FIT9131 Assignment A, S1 2023 - Dart-ificial Intelligence

FIT9131 Assignment A: Dart-ificial Intelligence

Introduction

This assignment is due by **11:55pm AEST on Friday of Week 7 (21 April, 2023)**. In addition there is a requirement that you show your work to your tutor in your Applied class in weeks 5 and 6, and attend an interview with your tutor following the submission of your assignment.

The assignment is worth 15% of the marks for your final assessment in this unit. **Heavy penalties will apply for late submission.** This is an **individual** assignment and must be your own work. You must attribute the source of any part of your code which you have not written yourself. Please note the section on plagiarism in this document.

You must not use generative artificial intelligence (AI) to generate any materials or content in relation to any assessment task in this unit.

The assignment must be done using the Assignment A workspace environment on the Ed platform (opposite).

The Java source code for this assignment must be implemented according to the **FIT9131 Coding Standards** (see Lesson 3.4).

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

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

1. Design and construct Java programs according to standard object-oriented principles

2. Apply and demonstrate debugging processes to Java applications

3. Develop strategies for efficient and effective program testing

4. Document code according to specific programming standards

Specification

For this assignment, you will write a program for two players to play a *Dart Game.* In this game there will be one human player. The computer, *Dart Vader*, will be the other player. This section specifies

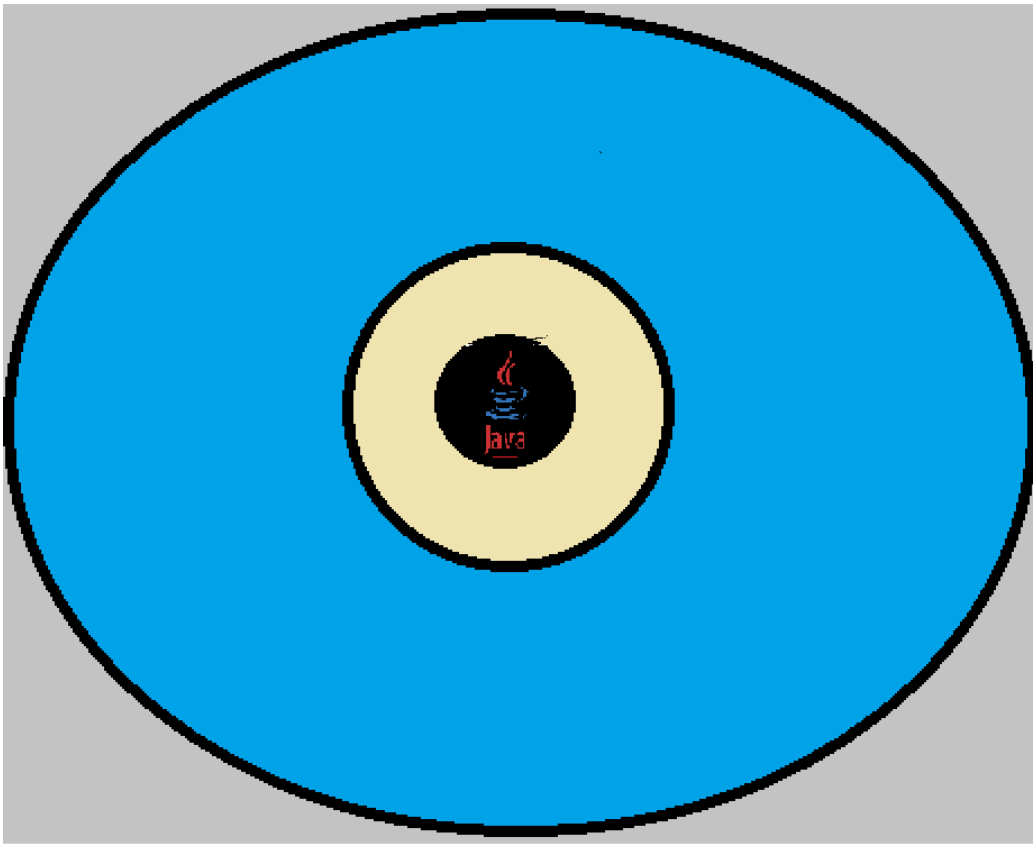
the required functionality of the program. **Only a text interface is required for this program**; however, more marks will be gained for a game that is easy to follow with clear information/error messages given to the player.

The aim of *Dart Game* is for the players to take turns to "throw" a number of darts at a dart board, and the winner is the player who gains the maximum score.

At the beginning of the game each player is given the same number of darts. The players take turns to throw the darts at the board. After each throw the player is shown their score for that turn, and their cumulative score for the game.

When all the darts have been thrown the winner is displayed on the screen.

The dart board consists of three concentric circles. These sit inside a square as shown below. The percentage of darts that land on the dart board is given by the ratio of the area of the circle to the area of the square, which is pi/4 times 100. The player only scores points if the dart lands on the board, i.e. within a circle. More points are gained if the dart lands inside the inner circles.



Specific rules of the game

The *Dart Game* begins with a message inviting the human player to enter their name. The name can contain only lowercase alphabetic characters and must be no more than 8 characters in length. If the player enters an invalid name then a warning message is displayed and the player is invited to enter another name.

The human player is then invited to choose the number of darts that will be thrown in the game. The number of darts must be from 1-5 inclusive. If the player enters an invalid value then a warning message is displayed and the player is invited to enter another number.

Playing the game

The human player is invited to throw the first dart. The game then progresses, with Dart Vader (the computer) throwing the next dart, then the players taking turns until all the darts have been thrown. Note that your program will generate a dart throw for each of Dart Vader's turns.

The following are the possible outcomes when a dart is thrown:

- the dart will land on the board (pi/4*100) percent of the throws. If the dart does not land on the board (i.e., it lands on the grey area instead) then 0 (zero) is scored.
- if the dart lands on the board then:
    - there is a 4% chance the dart will land in the black area. The player scores 5 points.
    - there is a 16% chance it will land in the yellow area. The player scores 2 points.
    - there is a 80% chance it will land in the blue area. The player scores 1 point.

When invited to throw a dart the human player may instead decide to abandon the game. Dart Vader will decide to abandon the game randomly on 5% of the throws.

The end of the game

When all the darts have been thrown the the game score for both players is displayed.

The player is then invited to play another game.

Program design

Your program should consist of at least three classes: Player, DartGame, and DartThrow. The following two sections give details of these classes.

Player class

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

Name – the name of the player

Score – the cumulative game score

The data type of each field must be chosen carefully and you must be able to justify the choice of the data type of the fields. You may want to include comments in the class to state any assumptions made. The class must also have a default constructor and a non-default constructor that accepts a

value for the name of the player.

The Player class should also have *appropriate* accessor and mutator methods for its fields. Validation of values for fields should also be implemented. 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 via methods from the Player class. A Player object should also be able to return its state in the form of a String.

DartGame class

The DartGame class will manage the playing of a *Dart Game*. It will have the following fields (at least):

Player - an object of type Player

The DartGame class will have methods to manage the playing of the game. These should include the **main()** method to start the program and methods for the following behaviours (at least):

- Display a welcome message on the screen.
- Request the player to enter their name.
- Request the player to enter the number of darts they wish to throw for the game.
- Calculate the score for a dart throw.
- Display the result of a dart throw.
- Display the result for the end the game.

DartThrow class

An object of the DartThrow class will generate a random number from 1 to a maximum value specified.

Assessment

Assessment for this assignment will be done via an interview with your tutor. The marks will be allocated as follows:

- 25% - Understanding and quality of object-oriented design. This will be assessed on the appropriate implementation of classes, fields, constructors, methods and validation of the object's state.
- 10% - Adherence to FIT9131 Java coding standards.
- 10% - Progress of code development, as shown via Ed workspace environment. Your tutor will assess your work during your applied session in weeks 5 and 6.
    - 5% in week 5 for Class Diagram (this can be hand-drawn)
    - 5% in week 6 for **Player** and **DartThrow** classes
- 55% - Understanding of program functionality and correctness in accordance to the

requirements.

You must use the workspace environment in the Ed platform to code all parts of your program. You must not copy paste huge chunks of code from somewhere else.

You must ensure that your program code meets the coding standard requirements outlined in the unit.

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

You must submit your work by the submission deadline on the due date (a late penalty of 10% per day, inclusive of weekends, of the possible marks will apply). There will be no extensions - so start working on it early.

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, your program 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.

- For **on-campus students**, interview times will be arranged in the applied classes in Week 7 and will take place on campus.
- For **online students**, the interviews will be organised in the applied classes in Week 7 and will take place online via zoom.

It is your responsibility to make yourself available for an interview time. **Any student who does not attend an interview will receive a fail grade for the assignment.**

Submission Requirements

The assignment must be submitted by **11:55pm AEST on Friday of Week 7 (21 April, 2023)**.

The submission requirements for Assignment A are as follows:

- The main class in your program **MUST** be called **DartGame.java** and it should contain the **main()** method to start the program.
- Submit your work via the Ed platform, using the workspace area next to this assignment specification.
- Re-submissions are allowed before the submission deadline, and are encouraged. You should

submit your progress as you go. However, please ensure that you do not click on the submit button *after* the due date. The last submission will be used for grading purposes and a submission after the deadline will incur a late penalty.

- A signed Assignment Cover Sheet. [Note: You are required to download the Assignment Coversheet, sign the document and upload the pdf file in the Ed platform (you may drag and drop to the Toggle Pane)]

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 a 10% 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).

Plagiarism

Plagiarism and collusion are viewed as serious offences. All submitted code will be subjected to a similarity checker, and any submissions determined to be similar to a submission from a current or past student will be investigated further. The outcome of the decision pertaining to plagiarism and/or collusion will be determined by the faculty administration. To ensure compliance with this requirement, be sure to do all your coding in the Ed workspace environment and do not copy and paste any code into the workspace environment.

In cases where plagiarism or collusion has been confirmed, students have been severely penalised, 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.

Assessment and Academic Integrity Policies and Procedures (https://www.monash.edu/learning-teaching/priorities-and-programs/assessment-and-academic-integrity/assessments-and-integrity-policy-and-procedure)