

# 21

## *With Primes*

### Introduction

This document specifies the programming component of Assignment 1. This component is due by 11pm Saturday of Week 7 (14<sup>th</sup> September, 2019). **Heavy penalties will apply for late submission.** This is an individual assessment task and must be your own work. You must attribute the source of any part of your code which you have not written yourself. However, you must ensure to be able to understand, explain, and/or modify any part of your program. Please note the section on plagiarism in this document.

**The assignment must be done using the BlueJ environment.**

The Java source code for this assignment must be implemented according to the **Java Coding Standards for this unit.**

Any points needing clarification may be discussed with your tutor in the lab classes.

Completion of this assignment contributes towards the following FIT9131 learning outcomes:

1. *design, construct, test and document small computer programs using Java;*
2. *interpret and demonstrate software engineering principles of maintainability, readability, and modularisation;*
3. *explain and apply the concepts of the "object-oriented" style of programming.*

### Specification


For this assignment you will write a program which will allow a player to play a game *21 With Primes* against the computer. This section specifies the required functionality of the program. **Only a text interface is required for this program;** however, more marks will be gained for a game that is easy to follow with clear information and error messages to the player.

The aim of *21 With Primes* is for a player and the computer to compete against each other to gather the most points while playing a single tile each turn with the goal of ensuring that the total for each round is less than or equal to 21.

To play the game, each player (human and computer) are given a set of 5 tiles. Each tile has an value and an associated score. For playing a tile, the player gets the associated score for that tile provided the total is less than 21.

**Game play**

The *21 With Primes* game begins with a menu displayed to the player as follows:

 BlueJ: Terminal Window - 21 With Primes

**Options**

```
Please select from the following options
Press 1 to register a player
Press 2 to start a new game
Press 3 to view a help menu
Press 4 to exit
```

This menu must repeat until the user decides to exit the game by selecting the appropriate option.

**Option 1:**

Option 1 allows a player to register themselves before playing the game. The player is asked for their name only. The player's name must be between 3 and 10 characters inclusive.

**Option 2:**

Option 2 allows a player to play the game against the computer. Option 2 can only be selected if Option 1 has been selected before. Option 2 will prompt the user for the number of rounds the player wishes to play. The number of rounds playable must be greater than 0 and less than 10.

**Option 3:**

Option 3 will display a descriptive help menu to the user which clearly outlines the rules of the game and how the game should be played. It should allow any user to be able to understand and use the game without any assistance.

**Option 4:**

Option 4 will end the game and allow the user to exit the program.

## Game Rules

The following are the game rules:

1. Each player starts the game with zero points.
2. Prior to each round, each player is given the following set of five tiles. Each tile has a value and an associated score:

**Table 1 Points awarded for different tiles in the game**

Tile Value	Score
1	5
2	4
3	3
5	2
7	1

3. No other tile value can be played other than the above primes<sup>1</sup>. This must be validated when the user selects a tile as well. An error message should be displayed for an incorrect selection and the user can reselect.
4. Each player has the same set of 5 tiles. Each player can use each of their 5 files only ONCE in each round.
5. Before each round, the game will automatically decide if the human player goes first or the computer player goes first.
6. For each round, each player will play ONE tile, with the tile value adding to the game total for that round. Provided the game total is less than or equal to 21, the player will get the score for using that tile. However, if the game total is greater than 21, no score is allocated to the player who played the last tile causing the score to become greater than 21. As per the following example:

Round 1 both players have 5 tiles each {1, 2, 3, 5, 7}, {1, 2, 3, 5, 7}

Human player plays tile 3. Game Total is now 3. Human player score is 3.

Computer player plays tile 1. Game Total is now 4. Computer player score is 5.

Round 2 both players have 4 tiles each {1, 2, 5, 7}, {2, 3, 5, 7}

Human player plays tile 5. Game Total is now 9. Human player score is 5.

Computer player plays tile 7. Game Total is now 16. Computer player score is 6.

---

<sup>1</sup> A prime number is a number which can only be divided by 1 and itself.

Round 3 both players have 3 tiles each {1, 2, 7}, {2, 3, 5}

Human player plays tile 1. Game Total is now 17. Human player score is 10.

Computer player plays tile 2. Game Total is now 19. Computer player score is 10.

Round 4 both players have 2 tiles each {2, 7}, {3, 5}

Human player plays tile 2. Game Total is now 21. Human player score is 14.

Round Over

Final Score

Human Player is  $14 + 5 - 0 = 19$

Computer Player is  $10 + 0 - 3 = 7$  (Since computer player did not use the tile with value 5)

7. Once the round ends, the following are the rules for scoring:
  - Each player will get their score based on the total of the tiles they have used during the round.
  - Any player who has NOT used the tile with the value of **5**, will get a penalty of -3 points.
  - The player who, after all deductions, has the highest score, will be the winner of that round and will get 5 points.
8. At the end of all the rounds, the player who has won the most rounds is declared the winner of the game.

**Game Requirements**

- Players must use the 5 tiles given with the associated values and scores.
- Players cannot pass in any given round.
- You must keep in mind that just staying under the total of 21 is not enough to win the game.
- The players **MUST** choose their tiles carefully in order to prevent the opponent from getting to 21 or to deliberately end the round if they already have a higher score for that round.
- The computer player **MUST** demonstrate some kind of logical reasoning behind the tile selection. The program must demonstrate some kind of simple algorithm which is used by the computer in order to determine which tile it must play during the game.
- Remember, the objective of the game is not just to reach the total of 21. Since the players are playing together, it is a combined score. Second, the game is won by the player who has won the most rounds. So if a player has a higher score, they may, sometimes, 'throw' the round by deliberately going over the total of 21 so the round ends and the other player cannot play any more high value tiles. Keep this in mind when trying to think of how the game should be played.

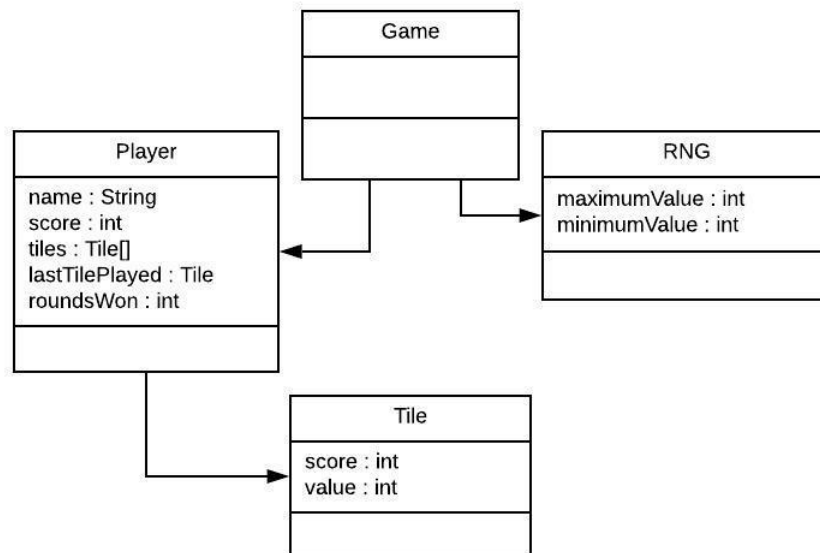
**Hints and Suggestions**

- You are encouraged to think of how the game can be played from the computer player's perspective as it requires you to apply some kind of logical reasoning.
- When working with multiple classes you are encouraged to understand and apply pass by reference so as to ensure appropriate design is included within your program.
- For assignment 1, you are not required to validate numeric inputs with the exception of the menu choice. All other numeric inputs are not required to be validated.

## Program design

Your program must consist of at least four classes: *Player*, *Game*, *Tile*, and *RNG*. A suggested class design is shown in Figure 1. There are suggested fields for each class. If you include any extra fields then you must be able to justify these. You may want to include comments in your program to state any assumption made.

The remainder of this section gives details of these classes.



**Figure 1 Class diagram for the 21 with Primes game**

### Player class

The Player class will specify the attributes and behaviours of a player. An object of the Player class will have the following fields:

*name* – the name of the player.

*score* – the score of the player for the current round

*tiles* – an array of tile objects which are used to play the game.

*lastTilePlayed* – a tile object representing the last tile which was played by the player

*roundsWon* – the number of rounds won by the player during the game.

The class must also have a default constructor and a non-default constructor.

The Player class should also have *appropriate* accessor and mutator methods for its fields. You should not allow an object of class Player to be set to an invalid state. There should be no input from the terminal or output to the screen. A Player object should also be able to return its state in the form of a String.

**RNG class**

An object of the RNG class will generate a random number from a minimum value to a maximum value. It will have two fields:

*maximumValue* – maximum value of the random number

*minimumValue* – minimum value of the random number

**Tile class**

The tile class will store each individual tile which can be used by the player to play the game. An object of the Tile class will have the following fields:

*score* - the score gained by the player from playing that tile during a round.

*value* - the face value of the tile which is used to calculate the round total while playing the game.

**Game class**

The Game class will be in the same BlueJ project that contains your Player class. The Game class will manage the playing of a game. It will have no fields.

The Game class will have methods to manage the playing of the game. You must demonstrate your understanding of the software engineering principles of abstraction and modularization by ensuring that each method does a single task only.

## Assessment

### Programming Task Assessment (10%)

Assessment for this component will be done via an interview with your tutor.

The marks for your program code will be allocated as follows:

- **40%** - Object-oriented design quality. This will be assessed on appropriate implementation of classes, fields, constructors, methods and validation of the object's state.
- **10%** - Adherence to FIT9131 Java coding standards.
- **50%** - Program functionality in accordance to the requirements.

You must submit your work by the submission deadline on the due date (**a late penalty of 20% per day**, inclusive of weekends, of the possible marks will apply - up to a maximum of 100%). **There will be no extensions** - so start working on it early.

Marks will be deducted for untidy/incomplete submissions, and non-conformances to the FIT9131 Java Coding Standards.

Please note that your program code will be submitted to a code similarity checker.

### Interview

You will be asked to demonstrate your program at an “interview” following the submission date. At the interview, you will be asked to explain your code/design, modify your code, and discuss your design decisions and alternatives. **Marks will not be awarded for any section of code/design/functionality that you cannot explain satisfactorily** (the marker may also delete excessive in-code comments before you are asked to explain that code).

In other words, **you will be assessed on your understanding of the code**, and not on the actual code itself.

Interview times will be arranged in the tutorial labs in Week 7. It is your responsibility to attend the lab and arrange an interview time with your tutor. **Any student who does not attend an interview will receive a mark of 0 for the assignment.**



## Submission Requirements

The assignment must be uploaded to Moodle by 11pm Saturday of Week 7 (14<sup>th</sup> September, 2019).

The submission requirements for Assignment 1 are as follows:

A .zip file uploaded to Moodle containing the following components:

- the BlueJ project you created to implement your assignment.
- a completed **Assignment Cover Sheet**. This will be available for download from the unit's Moodle site before the submission deadline. You simply complete the editable sections of the document, save it, and include it in your .zip file for submission.

The .zip file should be named with your Student ID Number. For example, if your id is 12345678, then the file should be named 12345678\_A1.zip. **Do not name your zip file with any other name.**

It is your responsibility to check that your ZIP file contains all the correct files, and is not corrupted, before you submit it. If you tutor cannot open your zip file, or if it does not contain the correct files, you will not be assessed.

Marks will be deducted for any of these requirements that are not complied with.

Warning: there will be no extensions to the due date. Any late submission will incur the 20% per day penalty. It is strongly suggested that you submit the assignment well before the deadline, in case there are some unexpected complications on the day (e.g. interruptions to your home internet connection).

## Extensions

All requests for extensions must follow the faculty guidelines and submit the required forms as soon as possible. In addition, when emailing for an extension, please remember to include all code completed until that time. This assignment cannot be completed in a few days and requires students to apply what we learn each week as we move closer to the submission date.

## Plagiarism

Cheating and plagiarism are viewed as serious offences. In cases where cheating has been confirmed, students have been severely penalised, with penalties ranging from losing all marks for an assignment, to facing disciplinary action at the Faculty level. Monash has several policies in relation to these offences and it is your responsibility to acquaint yourself with these.

Plagiarism (<http://www.policy.monash.edu/policy-bank/academic/education/conduct/plagiarism-policy.html>)