CS5001 Object-Oriented Modelling, Design and Programming
Practical 2 – Animal Chess

School of Computer Science
University of St Andrews

Due Friday week 5, weighting 35%
MMS is the definitive source for deadlines and weightings.

In this practical, you will be provided with an object-oriented model, and your task will be to implement it in Java. The model is specified by two things: a UML class diagram showing the required classes, and a suite of tests that define how your program should behave. The details of implementation are up to you: after creating the classes in Java, you will need to fill in their methods yourself, and decide on any additional fields, methods, classes etc. that you might need in order to meet the requirements.

For this practical, you may develop your code in the IDE or text editor of your choice, but you must ensure all your source code is in a folder named `CS5001-p2-animal-chess/src/animalchess`, where `animalchess` corresponds to the package name required by the specification.

Animal Chess

The game of chess reached Japan from India around one thousand years ago, and over the centuries it developed rather differently in Japan from the rest of the world. The game's focus switched towards pieces that move only one square at a time, and curious innovations arose, including a more complex system of promotions, and the ability to "drop" a captured piece onto the board with its colour changed. The most popular chess variant in Japan today is known as *Shogi* (or *The Game of Generals*); it has gained popularity over the years, and now has a significant number of players outside the country.

The subject of this practical is a variation of *Goro-Goro Dobutsu Shogi* (or *Purring Animal Chess*), a small Shogi variant designed by professional Shogi player Madoka Kitao, her aim being to introduce children to Shogi. The game is played on a 5×6 board, and features four different animals which all move in different ways, with similarities to pieces found in chess. In this practical we will use rules that are a little different from those designed by Kitao, so read the following rules carefully.

Game Rules

The two players sit facing each other, with eight pieces each: a lion, two giraffes, two cats and three chicks. Pieces are oriented on the board so that they appear "face up" to the player that owns them. Players take turns moving one of their own pieces.

- A chick moves one space forward.
- A giraffe moves one space forward, backward, left or right (4 directions).
- A cat moves one space in any diagonal direction, or directly forward (5 directions).
- A lion moves one space in any direction, either orthogonal or diagonal (8 directions).

These directions are shown on the playing pieces with red dots.

Figure 1: The board at the start of a game

A player cannot move a piece onto one of their own pieces, but they may move onto one of their opponent's pieces, in which case the opponent's piece is **captured**. The capturing player takes the piece into their **hand**, and on a future turn, instead of moving a piece, they may **drop** a captured piece, by placing it onto any empty space on the board as one of their own pieces. A player wins by capturing their opponent's lion.

The chick and cat pieces are **promotable**. If a player moves a chick or a cat into the farthest two

rows from them (the two rows closest to their opponent) then it promotes. In a real game, this is shown by flipping the piece over to show a different picture on the other side (a promoted chick looks like a rooster, and a promoted cat has a fish!).

A piece that has been promoted moves one space forward, backward, left or right, or diagonally forward (6 directions).

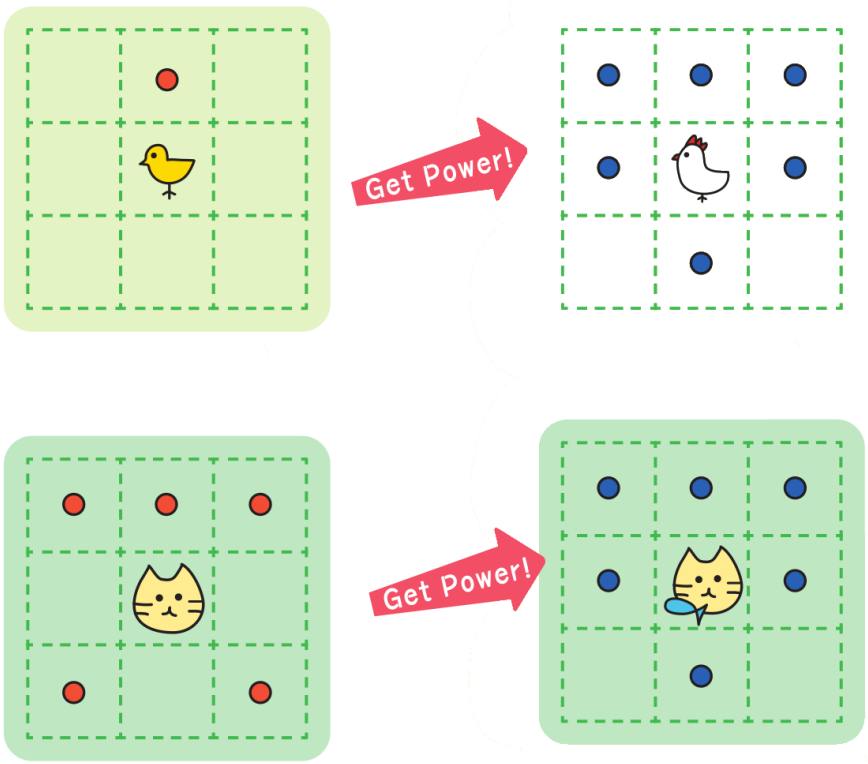

Figure 2: How chicks and cats move after promotion

Note the following important rules:

- There is no such thing as *check* or *checkmate* as in chess. You win simply by capturing your opponent's lion.
- There is no rule about endless repetition as in chess. A game can go on forever.
- When a promotable piece enters the promotion zone, it **must** promote.
- When a promotable piece is captured, it "un-promotes", returning to its original form.

Remember: the rules for this practical are as specified in this document and in the tests on StudRes. The rules are slightly different from similar games you can read about online.

System Specification

You are required to implement the classes shown in the following UML class diagram, including all the public methods and attributes shown. Your program should not require any input from the user, nor print any output to the screen. Your code can be tested by running the JUnit tests that are included with this practical. This can be done on the lab machines using *stacscheck*, by navigating into your `CS5001-p2-animal-chess/` directory and running the following command:

```
stacscheck /cs/studres/CS5001/Practicals/p2-animalchess/Tests
```

This will compile your classes and run all the provided tests on them. You can access *stacscheck* on the lab machines or via SSH, or if you use a Linux or Mac machine, you can install it on your own computer.

Important: The behaviour we want from the system is defined by these tests. If you're not absolutely sure what a class or method is supposed to do, you should look at the tests that correspond to it and examine the examples and assertions there. The relevant unit test files are in subdirectories of the `Tests/` directory, and they all have names of the form `<classname>Test.java`.

You can add to these by creating your own additional tests if you wish. You can run your own JUnit tests in an IDE, via the command line, or via *stacscheck*. If an aspect of the program's behaviour is not defined by the tests or the UML diagram, you should make your own choice about what the program should do, and document this in your code as a comment.

UML class diagram

The UML diagram included below shows the structure of the system you should build. You should include every class and every public method on the diagram, but you may implement more classes and methods if you wish, and the choice of private attributes and methods is up to you.

In this diagram, each yellow box is one class. The 'C' symbol in the top part refers to a class, while the 'A' symbol refers to an abstract class. `Chick`, `Giraffe`, `Cat` and `Lion` are all special types of `Piece`, so they inherit its public methods. `Chick` and `Cat` also inherit from `PromotablePiece`, which defines some special functionality for pieces that can promote. `PromotablePiece` and `Lion` show some methods marked with {*override*} – this is a hint that you might want to override these methods in these particular classes. Each `Piece` is owned by one `Player`, and may sit on one `Square` (if it is not in a player's hand). The whole `Game` has two `Players`, and 30 `Square`s (the 5×6 board shown above). When a player moves a piece onto a square, they can use the square's `isPromotionZone` method to test whether their piece is in the zone where it promotes (the farthest two ranks from the player).

Exception handling

An exception of the class `IllegalArgumentException` should be thrown if a player tries to drop a piece that is not in their hand, or if `placePiece` attempts to place a piece on a square that is already occupied. It may also be used in other situations where an operation cannot be carried out because of an illegal argument. A short message describing what went wrong should be included in the exception. Make sure to catch it and handle it where appropriate!

Packages

Also shown on the UML diagram is the name of the package into which your classes should be placed: `animalchess`. Your `.java` files should be placed in the corresponding subdirectory inside your `src/` directory. You should make sure to declare this package at the top of each file, with the line `package animalchess;`.

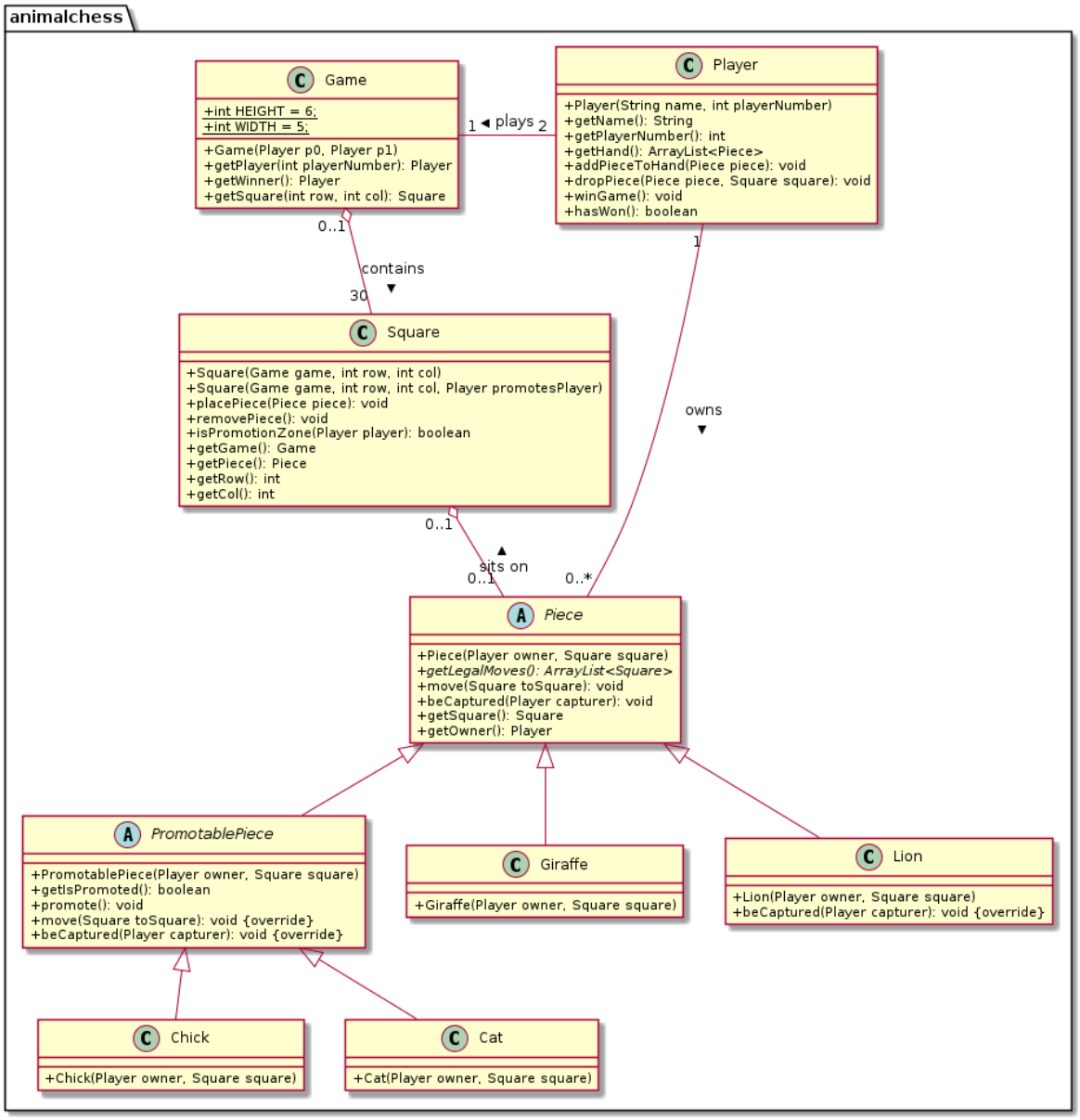

Figure 3: UML class diagram of the required system

Behaviour

A player has an attribute `playerNumber`, which should be either 0 or 1. Player 0 sits at the top of the board, and player 1 sits at the bottom of the board. Each Square has a position on the board defined by its `row` and `col` attributes: these should be numbers between 0 and 5, as shown by the `(row, col)` pairs in the following diagram.

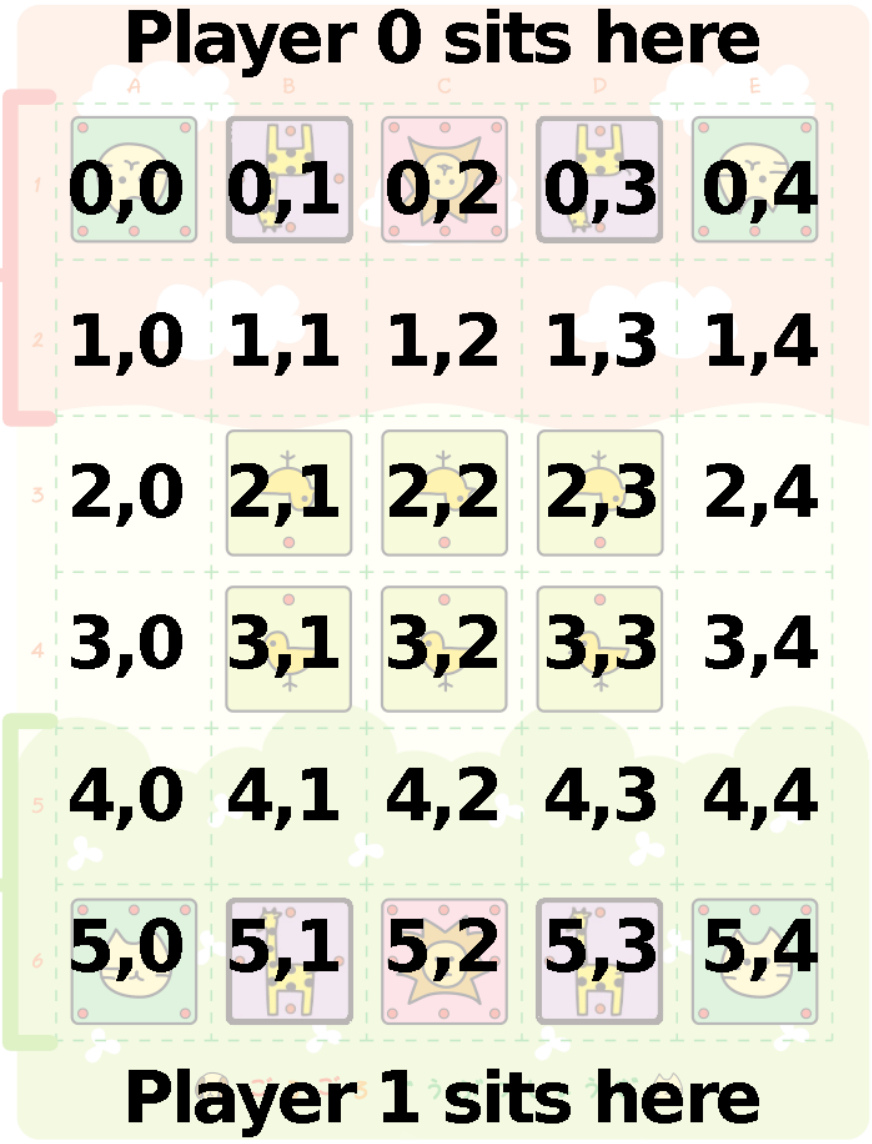

Figure 4: `(row, col)` coordinates on the board

Note: Certain events, such as a piece being promoted, or a player winning, are triggered by public methods. These events can be forced by calling the appropriate method – for example, `player.winGame()` – but should also be triggered according to the rules of the game. For example, a player should win when they take their opponent's lion.

Deliverables

A report is not necessary for this project, but if you have strayed from the specification at all, or if you have made any design decisions that you wish to explain, you can include a short readme file in your `CS5001-p2-animal-chess/` folder; this should include any necessary instructions for compiling or running your program. Hand in an archive of your assignment folder, including your `src/` directory and any local test subdirectories, via MMS as usual.

Marking

A submission that does not satisfy all the tests above may still receive a good grade. A submission that implements all classes and methods as shown on the UML diagram, using object-oriented methods, but which does not have the required behaviour in all cases, could receive a grade of up to 15 if it is implemented cleanly and intelligently.

A submission which satisfies all the requirements described, including passing all the tests, can achieve any grade up to 20, with the highest grades awarded to implementations that show clarity of design and implementation, with additional tests that show an insight into the problem.

Any extension activities that show insight into object-oriented design and test-driven development may increase your grade, but a good implementation of the stated requirements should be your priority.

See the standard mark descriptors in the School Student Handbook:

https://info.cs.st-andrews.ac.uk/student-handbook/learning-teaching/feedback.html#Mark_Descriptors

Lateness

The standard penalty for late submission applies (Scheme B: 1 mark per 8-hour period, or part thereof) as shown at:

https://info.cs.st-andrews.ac.uk/student-handbook/learning-teaching/assessment.html#lateness-penalties

Good Academic Practice

The University policy on Good Academic Practice applies:

https://www.st-andrews.ac.uk/students/rules/academicpractice/