Assignment 1: Playing an Arcade Game with AI

1. Group project: 2 – 4 members per group.

2. URL: https://github.com/liewweishiung/MsPacMan

3. Download and extract the zip file. Follow the setup instructions below in this document.

4. To modify the program parameters, use the file "Main.java" under "src/main/java/(default package)/".
   The folder "src/main/java/examples.StarterPacMan" contains the algorithms for controlling Ms Pac Man, including "MyPacMan.java" and "TreeSearchPacMan.java".
   To change the type of algorithm for controlling Ms Pac Man, you just need to modify the line in "Main.java", for example:

   ```
   executor.runGame(new FirstCustomAI(), ghosts, 1);
   ```

5. For Assignment 1, **implement two or more types of search algorithms for controlling Ms Pac Man**. Each algorithm should be a separate Java file in the folder "src/main/java/examples.StarterPacMan".
   For example: "FirstCustomAI.java", "SecondCustomAI.java", and "ThirdCustomAI.java".
   You can use any algorithm. For example, tree-search algorithms like MCTS and A-Star can be run immediately, while reinforcement learning or neuro-evolution may require some training to optimize the gameplay.
   You can find examples of how to implement search algorithms in Pac Man. For example:
   - A-Star.
   - Genetic Algorithms.
   - Monte Carlo.
   - Neural Networks.

6. In addition, **propose two or more fitness functions or evaluation metrics for measuring the performance of the algorithms**. For example:
   a. FitnessScore = score – total time elapsed.
   b. FitnessScore = number of ghosts eaten
   c. FitnessScore = total time elapsed / number of levels.

   Based on your fitness function, you can use any objective for the algorithms during the gameplay, not just maximizing the score. Examples of objectives include finishing each level as fast as possible, eating as many ghosts as possible, or playing for as long as possible.

7. **Report your findings in a Word file**. It should contain the following:
   a. Cover page: course code, Assignment 1, name and matric numbers of your group members.
   b. Introduction: summary of the report. What is the objective of the assignment. What algorithms and fitness functions did you use.
   c. Methodology: describe the algorithms and the fitness functions in detail.
   d. Results: report your findings. Compare the performance of the two algorithms for controlling Pac-Man. Compare the performance of the fitness functions. **Please include screenshots of the final Pac Man score.**

e. Discussion and Conclusion: what can be concluded from the Results? Which method is better and why?

8. **Zip your codes and your report and submit via UM Spectrum by 1st December 2023**. If the files are too large, submit to my email: liew.wei.shiung.phd@um.edu.my.

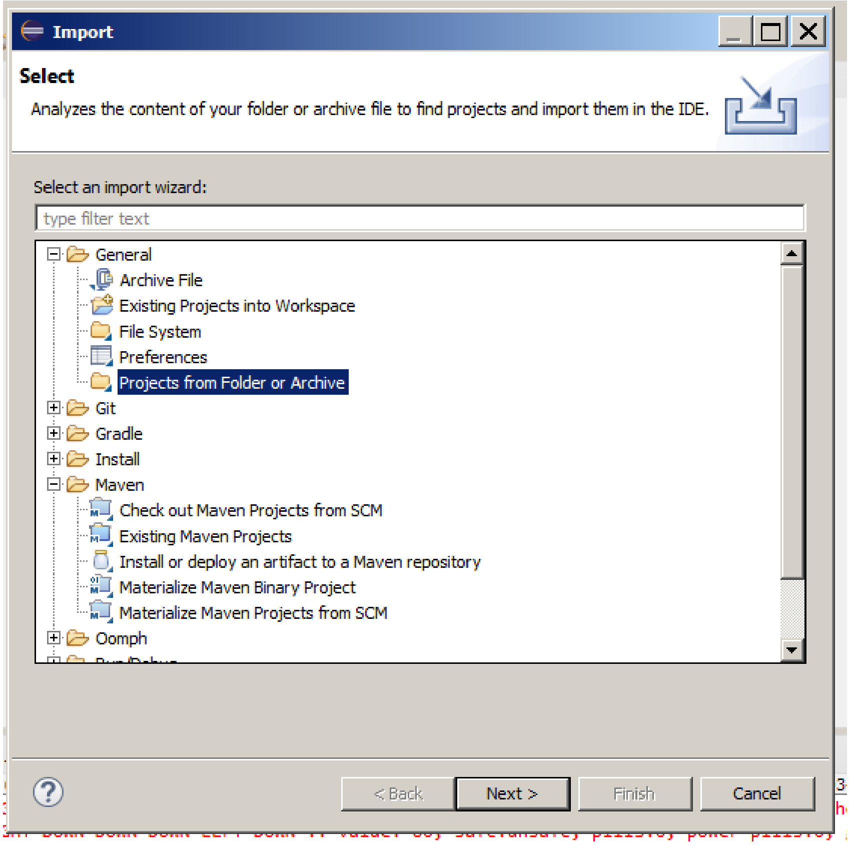
Marking scheme:

Criteria	Marks
Implemented a search algorithm for controlling the Pac-Man.	1 mark per search algorithm (max 3)
Implemented an evaluation metric to measure the performance of the search algorithm.	1 mark per evaluation metric (max 3)
Report Introduction and Methodology	2
Report Results	2
Report Discussion and Conclusion	2
Total	(max 10%)

Setup instructions:

1) Download and unzip the folder.

2) If using Eclipse Java, import the project from the folder.

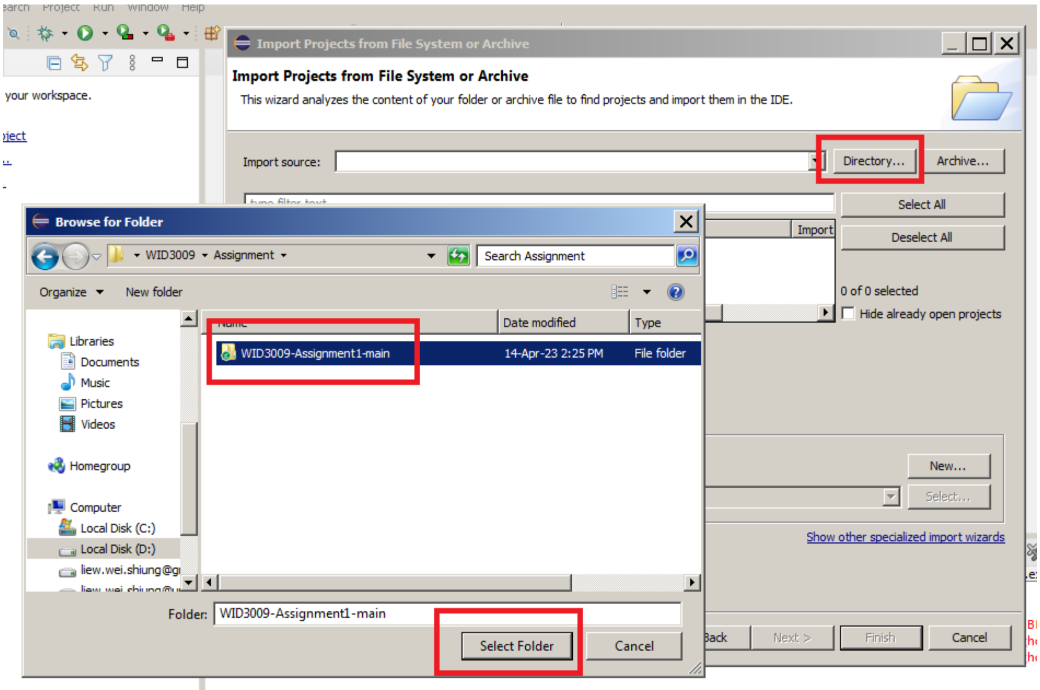


Select

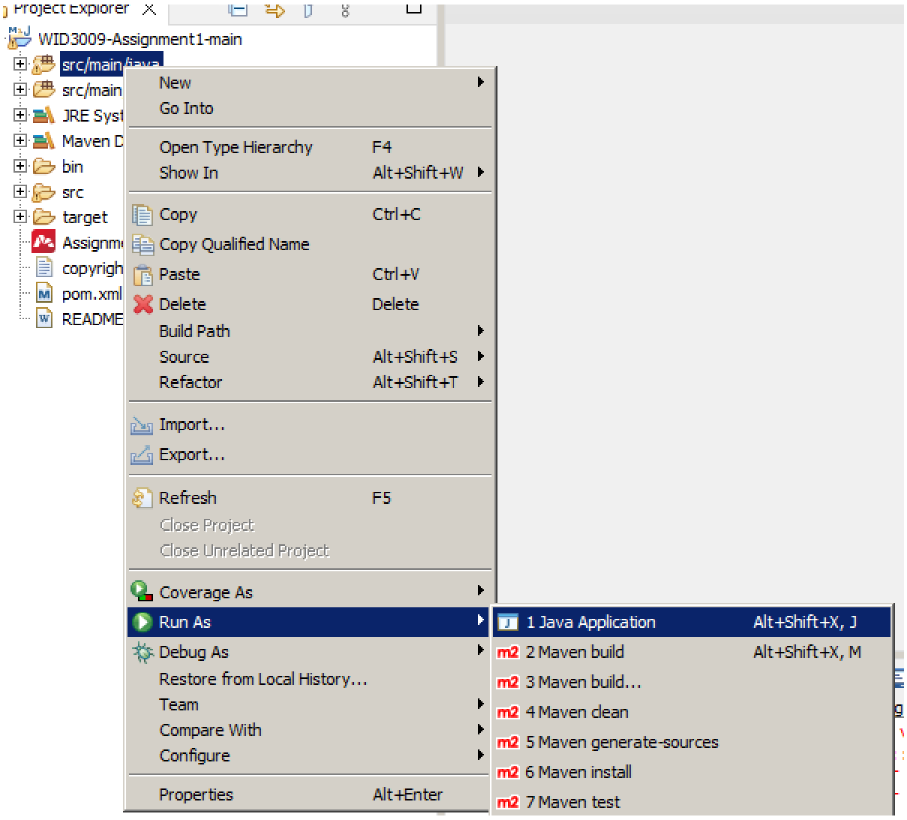
Analyzes the content of your folder or archive file to find projects and import them in the IDE.

Select an import wizard:

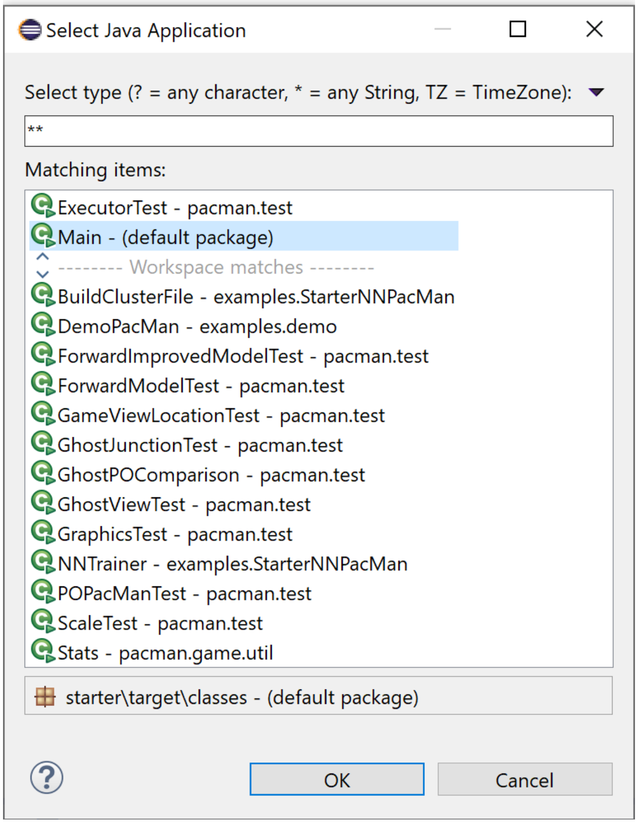
Import
type filter text
General
Archive File
Existing Projects into Workspace
File System
Preferences
Projects from Folder or Archive
Git
Gradle
Install
Maven
Check out Maven Projects from SCM
Existing Maven Projects
Install or deploy an artifact to a Maven repository
Materialize Maven Binary Project
Materialize Maven Projects from SCM
Oomph
< Back
Next >
Finish
Cancel



3) To run the file, right-click on the folder “src/main/java” and run as Java application.



4) Select Main – (default package) and click ok



Select Java Application

Select type (? = any character, * = any String, TZ = TimeZone):

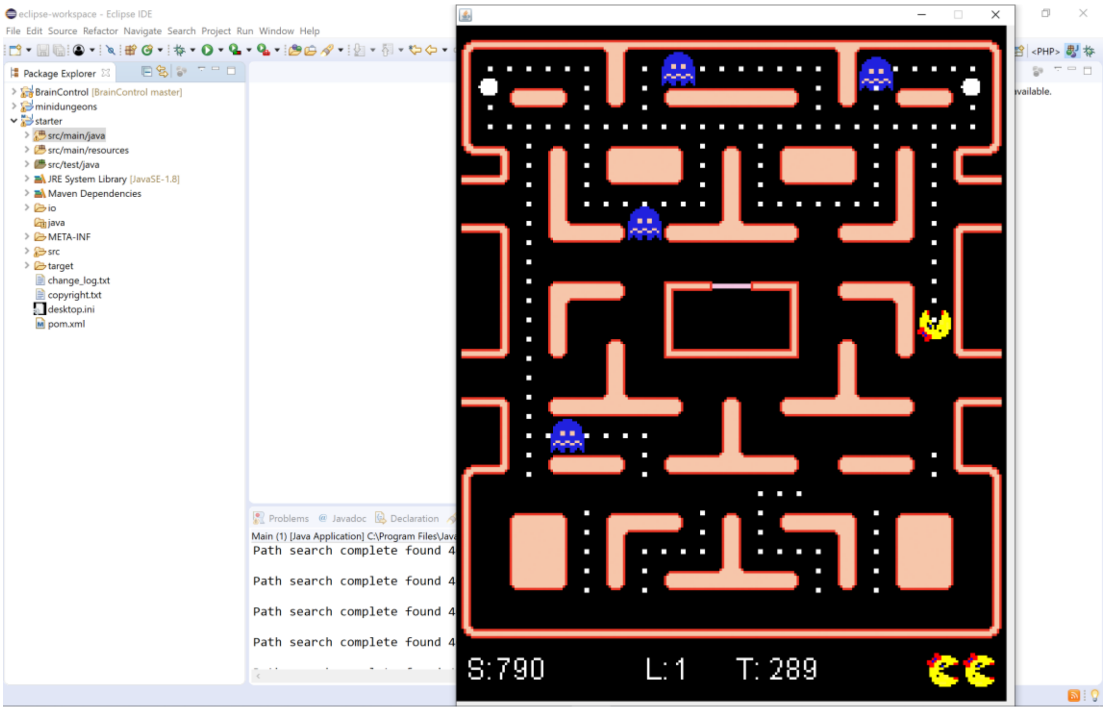
**

Matching items:

- ExecutorTest - pacman.test
- Main - (default package)
- -------- Workspace matches --------
- BuildClusterFile - examples.StarterNNPacMan
- DemoPacMan - examples.demo
- ForwardImprovedModelTest - pacman.test
- ForwardModelTest - pacman.test
- GameViewLocationTest - pacman.test
- GhostJunctionTest - pacman.test
- GhostPOComparison - pacman.test
- GhostViewTest - pacman.test
- GraphicsTest - pacman.test
- NNTrainer - examples.StarterNNPacMan
- POPacManTest - pacman.test
- ScaleTest - pacman.test
- Stats - pacman.game.util

starter\target\classes - (default package)

5) The game will run using default settings.



eclipse-workspace - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
Problems
Javadoc
Declaration
S:790
L:1
T:289

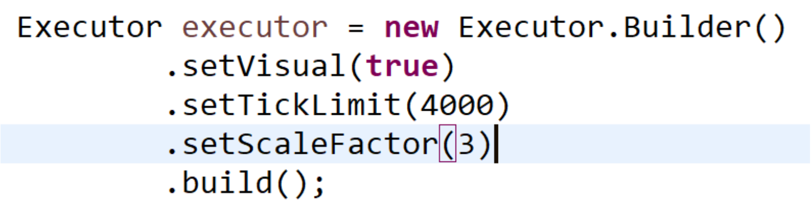
- BrainControl [BrainControl master]
- minidungeons
- starter
  - src/main/java
  - src/main/resources
  - src/test/java
  - JRE System Library [JavaSE-1.8]
  - Maven Dependencies
  - io
  - java
  - META-INF
  - src
  - target
  - change_log.txt
  - copyright.txt
  - desktop.ini
  - pom.xml

```
Main (1) [Java Application] C:\Program Files\Java
Path search complete found 4

Path search complete found 4

Path search complete found 4

Path search complete found 4
```

What to modify?

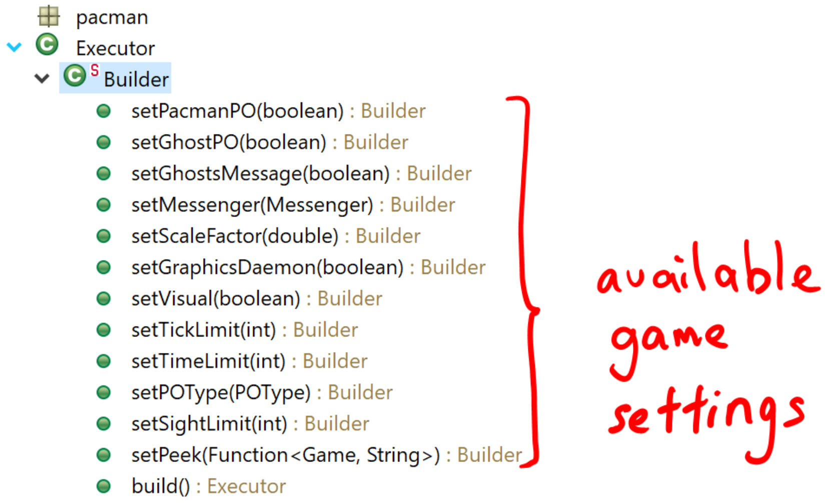
Open the Main.java file. This file is the entry point of the game which you can configure the game settings. You can change the scale factor, sight limit, whether to on or off visuals in this file.

For example, to increase the size of game visual, we set scale factor to 3.



```
Executor executor = new Executor.Builder()
        .setVisual(true)
        .setTickLimit(4000)
        .setScaleFactor(3)
        .build();
```

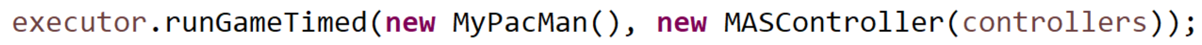
Here are the list of available game settings that you can modify:



```
pacman
Executor
  Builder
    setPacmanPO(boolean) : Builder
    setGhostPO(boolean) : Builder
    setGhostsMessage(boolean) : Builder
    setMessenger(Messenger) : Builder
    setScaleFactor(double) : Builder
    setGraphicsDaemon(boolean) : Builder
    setVisual(boolean) : Builder
    setTickLimit(int) : Builder
    setTimeLimit(int) : Builder
    setPOType(POType) : Builder
    setSightLimit(int) : Builder
    setPeek(Function<Game, String>) : Builder
    build() : Executor
```

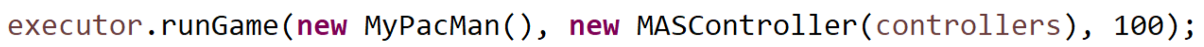
available game settings

Next, to remove the time limit imposed on the game, we change the last line of the file from:



```
executor.runGameTimed(new MyPacMan(), new MASController(controllers));
```

To



```
executor.runGame(new MyPacMan(), new MASController(controllers), 100);
```

Notice the 3rd argument is 100, which indicates 0.1s delay in each time-step in order to reduce the speed of the game for us to see.

Finally, you can modify the **MyPacMan** class to implement your AI method. The class should contain a getMove method that returns a single MOVE (LEFT, RIGHT, TOP, DOWN) for each time step.