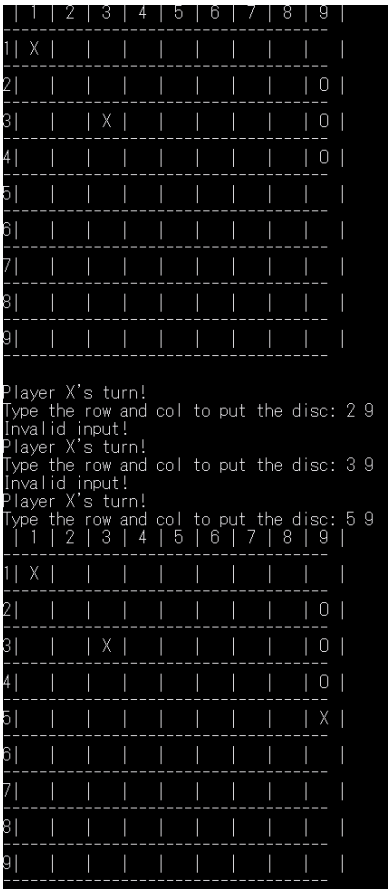

Figure 3: Example gameplay output shown in the command line window

2.4 Python Classes

Please follow the classes `Gomoku, Player, Human` and `Computer` defined below in your Python implementation. You are free to add other variables, methods or classes. We would start the program by running the command: `python Gomoku.py`

1. **Class** `Gomoku`

   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 9×9 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 referring to the player that should play in the current turn.

   - **Instance Method(s)**

     `__init__(self)`

- Create two players and initialize other instance variables.

`createPlayer(self,symbol,playerNum)`

- Create a player (human- or computer-controlled) with corresponding symbol ('O' or 'X') and player number (1 or 2) by prompting the user.

`startGame(self)`

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

`printGameBoard(self)`

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

`checkWin(self)`

- Check if any player has won the game.

`checkTie(self)`

- Check if the game is ending in a tie.

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').

     `gameBoard`

     - The gameboard the player is in.

   - **Constructor and Instance Method(s)**

     `__init__(self, symbol, gameboard):`

     - Initialize the player with its symbol 'X' or 'O' and corresponding gameboard.

     `nextMove(self)`

     - An abstract method to be implemented in subclasses, which returns the row and column (1-9 for each) of the next move.

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

3 Task 2: Demonstrating Advantages of Dynamic Typing

There are commonly-claimed advantages of Dynamic Typing:

1. More generic code can be written. In other words, functions can be applied on arguments of different types.
2. Possibilities of mixed type collection data structures.

Please provide concise example code to demonstrate the advantages of Dynamic Typing mentioned above. You are welcome to provide other advantages and disadvantages along with code segment for extra bonus points.

Dynamic Typing makes coding more flexible and convenient, but type checking can only be carried out at runtime, incurring time overhead and reliability issues.

4 Task 3: Survival Game

This task is to implement a survival game in Python. A Java program of the game is given. You need to understand its behavior and re-implement it in Python. Duck typing helps to simplify your codes. Use duck typing whenever possible. And you will see a difference between Java and Python, the former of which does not support Duck Typing.

4.1 Duck Typing

The following synopsis of Duck Typing is summarized from:

`http://en.wikipedia.org/wiki/Ducktyping`
`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' 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.

4.2 Background

There are $N = 2k$ residents in the world of Kafustrok with two races living there: Human and Chark. The races have equal population. Different races would have different attributes **health cap, weapon, mobility** available to them.

The world is currently in chaos. There are places filled with obstacles one cannot move to. The landscape varies and players are teleported from time to time. During the teleportation, the weapons of each player would get enhanced according to its type.

The chaos has eroded residents' sanity. Each player is enraged and wants to eliminate all other players, even players of the same race and probably also oneself. The game continues until only one player is left alive in the world of Kafustrok.

4.3 Task Description

You should read and execute the given Java program to understand its behavior and design. Please replicate all the behavior of the given Java Program in Python with Duck Typing, following the same class design. You should not introduce extra instance variables or instance methods. You program should run by calling `python survival_game.py`. You will also be evaluated on your programming style.

5 Task 4: For Glory!

Due to the effort of your implementation and thousands of game playing, the world has evolved and its order restored.

In the new civilization, residents gradually realize the importance of healing in such a dangerous world. A new kind of equipment called *Wand* is then developed for healing players of the same race, **including the owner himself**.

As residents are now more civilized, cannibalism is considered to be evil. You are not allowed to attack player of your own race. You are also not allowed to heal your opponent race.

The game ends when players of **one race** are all eliminated.

5.1 Enhanced Features

When a race is created, the *last* player of the race holds a wand instead of a weapon.

Wand is a new type of equipment and not a Weapon. The same instance variables and methods of the `Weapon` class are subsets of those of the `Wand` class, but you are not required to implement the extra ones. When a wand acts on another player of the same race, the wand exerts a healing effect on the player. A wand's initial healing effect is 10. Every time a wand gets enhanced, the effect increases by 5 points. A wand has range 5. When it acts on a player with the same race as its owner, that player's health increases by **"effect"** points, but cannot exceed the health cap.

5.2 Task Description

You are now required to modify/extend both the Java and Python implementation of Task 3. You will also be evaluated on your programming style.

6 Report

Your report should answer the following questions within TWO A4 pages.

1. Provide example code and necessary elaborations for demonstrating advantages of Dynamic Typing as specified in Task 2.
2. Using codes for Task 3, give two scenarios in which the Python implementation is better than the Java implementation. Given reasons.
3. Using codes for Task 4, illustrate further advantages of Dynamic Typing and Duck Typing.

7 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. Your submissions will be accepted latest by 11:59p.m. on Mar 13, but submissions made after the original deadline would be considered as **LATE** submissions and penalties will be imposed in the following manner:
   - Late submissions before 11:59p.m. on Mar 12: marks will be deducted by 20%.
   - Late submissions before 11:59p.m. on Mar 13: marks will be deducted by 50%.
2. 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*.
3. In your source files, insert the following header. REMEMBER to insert the header according to the comment rule of Python.

```
/*
 * 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://course.cse.cuhk.edu.hk/~csci3180/resource/header.txt`

4. *Task 2 report and the final report should be merged into one report file* and submitted to VeriGuide, which will generate a submission receipt. The report should be named "report.pdf". The VeriGuide receipt of report should be named "receipt.pdf". The report and receipt should be submitted together with codes the same ZIP archive.

5. `Tar` your source files to `username.tar` by

```
tar cvf tmchan.tar Gomoku.py task3_python.zip task4_java.zip \
task4_python.zip report.pdf receipt.pdf
```

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

```
gzip tmchan.tar
```

7. `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
```

8. Please submit your assignment using your Unix accounts.

9. 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.

10. You can check your submission status at

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

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

12. Enjoy your work :>