Introduction to Practical Programming with Objects

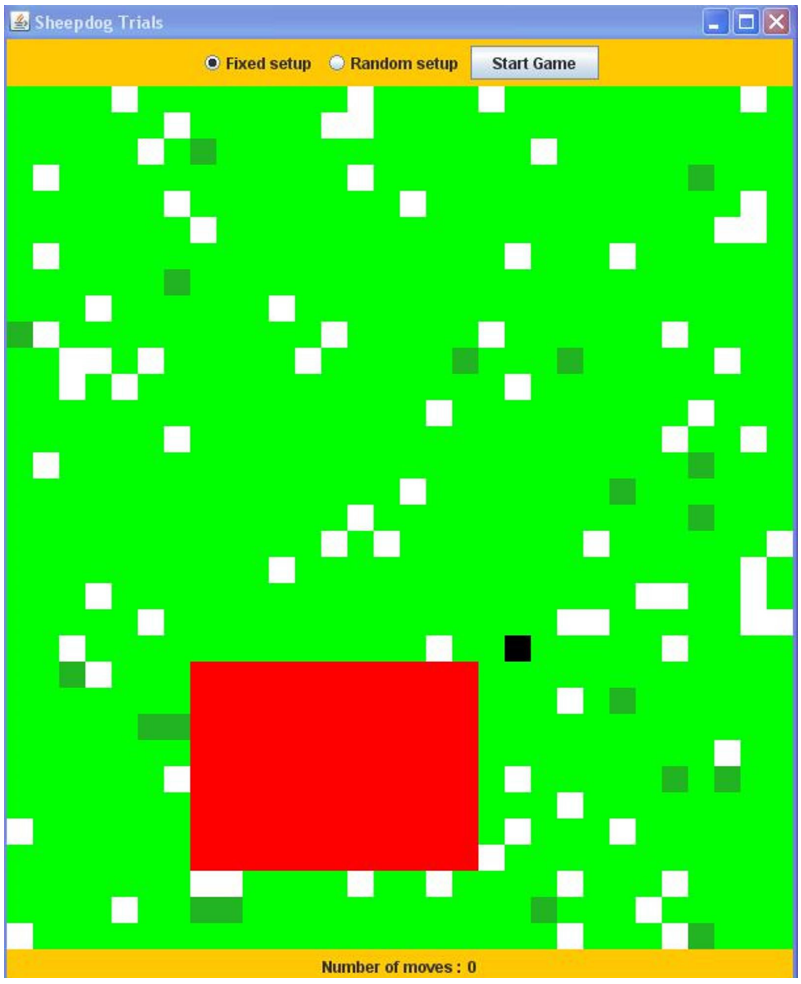
Assignment 2 - Sheepdog Trials

The University of Edinburgh
David Symons

Overview

This assignment is intended to give you practice in designing a program from the ground up. Your objective is to design and implement a simple Java game called "Sheepdog Trials". In this grid-based game, the player controls a dog among a heard of sheep. The sheep run away from the dog, which allows them to be chased into a pen. The game is won when all sheep are in the pen.

An example screenshot from the finished game is shown below. Empty patches of grass are shown as light green. The dog is represented by a black square and sheep by white squares. The pen area is red. The dark green squares are bushes (optional feature).



Marks will be awarded out of 100 and count for 60% of your overall grade for IPPO. Completion time may vary depending on your experience and the grade you are aiming for. Of the 100 hours the course is designed to take, 15 have been allotted for this assignment. This is the estimated time for completing the basic and intermediate tasks. Marks above 69% are expected to be rare and require you to tackle the advanced section, which goes beyond the course in terms of content and time.

Before you start, please read the following sections:

- *Restrictions*
- *Good Scholarly Practice*
- *Submission, late submission & extensions*
- *Marking Criteria*

Restrictions

Now that you have some experience using basic Java, you are encouraged to use the Java standard library. This means you can import classes such as *java.util.ArrayList* etc.

External libraries are not permitted.

If you have had time to look at the advanced chapters, you can use lambdas and streams. I still do not recommend this, however, as this course is mainly designed to teach imperative programming.

Good Scholarly Practice

Please remember the good scholarly practice requirements of the University regarding work for credit. You can find guidance at the School page, which also links to the relevant University pages.
https://web.inf.ed.ac.uk/infweb/admin/policies/academic-misconduct

Submission, late submission & extensions

Submission instructions will be provided separately on LEARN → Assessment.

The submission deadline is Friday 2nd December 2022 at 12:00.

Extension rule 3 applies to this assignment. Extensions and ETA can be no longer than 6 days.
More information regarding late submission is available from LEARN → Assessment.

Marking Criteria

Marks will be assigned in accordance with the University's Common Marking Scheme (CMS).
See: https://web.inf.ed.ac.uk/infweb/student-services/ito/students/common-marking-scheme

Tasks are grouped into basic, intermediate and advanced sections.
The basic requirements correspond to the description of a pass grade (up to 49%).
Intermediate features are required to demonstrated the mastery required for a 2nd class (up to 69%).
You will need to tackle the advanced section for a distinction (up to 100%).

Please note that attempting the more difficult features only removes grade ceilings. It does not guarantee you a mark in the corresponding range. You still need to fulfil all requirements for the lower categories as well!

This assignment focuses on program design. The program you develop must show evidence of a well considered, modular and object-oriented design. A solution using only one (or very few) classes may be possible, but is insufficient for a pass grade even if the program works! You are assessed on the design, your written justification thereof, the quality of the code you submit and functional correctness of the implemented features.

Before submitting, you should try to verify the correctness of your code. You can do this either by running the program or writing (JUnit) test cases. Testing is not assessed, but will allow you to catch mistakes and may lead to a better grade indirectly.

Setup
For this assignment, you are starting with a blank sheet. Just create a new project in your preferred IDE and experiment with your own designs.

Basic requirements
This section describes the features required for a basic implementation of Sheepdog Trials. A good solution will allow you to pass without attempting any of the intermediate or advanced features. Doing so risks a low mark, however, as unanticipated errors can mean a mark of less than 40%.

Required features
- A representation of the state of the game. The fixed setup is a 8x8 grid with one dog, 5 sheep and a pen area of size 2x3 (or 3x2). No bushes.
- The ability to control the dog by by entering commands on the terminal. To move the dog up, enter 'w' and press enter. Use 's' for down, 'a' for left and 'd' for right.
    - Sheep do not move yet (this is an intermediate feature).
- A text-based UI (user interface) showing the game state, which updates when the dog moves.
- The dog cannot go out of bounds, into the pen or onto a square occupied by a sheep.

Write a short report (minimum 1 page, maximum 2 pages) covering the following:
- An explanation of your program design mentioning which classes you defined and how they are related. You can include a class diagram if you like.
- Describe the flow of execution i.e. how the different components of your program communicate.
- Justify why you think this is a good design.

Intermediate features
Once you are satisfied that you have all basic features working, you can enhance your game by adding the functionality described here. This will allow you to obtain a mark of up to 69% provided you have a good solution to all basic tasks.

Required features
- The user can choose between random and fixed setup at start-up. For the random setup, the following are chosen at random: grid size, pen size and position, number and location of sheep. However, make sure that the game is playable.
- Sheep move away from the dog as it approaches. Dogs and sheep do not move out of bounds.
- When a sheep enters the pen area it will stay in the pen. It does not matter which part of the pen the sheep enters i.e. the squares of the pen are never counted as “occupied”.
- When the game is over, display a “victory” message and offer the user to restart the game.
- Display the number of moves made by the player so far.

Advanced features
Tasks described in this section are much more difficult and only required for a distinction.

Required features
- Add a graphical view of the game (GUI). You can use AWT/Swing or JavaFx.
- Provide sheep behaviour such as flocking when the dog is absent and reluctance to enter pen.
- Use a proper design pattern that allows text and GUI views to be shown at the same time