



SOFT2201/COMP9201

Assignment 3

Due: Sunday 27 October 2024, 11:59PM AEST

This assignment is worth 20% of your final assessment

IMPORTANT: You are only allowed to implement Assignment 3 Requirements in this assignment. Do not include any other features, otherwise mark deductions will be applied.

IMPORTANT: You must use the template we have provided and only the JavaFX library for your GUI. You may not use any other GUI library.

Task Overview

In assignment 3, you will extend and review a given codebase for the Pac-Man game. Please note that your goal is to extend and maintain this implementation. This means that you will add features to the existing implementation by using OO design principles and appropriate design patterns that you have learned throughout this UoS. Please ensure that you do this without breaking the existing implementation or using unnecessary 'hacks'.

You are not required to correct the existing design of the implementation - you must retain the existing design wherever changes are not required and you will be penalised for making changes without cause (for example, replacing the given implementation with your own assignment 2 code). The idea here is that you work with the existing design rather than against it, and minimise required changes to the existing structure (reasonable, limited-scope refactoring to support extensions is encouraged). To be clear, this means that you are not required to correct the given codebase to align with A2 requirements.

You **must provide the following features** in your implementation with at least 3 design patterns (that have not been used in the given scaffold/Assignment 2). You will find a detailed description of each feature in a later section.

- **Additional Ghost Types:** There are now 4 ghost types, each with their own pre-assigned corner of the map for SCATTER mode and a different strategy for chasing Pac-Man in CHASE mode.
- **Power Pellet:** When Pac-Man eats a power pellet, Pac-Man will temporarily gain the ability to eat ghosts and can earn additional points from eating them.
- **FRIGHTENED mode:** When Pac-Man eats a power pellet, the ghosts temporarily enter into FRIGHTENED mode, where they will become slower and move randomly through the maze. If Pac-Man collides with a ghost in FRIGHTENED mode, the ghost will die and be re-spawn in SCATTER mode.

What we provide to you

In the [code scaffold](#) given to you, you will be able to find:

- **JSON file:** A configuration file (`config.json`).
- **Map files:** A map text file (`map.txt`) for ASM2 and a map text file (`new-map.txt`) for ASM3.
- **Code Scaffold:** A scaffold codebase that you must use to extend with the requested features.
 - You must use this scaffold for Assignment 3. You are only allowed to make changes to it necessary for the implementation of the requested features.
 - The scaffold is runnable in its current state. Do not break it in your implementation.
 - You must use JavaFX for your GUI - no other GUI library is allowed.
 - You will need to update the scaffold codebase to use the `new-map.txt` file, once you have implemented the relevant features (e.g. the different ghost types, power pellets).
 - Please do not modify the configuration and map text files.
- **Sprites:** You will be using these to render the game entities. These can be found in `src/main/resources`.
- **Font:** The font used to display the game messages and score. This can be found in `src/main/resources`.

Note, to download the scaffold, open the File Explorer on the EdStem code submission page, and right-click to Download all files.

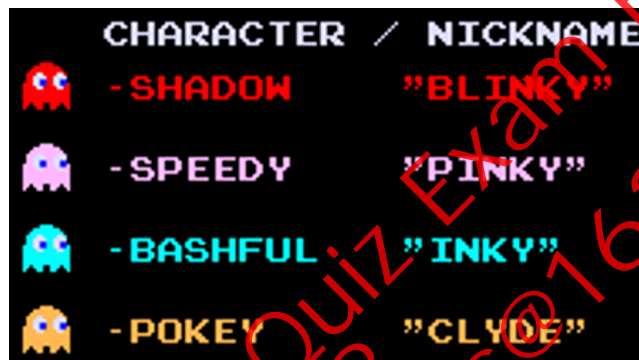
What we expect from you

Implementation Task (10 marks)

You will use the Java Programming Language to extend the given Pac-Man game with the features described below.

Additional Ghost Types

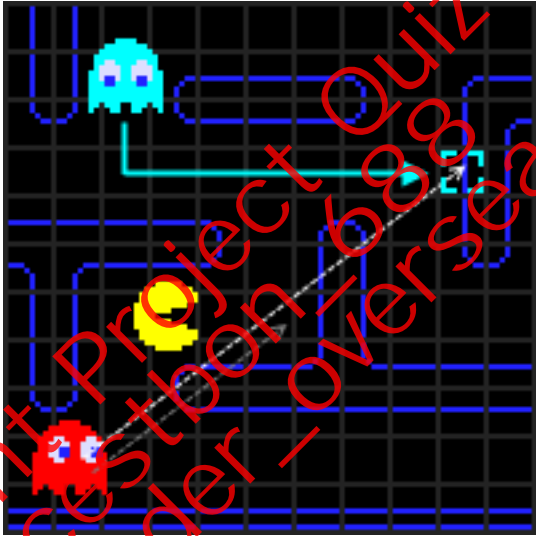
There are now 4 different Ghost types, each with their own unique behaviour. The Ghost types are described below.



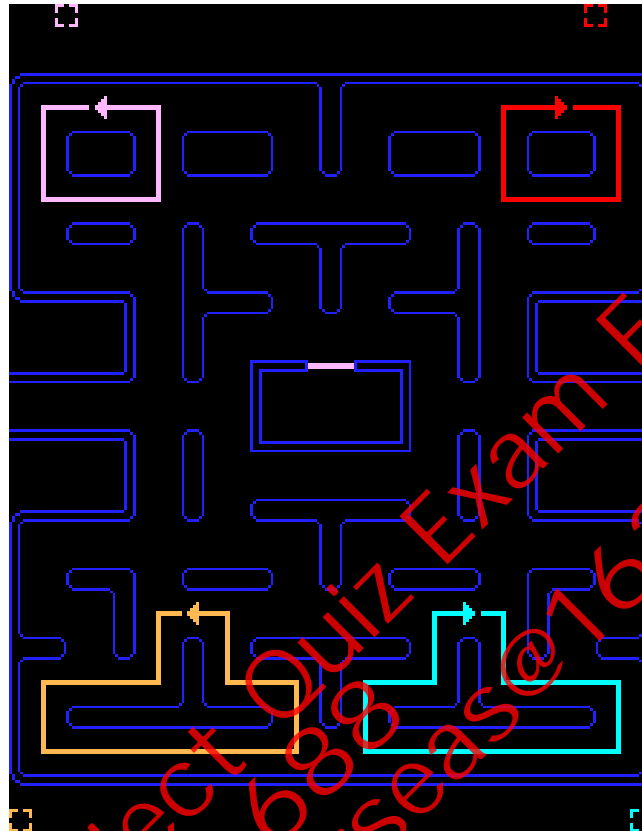
Their map file representations are as follows:

Ghost	Character
Shadow/Blinky	b
Speedy/Pinky	s
Bashful/Inky	i
Pokey/Clyde	c

Their movement patterns are the same as that of the Assignment 2 Ghost, in that they alternate between SCATTER mode – where they move towards one of the corners of the map, and CHASE mode, where they identify the target location and proceed in the direction closest to that location. The following outlines the specific adjustments to their behaviors:

Ghost	CHASE - Target Location	SCATTER - Target Location
Speedy/Pinky	Four grid spaces ahead of Pac-Man (based on Pac-Man's current direction)	Top-left corner
Shadow/Blinky	Pac-Man's position	Top-right corner
Pokey/Clyde	If more than 8 grid spaces away from Pac-Man (straight line distance), its target location is Pac-Man. Otherwise, its target location is the bottom-left corner	Bottom-left corner
Bashful/Inky	<p>Bashful's target position is found by first determining the position two grid spaces ahead of Pac-Man, and then doubling the vector from Shadow/Blinky to that position. For example, if Pac-Man is moving right at (3, 6) and Shadow/Blinky is at (1, 9), the position two spaces ahead of Pac-Man would be (5, 6). The vector from Shadow/Blinky to (5, 6) is (4, -3). Doubling this vector gives (8, -6), making Bashful's target location (9, 3).</p>  <p>Source</p>	Bottom-right corner

Below graphic may help you in implementing the target location of each ghost in SCATTER mode. The squares describe the approximate target location of each ghost, and the arrows illustrate the path of the ghosts if they were to only travel in SCATTER mode.



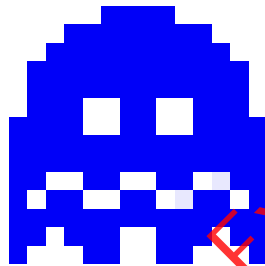
Recall that each grid space is 16 x 16 pixels.

In cases where the target location is outside the bounds of the grid, the closest point is used. For example, if a target location is $(-3, 4)$, the Ghost would target $(0, 4)$.

Power Pellet

There is an additional type of collectable called power pellet. It is twice the size of a normal pellet and is represented in the map file with the character 'z'. Note that you will have to apply an offset of (-8, -8) to centre the pellet in its grid space.

When Pac-Man collects a power pellet, all Ghosts enter FRIGHTENED mode. The frightened mode lasts for a predetermined amount of time unless the ghost is eaten by the Pac-Man. While in FRIGHTENED mode, Ghosts navigate intersections by making random turning decisions and will only reverse their direction if no other options are available. When in FRIGHTENED mode, Ghosts use an alternate sprite:



If Pac-Man collides/eats a Ghost while it is in FRIGHTENED mode, the player will gain additional points. During a single FRIGHTENED mode period, the points awarded to the player increase with each consecutive ghost eaten:

- 1st ghost: 200 points
- 2nd ghost: 400 points
- 3rd ghost: 800 points
- 4th ghost: 1600 points

If a Ghost is eaten during FRIGHTENED mode, it respawns at its original location and, after a 1-second delay, resumes movement through the maze in SCATTER mode using its standard sprite. After the FRIGHTENED mode ends, all ghosts should return to their normal sprite and continue through the maze in SCATTER mode. It's important to manage the duration of each Ghost's FRIGHTENED modes individually. For example, if Ghost A gets eaten during FRIGHTENED mode, it will respawn and switch to SCATTER mode, whereas the other ghosts will remain in FRIGHTENED mode until the FRIGHTENED mode duration expires.

For each level, the config file specifies the duration of FRIGHTENED mode in seconds in the `modeLengths` object and the speed of the ghosts during FRIGHTENED mode in the `ghostSpeed` object. Note that the ghosts are slower during FRIGHTENED mode.

If Pac-Man consumes another power pellet while already in FRIGHTENED mode, the FRIGHTENED mode timer resets. All Ghosts, including those that have respawned, re-enter FRIGHTENED mode under this new timer. In this scenario, a new FRIGHTENED period begins, and the player's point count starts again from 200 for each Ghost eaten.

As with normal pellets, after Pac-Man has eaten a power pellet, it should disappear from the map.

IMPORTANT: You can only implement the above features in your assignment and you should not include any other features. Mark deductions will be applied otherwise.

Assignment Project Quiz Exam Essay Help
WeChat: cestbon-6888
Email: accoder-overseas@163.com

Report Task (10 marks)

For your report, you must include the following sections:

1. **Code Review** Complete a code review of the existing codebase provided to you, which includes:
 - discussion on the use of OOP or design principles (**be specific** to the given code, a UML snippet needs to be provided)
 - NOTE: You do not have to discuss all design principles, only the ones you find relevant to the scaffold.
 - discussion on the use of design patterns (**be specific** to the given code, a UML snippet needs to be provided)
 - You should discuss all design patterns that have been implemented in the scaffold codebase, excluding the use of the Builder pattern.
 - Please provide a UML snippet for each design pattern you discuss, including its participants
 - discussion on the documentation (e.g., comments, etc.)
 - discussion on how easy or difficult the given codebase was to extend. In particular, take into account how the above points affected your ability to achieve the required functionality in this assignment.
2. **Feature Extension.** Discuss your extension of the codebase and implementation of the requested features.
 - Describe the actual changes (including extensions) you have made in your code and rationalise that the changes are necessary.
 - Document at least three different GoF design patterns used in your assignment, ensuring they are among those covered in the course. Justify their use based on SOLID and/or GRASP principles specific to your code, and include a UML diagram for each pattern showing the corresponding participants.
 - While you may include patterns from Assignment 2 (Factory, Singleton, Observer, Command) as part of your implementation, they do not count toward the required three patterns.
 - Ensure the new features utilise different design patterns from those in A2. (i.e. none of the features require the use of design patterns from A2)
 - Reflect on your extension design, highlighting any outstanding issues or improvements or discussing your impact on the extensibility of the code
3. Any acknowledgement/reference required.

Notes:

- When creating your UML diagrams, include only the classes/interfaces and relationships that are directly relevant to your discussion. Additionally, you may omit methods/attributes in classes that are unrelated to your current discussion point. For example, when explaining how a design pattern is used to implement a feature, display only the methods and attributes that are essential to that particular implementation.
- You must ensure your diagrams are readable/understandable to the marker (e.g. avoid crossing lines where possible). If your marker cannot understand your diagram, zero marks will be given for the corresponding section.
- Your marker will not download your PDF locally, so please ensure your diagrams are readable in Canvas. Please note the maximum amount you can zoom in on Canvas is 200%. If your diagrams are not clearly readable at 200% zoom, please use a higher screen quality when exporting it from your UML tool of choice, or export it as a PDF and combine it with your report PDF.

Assignment Project Quiz Exam Essay Help
WeChat: cestbon-6888
Email: accoder-overseas@163.com

Submission Details

You are required to submit all assessment items to their submission portals by the due date. This includes:

- **Report** (10 marks): Submit the report as a **SINGLE PDF** document on the [Canvas page for the report](#).
- **Code** (10 marks): Your code should be submitted as a **ZIP file containing only your src folder, build.gradle, and README** on the [EdStem page for the code](#).

– Compile & Run

- * **IMPORTANT:** The sample JSON configuration file and map.txt file must be included in your `src/main/resources` folder. Mark deductions will be applied if they are placed incorrectly.
- * The README file has to cover any details you want your marker to know. In your README, you must include the following items:
 - how to run your code (e.g., any quirks to run your application)
 - which files and classes are involved in each design pattern implemented
 - anything else that you would like your marker to know
- * We will execute your code by running `gradle clean build run` in the terminal with the environment configuration below:
 - Gradle 7.4.2
 - JDK 17
 - Unix-based System
- * If your code fails to run using the instructions provided above, you will receive a **ZERO** mark for the coding portion of this assessment.

– Style & Documentation

- * Please follow the [Google Java Style Guide](#)
- * Ensure names used are meaningful and the structure is clear and consistent
- * Javadoc is not required, but please ensure you provide comments when needed.
- * Put your pattern-related files in packages with pattern-related names, e.g. `observer`, `command`, `factory`