Assignment 2 - Pacman got Scary

Introduction

Welcome to the second assignment for COMP90041 - Programming and Software Development!

For this project, we will extend the PACMAN and add some monsters here.

PACMAN can now be played in two modes -

- Single Player - only one player plays.
- Multi Player - two players play one by one.

There are new entities in the game now.

- Multiple special foods for super powers.
- Multiple Monster
- Monsters are of different colours - RED, BLUE, YELLOW, GREEN

There is a scoreboard that reflects scores per game.

Preamble: "The Specifications"

The slides from this lesson module for the assignment act as "the specifications" for the system we are about to build.

"The specifications" in real-life software development is the document that details the features that should be implemented in any particular project. They are the features and requirements that you, the *Software Developer*, and the client have agreed should be implemented in the system.

As such, you should read these specifications carefully and ensure that your program implements the requirements of the specification correctly.

Tests will be run on your program to check that you have implemented these specifications correctly. **Note** that for Assignment 2, we will provide 10 visible tests that you can run to check that the basic functionality of your system is correct. However, there will also be 5 *hidden (i.e., invisible)* tests that we will run when assessing the correctness of your code. So once your program passes the basic tests, *we strongly recommend that you perform further tests yourself*, to ensure that the required features have been implemented correctly.

How to read the specifications?

- Some of the specifications will be referred back from Assignment 1. This will be notified using a green strip as shown below -

Carried Forward: This section is referred as is from Assignment 1

- Other specifications can be referred from Assignment 1 with slight modifications. This will be notified using a yellow strip.

Modifications: This section is referred from Assignment 1 with slight modifications

- New additions to the assignment will be marked using blue callouts like below-

Addition: This specification is entirely new to Assignment 2.

- Lastly, sometimes a developer thinks exhaustively and engages in continuous discussion with people to gather more information. We try to keep certain use cases out of scope as they bring more complexity to the system. Some advanced developers can cope through it but we would like to keep some things simpler for beginners. Please read carefully through the **out-of-scope specifications** as well. This may be embedded as a red strip in the specifications.

Out of Scope Scenario: This scenario ..... is out of scope for this assignment

- Other than this, we will use the general informational, warning, error, and assumption callouts using blue, yellow, red, and green coloured callouts.
- Also, note that the code snippets have some characters in bold that represent the inputs to the program.

Students can assume that test cases only contain outputs that are explicitly part of specifications. The test cases, visible or hidden, do not produce any output that is not mentioned in the specifications explicitly. Please note that the specifications will give you warnings and important notes where we expect special input handling. You should observe those and implement them accordingly as they will be tested while marking your program.

Preamble: Intended Learning Outcomes

The Intended Learning Outcomes for this Assignment are mentioned below -

- Control Flows - use branching and looping to navigate the special use cases in specifications.
- Classes - identify the entities and encapsulate their data and actions together in a class.
- Static Variables & Methods - identify and implement a use case that can be solved using static variables and methods
- Arrays - to use 1D and 2D arrays and perform associated operations
- Enums - can create enums to define a particular variation type
- Packages - identify and group logical entities(classes) together
- Javadoc - generate javadocs from comments used in classes (all classes including the one in packages)

ArrayLists and other Collection types like Lists, HashMaps etc. are prohibited. In Assignment 2, we want to test your ability to work with array, array indices and adding and removing items from an array, resizing array. ArrayLists provide all these capabilities out of the box and hence are prohibited from use in this assignment. ArrayLists and other Collections types will be covered in the next assignment.

A warning to those who have previous programming experience in other programming languages (like C or Python) that follow a procedural programming paradigm: Be careful to develop your program using object-oriented principles, as programming and structuring your code in a way that is not object-oriented is an easy way to lose marks for structure in the assignments.

Preamble: Structure and Style

We will also be assessing your code for good structure and style.

Use methods when appropriate to simplify your code, and **avoid duplicate code.**

Use correct Java naming conventions for class names, variable names, and method names and put them in a well-organised way in your code i.e. it should be readable as well. Look at the conventions provided by Oracle here.

Code structure in a file is very important and **improves readability**. Look at the code organisation conventions provided by Oracle here.

Make sure the **names** you choose are **meaningful**, to improve the readability of your code.

Ensure to add meaningful comments in between your code and **Javadoc comments** for classes and methods.

We will provide a marking scheme to help guide you in the development of your program.

Academic Honesty

- All assessment items (assignments, tests, and exams) must be your own, individual, original work.
- Any code that is submitted for assessment will be automatically compared against other students' code and other code sources using sophisticated similarity-checking software.
- Cases of potential copying or submitting code that is not your own may lead to a formal academic misconduct hearing.
- Potential penalties can include getting zero for the project, failing the subject, or even expulsion from the university in extreme cases.
- For further information, please see the university's Academic Honesty and Plagiarism website, or ask your lecturer.

You can find the Academic Integrity Module in your Canvas here.

Game Rules

There are multiple entities in the game. These are described below.

- Maze
- PACMAN
- Special Food
- Monsters
- Game Loop
- ScoreBoard
- LocationGenerator

Maze

We have looked at what PACMAN and Maze look like in the previous assignments. More details can be found here.

PACMAN

PACMAN game starts with the initial position of PACMAN in row = 1 and column = 1. (Remember, counting in JAVA starts from 0). PACMAN now has 5 things to manage -

- number of times it hits the wall or boundaries
- number of moves it takes before the game ends
- number of special foods eaten
- number of monsters killed
- superpower to kill the monsters

Special Food

Special Food gives PACMAN superpowers to kill the monster. Once you have consumed the special food, the game doesn't stop there.

- When PACMAN consumes a special food, its superpower increases by 1.
- When PACMAN meets a monster if it has superpower then it can kill the monster and super power decreases by 1.
- If it doesn't have a superpower, the monster kills PACMAN and the game ends.
- In this case, there will be 4 special foods to kill 4 monsters.

Monsters

There are four kinds of Monsters. They are differentiated by their colour - RED, BLUE, GREEN, YELLOW. In the actual version of the game, monsters can move. **In this version of the game monster's locations are fixed. They do not move.**

The monsters are mentioned in the maze using the letters - R, B, G, Y for RED, BLUE, GREEN, and YELLOW Monsters respectively.

Game Loop

This is how the game works -

- The user will select a player mode.
- Every time a game is played, a maze is created. Even if the game is discarded, the user first needs to create a new maze. This is slightly different from Assignment 1.
- The user can play the game, pause it or resume it.
- The user can view the score.
- Once the user has started the game, PACMAN can move around, eat special food and kill monsters.
- A game ends when either all monsters are killed or a monster kills the PACMAN.
- The user will calculate the game score after the game ends.
- If it is a two-player game, then the user plays for Player 1 first and then plays for Player 2.
- In a two-player game, if the user pauses the game, the user should resume with the player who was playing last.
- At the end of a two-player game, both of their scores are calculated but the ScoreBoard is only updated with the one who has the most points. **If there is a tie, choose Player 1 as a winner.**

The score for a game is calculated using the following formula -

```
Score = 0.5 * numOfHits - 0.25 * numOfMoves + 5 * foodEaten + 10 * monsterKilled +  20 points if no
```

ScoreBoard

The scoreboard comprises of following multiple games. Each Game has these data points -

- a game ID: A game ID is generated whenever we want to play a new game. It starts from 1 and keeps incrementing with a new game. Even if a game is discarded, we keep incrementing the game id.
- Player Name: Player name could only be either `Player 1` or `Player 2` whoever wins the game is based on a higher score. If it is a tie, then `Player 1` wins. If it is a single-player game

then it is always `Player 1` who wins.

- Food Eaten: number of special foods eaten by PACMAN.
- Monster Killed: number of monsters killed by PACMAN.
- Hits: number of times PACMAN hits the boundary or the wall.
- Moves: number of moves PACMAN made before the game ends.
- Score: Total score of the game as mentioned by the formula above.

The format of the scoreboard looks like this -

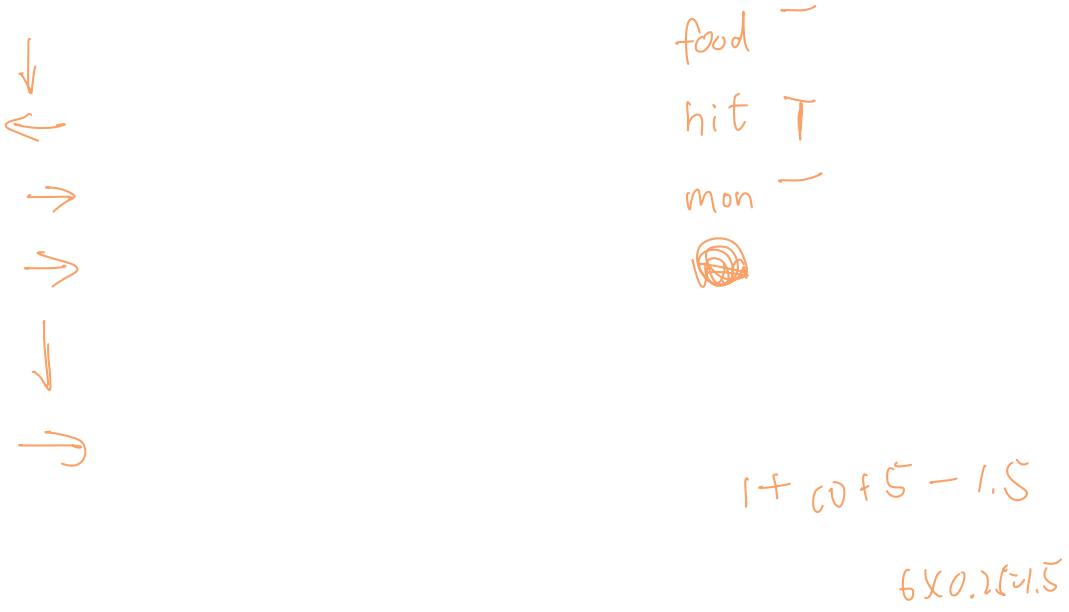
```
|  # Game|    Player Name|   # Food Eaten|# Monster Killed|  # Hits| # Moves| # Score|
|========|===============|===============|================|========|========|========|
|       1|       Player 1|              1|               1|       1|       9|   13.25|
|       5|       Player 2|              1|               1|       0|      14|   11.50|
```

Tip: In above scoreboard game 2, 3, 4 were discarded.

LocationGenerator

In the previous assignment, we provided you with a FoodGenerator class. In this assignment, it is renamed to LocationGenerator as we will be generating positions for monsters and food both.

Warning : Do not change LocationGenerator code.



Main Program Execution & Player Menu

Your main program has slightly changed. Below are the specifications of how your program will be run initially.

Carried Forward: The section mentioned below is same as Assignment 1. Refer to the slides here

Inputs

Your program must take command line inputs in the below order.

- maze length - Length of the maze
- maze width - Width of the maze
- random generator seed - some seed value to be passed to a LocationGenerator. The LocationGenerator code has already been written for you. You need to invoke the constructor and methods appropriately.

This means when you compile your program and run it, the commands on the terminal should look like this

```
javac GameEngine.java
java GameEngine 12 7 123
```

Warning: You must not hard code this in your program. We may provide all kind of integer outputs to test these.

In case there are insufficient inputs (missing input) or if any of the input is 0 or negative, then exit the program by printing the error message -

```
javac GameEngine.java
java GameEngine 12 0 123
Invalid Inputs to set layout. Exiting the program now.
```

If the inputs are valid, the program will show a welcome message. A pre-defined method has already been provided to you in the files. You need to invoke the method appropriately.

```
 ____         __          ___        _  _         __         __ _
(  _ \       / _\        / __)      ( \/ )       / _\       (  ( \
 ) __/      /    \      ( (__       / \/ \      /    \      /    /
(__)        \_/\_/       \___)      \_)(_/      \_/\_/      \_)__)

Let the fun begin
(`<    ...   ...  ...  ..........  ...
```

Out of Scope: You can assume that the command line arguments are always integers and not String/Double or any other data types.

Player Menu

Addition: This specification is entirely new to Assignment 2.

The program must ask whether the user wants to play single-player or multi-player mode.

```
 ____          __           ___        _  _          __          __ _
(  _ \        / _\         / __)      ( \/ )        / _\        (  ( \
 ) __/       /    \       ( (__       / \/ \      /    \       /    /
(__)         \_/\_/        \___)      \_)(_/      \_/\_/       \_)__)

Let the fun begin
(`<    ...   ...  ...  ..........  ...

Make player selection.
Press 1 for Single Player.
Press 2 for Multi Player.
Press 3 to exit.
>
```

Option 1 & 2

Once a player mode is selected the program must continue to print the main menu. Once you exit from the main menu, the program will again prompt for the player menu unless the user selects 3 and exits from the PACMAN game for good.

Carried Forward: The main menu options are same as Assignment 1. Refer to the slides here

```
Make player selection.
Press 1 for Single Player.
Press 2 for Multi Player.
Press 3 to exit.
> 1
Select an option to get started.
Press 1 to select a pacman maze type.
Press 2 to play the game.
Press 3 to resume the game.
Press 4 to view the scores.
Press 5 to exit.
>
```

Option 3

In case, the user selects option 3, you must gracefully quit the program and print a goodbye message.

Warning: Do not use System.exit() as it is not a graceful exit.

```
 ____          __           ___        _  _          __          __ _
(  _ \        / _\         / __)      ( \/ )        / _\        (  ( \
 ) __/       /    \       ( (__       / \/ \       /    \       /    /
(__)         \_/\_/        \___)      \_)(_/       \_/\_/       \_)__)

Let the fun begin
(`<    ...   ...  ...  ..........  ...

Make player selection.
Press 1 for Single Player.
Press 2 for Multi Player.
Press 3 to exit.
> 3
Pacman says - Bye Bye Player.
```

Invalid Input

In case, the user provides an invalid input like 7 or 9, the program should be able to print “Invalid Input” and show the player menu again.

```
Make player selection.
Press 1 for Single Player.
Press 2 for Multi Player.
Press 3 to exit.
> 7
Invalid Input.
Make player selection.
Press 1 for Single Player.
Press 2 for Multi Player.
Press 3 to exit.
>
```

Out of Scope: The program only expects an integer value for the main menu. You need not handle string or double values as input at this point of time.

Main Menu

Main Menu

✅ **Carried Forward:** The section mentioned below is same as Assignment 1. Refer to the slides here

Once the user has selected a player mode, the program should show the main menu.

```
Select an option to get started.
Press 1 to select a pacman maze type.
Press 2 to play the game.
Press 3 to resume the game.
Press 4 to view the scores.
Press 5 to exit.
>
```

Option 5: Exit Option

⚠️ **Modifications:** This section is referred from Assignment 1 with slight modifications.

In case, the user selects option 5, you should show a message to return to the player selection menu. If there is a paused game, exiting from main menu will automatically discard the game. However, game counter will keep incrementing with the new game.

```
Select an option to get started.
Press 1 to select a pacman maze type.
Press 2 to play the game.
Press 3 to resume the game.
Press 4 to view the scores.
Press 5 to exit.
> 5
Exiting main menu, return to Player Selection.
Make player selection.
Press 1 for Single Player.
Press 2 for Multi Player.
Press 3 to exit.
>
```

Invalid Command

✅ **Carried Forward:** The section mentioned below is same as Assignment 1. Refer to the slides here

In case, the user provides an invalid input like 7 or 9, the program should be able to print “Invalid

Input" and show the main menu. Sample output -

```
Select an option to get started.
Press 1 to select a pacman maze type.
Press 2 to play the game.
Press 3 to resume the game.
Press 4 to view the scores.
Press 5 to exit.
> 7
Invalid Input.
Select an option to get started.
Press 1 to select a pacman maze type.
Press 2 to play the game.
Press 3 to resume the game.
Press 4 to view the scores.
Press 5 to exit.
>
```

Out of Scope: The program only expects an integer value for the main menu. You need not handle string or double values as input at this point of time.

Create Maze

Option 1: Create the Maze

Carried Forward: The section mentioned below is same as Assignment 1. Refer to the slides here

This is similar to Assignment 1. How to create a maze is mentioned below.

```
Select an option to get started.
Press 1 to select a pacman maze type.
Press 2 to play the game.
Press 3 to resume the game.
Press 4 to view the scores.
Press 5 to exit.
> 1
Please select a maze type.
Press 1 to select lower triangle maze.
Press 2 to select upper triangle maze.
Press 3 to select horizontal maze.
> 2
Maze created. Proceed to play the game.
Select an option to get started.
Press 1 to select a pacman maze type.
Press 2 to play the game.
Press 3 to resume the game.
Press 4 to view the scores.
Press 5 to exit.
>
```

If the user provide an invalid input then the program must show the message and prompt the user to select a maze again. See the slide here for the details.

```
Select an option to get started.
Press 1 to select a pacman maze type.
Press 2 to play the game.
Press 3 to resume the game.
Press 4 to view the scores.
Press 5 to exit.
> 1
Please select a maze type.
Press 1 to select lower triangle maze.
Press 2 to select upper triangle maze.
Press 3 to select horizontal maze.
> 6
Invalid Input.
Please select a maze type.
Press 1 to select lower triangle maze.
Press 2 to select upper triangle maze.
```

```
Press 3 to select horizontal maze.
> 3
Maze created. Proceed to play the game.
Select an option to get started.
Press 1 to select a pacman maze type.
Press 2 to play the game.
Press 3 to resume the game.
Press 4 to view the scores.
Press 5 to exit.
>
```

How to Create a Maze?

We have looked at different kinds of mazes before and how to create them. In this assignment, you may choose to use Arrays.

> ✕ There will be a deduction in marks if you use ArrayList or any other Collection types. ArrayLists & Collection Types are prohibited.

To understand how mazes work please refer to the slide from Assignment 1 here.

It is important to understand how to generate the position of monsters and food, otherwise your program may fail. Here is a step-by-step guide on how to do it.

Step 1: Create a maze by creating an array. The rows and columns of the array are defined by the maze width and maze length.

Step 2: Start filling up your array with boundaries '#'.

Step 3: Set the location of PACMAN to be row = 1, col = 1.

Step 4: Set the walls based on your maze type. This is similar to Assignment 1.

Step 5: Fill everything else using dots.

Step 6: Create an object for LocationGenerator using the seed. This should be done exactly once.

Step 7: Invoke the generatePosition method to generate a colPos and then rowPos. See below code

```
generatePosition(generator /*you can add other params here*/);
```

Perform step 7 for

- Generating position of Red Monster
- Generating position of Blue Monster
- Generating position of Green Monster
- Generating position of Yellow Monster
- Generating position of 4 special foods. So repeat it 4 times.

This has to be done in the right order for your code to match with the assessment code.

Step 8: Set the generated positions of monsters and food in the maze. These locations were previously filled with dots (Step 5) now they will be overwritten by monster initials (R,B,Y,G) or the special food (*).

Out of Scope Scenario If we are in the middle of the game can the user select a new maze? Ideally no. But this case is not tested here. So you can ignore this use case.

Play the Game

Option 2: Play the Game

When the user selects option 2 to start the game, there are three possibilities-

- The user hasn't selected a maze type.
- The user has selected a maze type and is ready to play the game.
- The user has selected a maze type but is in the middle of a game that is not over yet.

Scenario 1: Maze Type not selected

Carried Forward: This section is referred as is from Assignment 1

In this case, the program should show an error message and repeat the main menu.

```
Select an option to get started.
Press 1 to select a pacman maze type.
Press 2 to play the game.
Press 3 to resume the game.
Press 4 to view the scores.
Press 5 to exit.
> 2
Maze not created. Select option 1 from main menu.
Select an option to get started.
Press 1 to select a pacman maze type.
Press 2 to play the game.
Press 3 to resume the game.
Press 4 to view the scores.
Press 5 to exit.
>
```

Scenario 2: Maze Type selected new game started

Modifications: This section is referred from Assignment 1 with slight modifications.

If the maze is created you can start the game. At the start of the new game, you show some information about how you will gain points and show the user a menu to move the PACMAN. How to move the PACMAN will be shown in the move menu here.

```
Maze created. Proceed to play the game.
Select an option to get started.
Press 1 to select a pacman maze type.
Press 2 to play the game.
Press 3 to resume the game.
```

```
Press 4 to view the scores.
Press 5 to exit.
> 2
Move the Pacman towards the food pellet and gain super power to kill monsters.
  > You gain 20 points if Pacman finishes the game without dying.
  > You gain 10 more points for every monster you killed.
  > You gain 5 points for every special food that you have eaten.
  > You lose 0.5 point when you hit the wall/boundary.
  > You lose 0.25 points for every move.
  > Score = 5 * foodEaten + 10 * monsterKilled - 0.5 * numOfHits - 0.25 * numOfMoves  +  20 if not
Player 1 game begins.
############
#P.........#
#-.Y..B.*..#
#--...R*...#
#---.G**...#
#----......#
############
Press W to move up.
Press A to move left.
Press S to move down.
Press D to move right.
Press Q to exit.
> Q
Your game is paused and saved.
```

Scenario 3: Maze Type selected but previous game in progress

In this case, the program will show a message to the user that the previous game is in progress and ask if the user wants to keep the existing game.

- If yes, the user presses N, then the program prompts the main menu again.
- If not, the user presses any other key, then the program will discard the current game and start over a new game but a new maze needs to be created first. This is slightly different than Assignment 1.

Sample output here -

Output for when the user selects N

Carried Forward: This section is referred as is from Assignment 1

```
Your game is paused and saved.
Select an option to get started.
Press 1 to select a pacman maze type.
Press 2 to play the game.
Press 3 to resume the game.
Press 4 to view the scores.
Press 5 to exit.
> 2
```

```
Previous game hasn't ended yet. Do you want to discard previous game?
Press N to go back to main menu to resume the game or else press any key to discard.
> N
Select an option to get started.
Press 1 to select a pacman maze type.
Press 2 to play the game.
Press 3 to resume the game.
Press 4 to view the scores.
Press 5 to exit.
>
```

Output when the user selects any other key

Modifications: This section is referred from Assignment 1 with slight modifications.

```
Select an option to get started.
Press 1 to select a pacman maze type.
Press 2 to play the game.
Press 3 to resume the game.
Press 4 to view the scores.
Press 5 to exit.
> 2
Previous game hasn't ended yet. Do you want to discard previous game?
Press N to go back to main menu to resume the game or else press any key to discard.
> S
Please select a maze type.
Press 1 to select lower triangle maze.
Press 2 to select upper triangle maze.
Press 3 to select horizontal maze.
> 2
New maze created. Proceed to play the game.
Move the Pacman towards the food pellet and gain super power to kill monsters.
  > You gain 20 points if Pacman finishes the game without dying.
  > You gain 10 more points for every monster you killed.
  > You gain 5 points for every special food that you have eaten.
  > You lose 0.5 point when you hit the wall/boundary.
  > You lost 0.25 points for every move.
  > Score = 5 * foodEaten + 10 * monsterKilled - 0.5 * numOfHits - 0.25 * numOfMoves  +  20 if not
Player 1 game begins.
############
#P.........#
#-.Y..B.*..#
#--...R*...#
#---.G**...#
#----......#
############
Press W to move up.
Press A to move left.
Press S to move down.
Press D to move right.
Press Q to exit.
>
```

Option 3: Resume the Game

A game is paused when the user quit moving the PACMAN to go back to the main menu before the game has ended. The user can only resume an ongoing game. If there is no ongoing game then the program should show an error message. There are four scenarios here -

- The user hasn't selected a maze-type
- The user is resuming a paused game.
- The previous game has ended and the new game hasn't started.
- The user has to exit from the main menu and re-enter the main menu after player selection. In this case, the game has restarted and the user haven't selected a maze type.

Scenario 1 & 4: Maze Type not selected.

Carried Forward: This section is referred as is from Assignment 1

```
Select an option to get started.
Press 1 to select a pacman maze type.
Press 2 to play the game.
Press 3 to resume the game.
Press 4 to view the scores.
Press 5 to exit.
> 3
Maze not created. Select option 1 from main menu.
Select an option to get started.
Press 1 to select a pacman maze type.
Press 2 to play the game.
Press 3 to resume the game.
Press 4 to view the scores.
Press 5 to exit.
>
```

Scenario 2: Resuming a paused game

Modifications: This section is referred from Assignment 1 with slight modifications.

```
Your game is paused and saved.
Select an option to get started.
Press 1 to select a pacman maze type.
Press 2 to play the game.
Press 3 to resume the game.
Press 4 to view the scores.
Press 5 to exit.
> 3
Restart your game from the last position you saved.
Player 1 game begins.
############
```

```
#P.........#
#-.Y..B.*..#
#--...R*...#
#---.G**...#
#----......#
############
Press W to move up.
Press A to move left.
Press S to move down.
Press D to move right.
Press Q to exit.
>
```

Note that it shows that the game is for Player 1.

Scenario 3: No paused game found

Carried Forward: This section is referred as is from Assignment 1

```
Maze created. Proceed to play the game.
Select an option to get started.
Press 1 to select a pacman maze type.
Press 2 to play the game.
Press 3 to resume the game.
Press 4 to view the scores.
Press 5 to exit.
> 3
No paused game found. Select option 2 from main menu to start a new game.
Select an option to get started.
Press 1 to select a pacman maze type.
Press 2 to play the game.
Press 3 to resume the game.
Press 4 to view the scores.
Press 5 to exit.
>
```

Option 4: View Score

There are multiple possible scenarios.

- No completed games found - Show an error message.
- All games have been completed - Show scores of all games.
- Some games have been completed and one game is ongoing - Only show scores of all completed games.

Note: You should maintain a variable for game counter in such a way that every time game begins it is automatically assigned a game Id. Apply some OOP concepts here instead of maintaining plain-old variables in a loop.

Scenario 1: No completed games found

Carried Forward: This section is referred as is from Assignment 1

```
Select an option to get started.
Press 1 to select a pacman maze type.
Press 2 to play the game.
Press 3 to resume the game.
Press 4 to view the scores.
Press 5 to exit.
> 4
No completed games found.
Select an option to get started.
Press 1 to select a pacman maze type.
Press 2 to play the game.
Press 3 to resume the game.
Press 4 to view the scores.
Press 5 to exit.
>
```

Scenario 2 & 3: Show all completed games score

Modifications: This section is referred from Assignment 1 with slight modifications.

```
Select an option to get started.
Press 1 to select a pacman maze type.
Press 2 to play the game.
Press 3 to resume the game.
Press 4 to view the scores.
Press 5 to exit.
> 4
|  # Game|    Player Name|   # Food Eaten|# Monster Killed|  # Hits| # Moves| # Score|
|========|===============|===============|================|========|========|========|
|       1|       Player 1|              1|               1|       1|       9|   13.25|
|       5|       Player 2|              1|               1|       0|      14|   11.50|
Select an option to get started.
Press 1 to select a pacman maze type.
Press 2 to play the game.
Press 3 to resume the game.
Press 4 to view the scores.
Press 5 to exit.
>
```

Tip: Use String.format method to format the strings. Use the String formatter **"|%8s|%15s|%15s|%16s|%8s|%8s|%8s|%n"** for the header and formatter **"|%8d|%15s|%15d|%16d|%8d|%8d|%8.2f|%n"** for printing the actual values.

Out of Scope: When your program terminates you don't have to save the game counter or any other data. The program resets to default.

Move PACMAN

Once the user start/resume the game they should be able to move the PACMAN Up/Down/Right/Left. Moving the PACMAN is previously mentioned here. Look at the sections -

- Invalid Input
- Movement - Right/Left/Top/Bottom
- Invalid Moves
- Exiting from the PACMAN Movement Menu.

If the user resumes the game by selecting option 3 from the main menu, the game must restart at the same location.

Special Moves

Eating a special food

Addition: This specification is entirely new to Assignment 2.

Unlike Assignment 1, if the PACMAN collects a special food, the game should continue. In this assignment, when the PACMAN reaches the special food, they can consume it and their superpower increases by 1. The food is now overwritten by the PACMAN position and later on when the PACMAN moves away, it is now filled with a dot ".". The program also shows a message `Power up!` The number of moves for PACMAN also increases by 1 in this case.

```
############
#P.........#
#*--------.#
#..Y**RB.G.#
#*--------.#
#.........#
############
Press W to move up.
Press A to move left.
Press S to move down.
Press D to move right.
Press Q to exit.
> S
Power up!
############
#.........#
#P--------.#
#..Y**RB.G.#
#*--------.#
#.........#
```

```
############
Press W to move up.
Press A to move left.
Press S to move down.
Press D to move right.
Press Q to exit.
> S
############
#.........#
#.--------.#
#P.Y**RB.G.#
#*--------.#
#.........#
############
Press W to move up.
Press A to move left.
Press S to move down.
Press D to move right.
Press Q to exit.
>
```

Killing a monster

Addition: This specification is entirely new to Assignment 2.

Similarly, if the PACMAN has acquired a superpower (as in the above example) they can now kill a monster. The monster should be removed from the maze. The PACMAN first overrides the monster's position and later on, the position is filled by a dot ".". The program also shows a message `Hurray! A monster is killed.` The program also shows a message `Power up!` The number of moves for PACMAN also increases by 1 in this case.

```
############
#.........#
#.--------.#
#.PY**RB.G.#
#.--------.#
#.........#
############
Press W to move up.
Press A to move left.
Press S to move down.
Press D to move right.
Press Q to exit.
> D
Hurray! A monster is killed.
############
#.........#
#.--------.#
#..P**RB.G.#
#.--------.#
#.........#
```

```
############
Press W to move up.
Press A to move left.
Press S to move down.
Press D to move right.
Press Q to exit.
> A
############
#.........#
#.--------.#
#.P.**RB.G.#
#*--------.#
#.........#
############
Press W to move up.
Press A to move left.
Press S to move down.
Press D to move right.
Press Q to exit.
>
```

Ending a Game

A game ends in two scenario:

- A monster killed PACMAN.
- If PACMAN has killed all the monsters.

Scenario 1: Monster killed PACMAN

If the monster killed PACMAN then the number of moves for PACMAN will not be increased. Also, a message is printed `Boo! Monster killed Pacman.`

```
Move the Pacman towards the food pellet and gain super power to kill monsters.
  > You gain 20 points if Pacman finishes the game without dying.
  > You gain 10 more points for every monster you killed.
  > You gain 5 points for every special food that you have eaten.
  > You lose 0.5 point when you hit the wall/boundary.
  > You lose 0.25 points for every move.
  > Score = 5 * foodEaten + 10 * monsterKilled - 0.5 * numOfHits - 0.25 * numOfMoves  +  20 if not
Player 1 game begins.
############
#P.........#
#-.Y..B.*..#
#--...R*...#
#---.G**...#
#----......#
############
Press W to move up.
Press A to move left.
Press S to move down.
Press D to move right.
```

```
Press Q to exit.
> d
############
#.P........#
#-.Y..B.*..#
#--...R*...#
#---.G**...#
#----......#
############
Press W to move up.
Press A to move left.
Press S to move down.
Press D to move right.
Press Q to exit.
> d
############
#..P.......#
#-.Y..B.*..#
#--...R*...#
#---.G**...#
#----......#
############
Press W to move up.
Press A to move left.
Press S to move down.
Press D to move right.
Press Q to exit.
> s
Boo! Monster killed Pacman.
Game has ended! Your score for this game is -0.5
Select an option to get started.
Press 1 to select a pacman maze type.
Press 2 to play the game.
Press 3 to resume the game.
Press 4 to view the scores.
Press 5 to exit.
>
```

Note: The game score can be negative.

Scenario 2: PACMAN killed all the monsters

```
Select an option to get started.
Press 1 to select a pacman maze type.
Press 2 to play the game.
Press 3 to resume the game.
Press 4 to view the scores.
Press 5 to exit.
> 2
Move the Pacman towards the food pellet and gain super power to kill monsters.
  > You gain 20 points if Pacman finishes the game without dying.
  > You gain 10 more points for every monster you killed.
```

```
  > You gain 5 points for every special food that you have eaten.
  > You lose 0.5 point when you hit the wall/boundary.
  > You lose 0.25 points for every move.
  > Score = 5 * foodEaten + 10 * monsterKilled - 0.5 * numOfHits - 0.25 * numOfMoves  +  20 if not
Player 1 game begins.
############
#P.........#
#*--------.#
#..Y**RB.G.#
#*--------.#
#.........#
############
Press W to move up.
Press A to move left.
Press S to move down.
Press D to move right.
Press Q to exit.
> S
Power up!
############
#.........#
#P--------.#
#..Y**RB.G.#
#*--------.#
#.........#
############
Press W to move up.
Press A to move left.
Press S to move down.
Press D to move right.
Press Q to exit.
> S
############
#.........#
#.--------.#
#P.Y**RB.G.#
#*--------.#
#.........#
############
Press W to move up.
Press A to move left.
Press S to move down.
Press D to move right.
Press Q to exit.
> S
Power up!
############
#.........#
#.--------.#
#..Y**RB.G.#
#P--------.#
#.........#
############
Press W to move up.
```

```
Press A to move left.
Press S to move down.
Press D to move right.
Press Q to exit.
> W
############
#.........#
#.--------.#
#P.Y**RB.G.#
#.--------.#
#.........#
############
Press W to move up.
Press A to move left.
Press S to move down.
Press D to move right.
Press Q to exit.
> D
############
#.........#
#.--------.#
#.PY**RB.G.#
#.--------.#
#.........#
############
Press W to move up.
Press A to move left.
Press S to move down.
Press D to move right.
Press Q to exit.
> D
Hurray! A monster is killed.
############
#.........#
#.--------.#
#..P**RB.G.#
#.--------.#
#.........#
############
Press W to move up.
Press A to move left.
Press S to move down.
Press D to move right.
Press Q to exit.
> D
Power up!
############
#.........#
#.--------.#
#...P*RB.G.#
#.--------.#
#.........#
############
Press W to move up.
```

```
Press A to move left.
Press S to move down.
Press D to move right.
Press Q to exit.
> D
Power up!
############
#..........#
#.--------.#
#....PRB.G.#
#.--------.#
#..........#
############
Press W to move up.
Press A to move left.
Press S to move down.
Press D to move right.
Press Q to exit.
> D
Hurray! A monster is killed.
############
#..........#
#.--------.#
#.....PB.G.#
#.--------.#
#..........#
############
Press W to move up.
Press A to move left.
Press S to move down.
Press D to move right.
Press Q to exit.
> D
Hurray! A monster is killed.
############
#..........#
#.--------.#
#......P.G.#
#.--------.#
#..........#
############
Press W to move up.
Press A to move left.
Press S to move down.
Press D to move right.
Press Q to exit.
> D
############
#..........#
#.--------.#
#.......PG.#
#.--------.#
#..........#
############
```

```
Press W to move up.
Press A to move left.
Press S to move down.
Press D to move right.
Press Q to exit.
> D
Hurray! A monster is killed.
Game has ended! Your score for this game is 77.0
Select an option to get started.
Press 1 to select a pacman maze type.
Press 2 to play the game.
Press 3 to resume the game.
Press 4 to view the scores.
Press 5 to exit.
>
```

Multi-Player Mode

When the user selects Multi-Player Mode, all the instructions remain the same. However, you will play the game one player at a time starting from Player 1 and then moving to Player 2. These are some game flows to be considered

- Create a maze for each player.
- Player 1 plays and completes a game.
- Player 2 plays and completes a game.

Both game mazes will have the same initial configuration (maze type/ length/ width). If a player pauses the game, they must resume and finish it before the next player can play.

At the end of the game, you also need to show which player won, before returning to the main menu. Look for the message `Player 2 wins. Returning to main menu.` or `Player 1 wins. Returning to main menu.` This message is not present for single-player mode.

Examples:

- Player 1 plays the game and ends the game, Player 2 will start the game and end the game.
- Player 1 plays the game and pauses it. The game should resume from Player 1. Player 2 cannot play until player 1 is finished.
- Player 1 plays the game and ends the game. Player 2 starts the game and pauses it. The game will resume with Player 2.
- Player 1 plays, pauses, ends. Player 2 plays, pauses, ends.

Scenario 1: Player 1 plays and then player 2 plays.

Note: Some of the messages are highlighted for you. You don't have to make them bold in your output.

```
Make player selection.
Press 1 for Single Player.
Press 2 for Multi Player.
Press 3 to exit.
> 2
Select an option to get started.
Press 1 to select a pacman maze type.
Press 2 to play the game.
Press 3 to resume the game.
Press 4 to view the scores.
Press 5 to exit.
> 1
Please select a maze type.
Press 1 to select lower triangle maze.
```

```
Press 2 to select upper triangle maze.
Press 3 to select horizontal maze.
> 2
Maze created. Proceed to play the game.
Select an option to get started.
Press 1 to select a pacman maze type.
Press 2 to play the game.
Press 3 to resume the game.
Press 4 to view the scores.
Press 5 to exit.
> 2
Move the Pacman towards the food pellet and gain super power to kill monsters.
  > You gain 20 points if Pacman finishes the game without dying.
  > You gain 10 more points for every monster you killed.
  > You gain 5 points for every special food that you have eaten.
  > You lose 0.5 point when you hit the wall/boundary.
  > You lose 0.25 points for every move.
  > Score = 5 * foodEaten + 10 * monsterKilled - 0.5 * numOfHits - 0.25 * numOfMoves  +  20 if not
Player 1 game begins.
############
#P.........#
#-.Y..B.*..#
#--...R*...#
#---.G**...#
#----......#
############
Press W to move up.
Press A to move left.
Press S to move down.
Press D to move right.
Press Q to exit.
> d
############
#.P........#
#-.Y..B.*..#
#--...R*...#
#---.G**...#
#----......#
############
Press W to move up.
Press A to move left.
Press S to move down.
Press D to move right.
Press Q to exit.
> d
############
#..P.......#
#-.Y..B.*..#
#--...R*...#
#---.G**...#
#----......#
############
Press W to move up.
Press A to move left.
```

```
Press S to move down.
Press D to move right.
Press Q to exit.
> s
Boo! Monster killed Pacman.
Game has ended! Your score for this game is -0.5
Move the Pacman towards the food pellet and gain super power to kill monsters.
  > You gain 20 points if Pacman finishes the game without dying.
  > You gain 10 more points for every monster you killed.
  > You gain 5 points for every special food that you have eaten.
  > You lose 0.5 point when you hit the wall/boundary.
  > You lose 0.25 points for every move.
  > Score = 5 * foodEaten + 10 * monsterKilled - 0.5 * numOfHits - 0.25 * numOfMoves  +  20 if not
Player 2 game begins.
############
#P.........#
#-.Y..B.*..#
#--...R*...#
#---.G**...#
#----......#
############
Press W to move up.
Press A to move left.
Press S to move down.
Press D to move right.
Press Q to exit.
> d
############
#.P........#
#-.Y..B.*..#
#--...R*...#
#---.G**...#
#----......#
############
Press W to move up.
Press A to move left.
Press S to move down.
Press D to move right.
Press Q to exit.
> d
############
#..P.......#
#-.Y..B.*..#
#--...R*...#
#---.G**...#
#----......#
############
Press W to move up.
Press A to move left.
Press S to move down.
Press D to move right.
Press Q to exit.
> s
Boo! Monster killed Pacman.
```

```
Game has ended! Your score for this game is -0.5
Player 1 wins. Returning to main menu.
Select an option to get started.
Press 1 to select a pacman maze type.
Press 2 to play the game.
Press 3 to resume the game.
Press 4 to view the scores.
Press 5 to exit.
> 4
|  # Game|    Player Name|   # Food Eaten|# Monster Killed|  # Hits| # Moves| # Score|
|========|===============|===============|================|========|========|========|
|       1|       Player 1|              0|               0|       0|       2|   -0.50|
Select an option to get started.
Press 1 to select a pacman maze type.
Press 2 to play the game.
Press 3 to resume the game.
Press 4 to view the scores.
Press 5 to exit.
>
```

Note: See how both players made same move and has same scores but we chose Player 1 since there is a tie.

Scenario 2: Player 1 plays pauses and resumes. Then Player 2 plays.

```
Select an option to get started.
Press 1 to select a pacman maze type.
Press 2 to play the game.
Press 3 to resume the game.
Press 4 to view the scores.
Press 5 to exit.
> 2
Move the Pacman towards the food pellet and gain super power to kill monsters.
  > You gain 20 points if Pacman finishes the game without dying.
  > You gain 10 more points for every monster you killed.
  > You gain 5 points for every special food that you have eaten.
  > You lose 0.5 point when you hit the wall/boundary.
  > You lose 0.25 points for every move.
  > Score = 5 * foodEaten + 10 * monsterKilled - 0.5 * numOfHits - 0.25 * numOfMoves  +  20 if not
Player 1 game begins.
############
#P.........#
#-.Y..B.*..#
#--...R*...#
#---.G**...#
#----......#
############
Press W to move up.
Press A to move left.
Press S to move down.
Press D to move right.
Press Q to exit.
> d
```

```
############
#.P........#
#-.Y..B.*..#
#--...R*...#
#---.G**...#
#----......#
############
Press W to move up.
Press A to move left.
Press S to move down.
Press D to move right.
Press Q to exit.
> Q
Your game is paused and saved.
Select an option to get started.
Press 1 to select a pacman maze type.
Press 2 to play the game.
Press 3 to resume the game.
Press 4 to view the scores.
Press 5 to exit.
> 3
Restart your game from the last position you saved.
Player 1 game begins.
############
#.P........#
#-.Y..B.*..#
#--...R*...#
#---.G**...#
#----......#
############
Press W to move up.
Press A to move left.
Press S to move down.
Press D to move right.
Press Q to exit.
> d
############
#..P.......#
#-.Y..B.*..#
#--...R*...#
#---.G**...#
#----......#
############
Press W to move up.
Press A to move left.
Press S to move down.
Press D to move right.
Press Q to exit.
> s
Boo! Monster killed Pacman.
Game has ended! Your score for this game is -0.5
Move the Pacman towards the food pellet and gain super power to kill monsters.
  > You gain 20 points if Pacman finishes the game without dying.
  > You gain 10 more points for every monster you killed.
```

```
  > You gain 5 points for every special food that you have eaten.
  > You lose 0.5 point when you hit the wall/boundary.
  > You lose 0.25 points for every move.
  > Score = 5 * foodEaten + 10 * monsterKilled - 0.5 * numOfHits - 0.25 * numOfMoves  +  20 if not
Player 2 game begins.
############
#P.........#
#-.Y..B.*..#
#--...R*...#
#---.G**...#
#----......#
############
Press W to move up.
Press A to move left.
Press S to move down.
Press D to move right.
Press Q to exit.
>
```

Scenario 3: Player 1 plays then player 2 plays, pauses and resumes.

```
Move the Pacman towards the food pellet and gain super power to kill monsters.
  > You gain 20 points if Pacman finishes the game without dying.
  > You gain 10 more points for every monster you killed.
  > You gain 5 points for every special food that you have eaten.
  > You lose 0.5 point when you hit the wall/boundary.
  > You lose 0.25 points for every move.
  > Score = 5 * foodEaten + 10 * monsterKilled - 0.5 * numOfHits - 0.25 * numOfMoves  +  20 if not
Player 1 game begins.
############
#P.........#
#-.Y..B.*..#
#--...R*...#
#---.G**...#
#----......#
############
Press W to move up.
Press A to move left.
Press S to move down.
Press D to move right.
Press Q to exit.
> d
############
#.P........#
#-.Y..B.*..#
#--...R*...#
#---.G**...#
#----......#
############
Press W to move up.
Press A to move left.
Press S to move down.
Press D to move right.
```

```
Press Q to exit.
> d
############
#..P.......#
#-.Y..B.*..#
#--...R*...#
#---.G**...#
#----......#
############
Press W to move up.
Press A to move left.
Press S to move down.
Press D to move right.
Press Q to exit.
> s
Boo! Monster killed Pacman.
Game has ended! Your score for this game is -0.5
Move the Pacman towards the food pellet and gain super power to kill monsters.
  > You gain 20 points if Pacman finishes the game without dying.
  > You gain 10 more points for every monster you killed.
  > You gain 5 points for every special food that you have eaten.
  > You lose 0.5 point when you hit the wall/boundary.
  > You lose 0.25 points for every move.
  > Score = 5 * foodEaten + 10 * monsterKilled - 0.5 * numOfHits - 0.25 * numOfMoves  +  20 if not
Player 2 game begins.
############
#P.........#
#-.Y..B.*..#
#--...R*...#
#---.G**...#
#----......#
############
Press W to move up.
Press A to move left.
Press S to move down.
Press D to move right.
Press Q to exit.
> d
############
#.P........#
#-.Y..B.*..#
#--...R*...#
#---.G**...#
#----......#
############
Press W to move up.
Press A to move left.
Press S to move down.
Press D to move right.
Press Q to exit.
> Q
Your game is paused and saved.
Select an option to get started.
Press 1 to select a pacman maze type.
```

```
Press 2 to play the game.
Press 3 to resume the game.
Press 4 to view the scores.
Press 5 to exit.
> 3
Restart your game from the last position you saved.
Player 2 game begins.
############
#.P........#
#-.Y..B.*..#
#--...R*...#
#---.G**...#
#----......#
############
Press W to move up.
Press A to move left.
Press S to move down.
Press D to move right.
Press Q to exit.
> d
############
#..P.......#
#-.Y..B.*..#
#--...R*...#
#---.G**...#
#----......#
############
Press W to move up.
Press A to move left.
Press S to move down.
Press D to move right.
Press Q to exit.
>
```

Scenario 4: Player 1 plays, pauses, and ends. Player 2 plays, pauses, ends.

You can refer back to the message above. Understand which message needs to be shown and when.

Guidance: Object Oriented Programming

Think Object Oriented Programming, Think Classes!

To assist you with writing some code, some empty files with comments are mentioned. You can find the description below about how to think from an Object Oriented Programming approach and which classes should be created.

- Create different entities. Some of them are related to Pacman while others are related to game and scorekeeping. Start by defining data variables in these classes.
- The maze should be an array. Define the array in a class and write methods to manipulate the array.
- Add methods in these classes. Decide which method goes where. Does Maze move? or does the PACMAN move? Where should the move method go?
- Game IDs were variable in the previous assignment and were incremented with loop runs. Have we learnt a way in the class where we can auto-increment game ID every time an object of the game is created?
- ScoreBoard contains a series of games and their scores. To do so you can create an array. Think about what would be the initial size of this array and how can you demonstrate Array-resizing.

Please note that in Object Oriented Programming, we try to abstract out code into many classes that make logical sense. Creating one God-Like Class is a paradigm we avoid. Read more about God Classes here.

Packages

You should group the logical entities in packages. One suggestion is to group the PACMAN/Monsters into a package. These packages are not created for you and we recommend you move the provided classes to necessary packages. **Can you think of anything else?**


Please do not move GameEngine.java into any packages.

JavaDoc

You must annotate your classes and methods using Javadoc-style comments. How to create Javadoc is shown in the class. Remember, to generate Javadocs at the root directory you should run

```
javadoc -d docs/ *.java
```

However, the above method will not generate javadocs for classes in a package. You must additionally run the below command on your terminal.

```
javadoc -d docs/ */*.java
```

Your program should generate all the javadocs without any warnings/errors. **The warning related to default constructor can be ignored but I suggest you create a default constructor in your classes.**

PACMAN got scary!

Some starter codes have been provided for you. Write the code to implement your program there

We also provided some tests for your code, which will run automatically every time you hit "Mark". Your submission is saved when you hit the Mark button. You can make as many submissions as you want and run these tests as many times as you would like until the submission deadline.

Submission will close on 20 Sep 2024 9 AM AEST.

Happy coding!

Assessment

This project is worth **15%** of the total marks for the subject. Your Java program will be assessed based on the correctness of the output as well as the quality of code implementation.

Automatic tests will be conducted on your program by compiling, running, and comparing your outputs for several test cases with generated expected outputs. The automatic test will deem your output wrong if your output does not match the expected output, even if the difference is just having extra space or missing a colon. Therefore, it is crucial that your output follows exactly the same format shown in the provided examples.

Passing the tests we provide here on edstem does not mean you will score full marks. Each submission will further be subject to hidden tests as well as manual inspection by our teaching team.

Marking Scheme

Warning! You must make sure your code runs on edstem. If your code does not compile on edstem, 0 marks will be given.

COMP90041 Assignment 2: Marking Scheme

Student: ***[Student ID goes here]***

Program Presentation

Including layout and style, readability, adherence to coding expectations and conventions, general care and appearance. The full marks for this section of marking is **4**.

Gaining marks

Gain **0.5** marks for any of the achievements listed below.

- all choices of variable (except array indices), and method names were meaningful;
- variable, method, and class names follow Java convention (camelCase and PascalCase);
- Constant names follow Java convention (UPPER_SNAKE_CASE);
- comments were sufficient and meaningful; private methods or complex code blocks have appropriate comments.
- consistent bracket placement and indentation;
- authorship statement (name, university email, student number) provided;
- all magic numbers (essentially numbers other than 0 or 1) were assigned to constants; repeated Strings are created as constants. constants were defined as `final static`;
- clearly structured code with the correct order of variables, constructors, getters/setters, and methods.

Deductions

- stylistic issue, if major **1.0** mark per issue; if minor, **0.5** mark per issue. The minimum mark in this section can be 0.

Structure and Approach

Including decomposition into methods, declaration of instance variables at the appropriate locations, and choice of parameters to methods.

The full mark for this section of marking is **11**.

Gaining marks

Gain **1** mark for any achievement listed below.

- Good use of methods; No methods were too long or too complex;
- code was not duplicated (tasks to be done in more than one place are in the method);
- no more than 3 `static` methods (including main) were used -- most methods should be bound to objects;
- no more than 4 `static` variables were used -- most variables should be non-static **("`final static`" constants are not variables); Must use at least one static variable and method. You should read the specifications to identify how can you use static.**
- appropriate use of encapsulation; limiting the scope of variables to `private` or `protected` except where reasonable justification is given. **No privacy leaks are present.**
- **use of Arrays, creating 1D and 2D arrays wherever appropriate.**
- **Array resizing demonstrated.**
- **enums created**
- classes have high cohesion or low coupling ie appropriate methods are placed in appropriate classes.
- JavaDoc has been generated appropriately for all classes in folder and subfolders without any warnings. Comments should be present for all constructors/public methods/public variables. Warnings for default constructor can be ignored.
- At least 2 packages have been created to group the related classes.

Deductions

- structural issues, if major **2.0** marks per issue, otherwise **1.0** per issue. The minimum mark in this section can be 0.
- ✕ **Use of ArrayList/other Collection types gets a penalty of 3.0 marks**

Program Execution

Including compilation, execution of test data, output presentation, and readability.

Programs that do not compile in the test environment will lose all marks in this section. Be sure to verify your submission and **check the output** before you say "finished" to yourself.

The full mark for this section of marking is **15**.

Marks awarded

Gain **1** marks per test case passed.

Deductions

Gain **0.5** marks for tests with ***slightly*** different output (e.g., small changes in whitespace/newline are fine but a lot of white spaces or new line skipped may be an issue; if the marker says the difference is not slight, that is final. This should not occur if the available test cases are all checked. White space issues in visible test cases will be penalised);

Gain **0** for other failed tests

Visible tests passed: / 10

Hidden tests passed: / 5

Total tests passed: /15

Total marks for this section: **/ 15**

Total Marks for the Assignment: / 30

Total Marks for the Assignment(scaled) : marks out of 30 /2 = (marks out of 15)

Overall comments from marker: ***[Comments go here]***

Assignment Marker: ***[Assignment marker name goes here]***

If you have any questions regarding your mark, please contact the lecturers

Submission

Your submission should have the Java classes already provided to you and additional classes if you wish to include them. Your submission should also contain some packages.

⚠ **Rearrange the classes to create some packages and update the import statements accordingly.**

A starter code has been provided to you here on the edstem platform, but you are welcome to develop the project outside of the edstem environment, using your own IDEs.

Submission is made via edstem, however, and will be whatever code is saved on the Code Challenge when the project deadline closes.

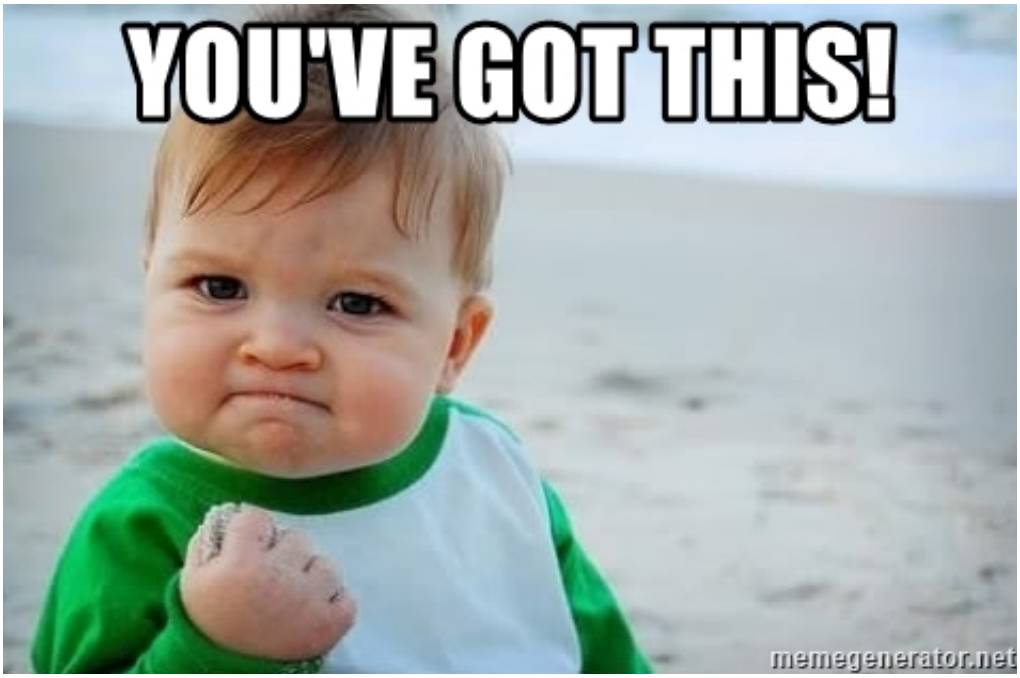
⚠ Your code MUST compile and run here on edstem. If your code does not compile we cannot mark it and you risk getting 0 marks.

"I can't get my code to work on edstem but it worked on my local machine" is not an acceptable excuse for late submissions. In addition, **submissions via email will not be accepted.**

Be sure to copy your code into your Code Challenge workspace well before the deadline to avoid being locked out of submissions last minute.

Only the last version of your code before the submission deadline will be graded. It is highly recommended that you update your code on edstem frequently and well before the submission deadline. Last-minute *"connection errors"* are not a valid excuse.

In case you need an extension due to valid reasons up to 3 days, please fill out the form mentioned in the slides here and send it to the subject coordinator's email. We will follow up with you. Ensure you have a valid reason and proper documentation with you before seeking the extension. If you seek an extension beyond 3 days, check the new Special Consideration Module on Canvas and raise your request using this portal. See more here.


YOU'VE GOT THIS!
memegenerator.net