COMP2396 - Assignment 2

Due: 26 Feb, 2018 23:59

Introduction

This assignment tests your understanding of **inheritance** and **polymorphism** in Java.

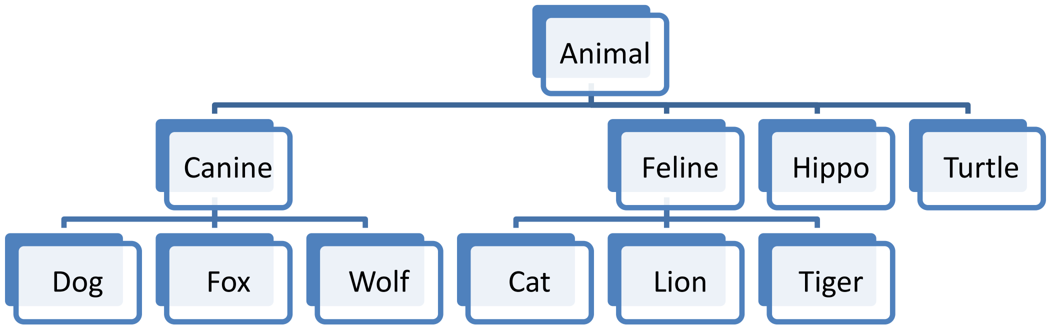
You are required to simulate the wild life in a forest. Program design **will be evaluated** in this assignment. You must make good use of **inheritance** and **polymorphism** to get all the marks for this assignment.

You are also required to write **JavaDoc** for all non-private classes and non-private class members. **Programs without JavaDoc will not be marked.**

Task

You need to implement the main program, `Forest.java`, which simulate a forest. A forest is represented by a 2D array of cells. Each cell can be used to hold **one animal** (an object of **Animal** class or a sub-class of **Animal** class).

Similar to the **Animal** class hierarchy in the lecture, you are required to implement a hierarchy of animals, as shown below:



When the program starts (i.e., the `main()` method inside `Forest.java`), it will create a forest of size **15x15**, populate it randomly with **8 animal objects**, namely a **Dog**, a **Fox**, a **Wolf**, a **Cat**, a **Lion**, a **Tiger,** a **Hippo** and a **Turtle**. Once the forest is generated, print the layout of the forest as follow:

```
..c............
...t...........
...............
............d..
...............
...............
..f............
.l.............
h..............
.....w.........
...............
..u............
```

A **dot** indicates an empty space, and a letter indicate an animal at that location. **The first letter** of each of the 8 animals is used to label them except **Turtle** is labeled with '**u**'. If a location contains a dead animal body, it will be labeled with the upper case letter of the animal except dead Turtle is labeled with 'U'.

The program than ask the user **press enter** to run a cycle of simulation, or type "**exit**" to leave:

```
Press enter to iterate, type 'exit' to quit:
```

Refer to the tutorial slides on how it can be done.

In every cycle of simulation, all animals in the forest will take turns to move. If the target location is already occupied by another animal, the moving animal will perform an attack. For an animal moves 2 steps and if another animal exists on the path of movement, i.e. on the first step of the moving animal, they will fight. Either one of the animal involved will die at the end of the attack. The survived animal will take up the target location while the dead animal body will be moved to one of the nearby empty location with equal chance, e.g. if the target location has 6 empty nearby locations, the dead animal body will be moved to one of those locations with probability 1/6.

The program should ask for user input after every cycle. The program terminates only when user type "**exit**". When the program terminates, it will print out the list of living animals and their locations, followed by the list of dead animals and their locations.

Animal moving

In every cycle of simulation, animals will be moved in the following order: **Cat**, **Dog**, **Fox**, **Hippo**, **Lion**, **Tiger**, **Turtle**, **Wolf**.

Different animals move **randomly** in a different manner:

- **Feline** moves in all **eight directions**, **one** step a time.
- **Canine** moves in **four directions**, **one or two** steps a time.
- **Turtle** has 50% chance stay in the **same position**, and 50% chance move in **four directions**, one step at a time.
- All other animals move in **four directions**, **one** step a time.
- If the animal is located at the corner, it stays or move to the available positions with **equal chance**.

Note that the animals should **not** move out of the forest. When an animal moves from one location to another, your program should print the following information:

```
Animal_type moved from ?, ? to ?, ?
```

For example:

```
Fox moved from 2, 0 to 0, 0
Hippo moved from 3, 0 to 4, 0
...
```

Animal attacking

When an animal moves to a location that is occupied by another animal, the moving animal will **attack** the occupying animal **before moving**. The result of an attack follows the following rules:

- If a **Feline** attacks a **Canine**, **Feline** wins and **Canine** dies.
- If a **Canine** attacks a **Feline**, there is a 50% chance that one wins and the other dies.
- If a **Lion** attacks a **Hippo**, **Lion** wins and **Hippo** dies.
- If a **Fox** attacks a **Cat**, **Fox** wins and **Cat** dies.
- If any **Animal** attacks a **Turtle**, there is a 20% chance that the **Animal** wins and the **Turtle** dies.
- If a **Turtle** attacks an **Animal**, there is a 50% chance that the **Turtle** wins and the **Animal** dies.
- For all other cases, the **attacker loses and dies**.
- If the animal dies, it disappear and does not occupied the position.
- The animal may attack multiple times if multiple animal blocked its path.

When an animal attacks another animal, your program should print the following information:

```
Attacker_type from ?, ? attacks occupant_type at ?, ? and wins/loses
The Loser dies at ?, ?
```

The location of the dead animal should be the location of the dead body.

For example:

```
Tiger from 2, 1 attacks Cat at 2, 2 and loses
Tiger dies at 2, 1
Lion from 4, 6 attacks Hippo at 3, 5 and wins
Hippo dies at 2, 5
Lion moved from 4,6 to 3, 5
```

Note that an animal will move if attack is successful.

Sample run

A sample run is provided as `sampleRun.txt` and is available on Moodle.

Marking

- **40% marks** are given to the **program design.**
  - You will be awarded all the marks if you are implementing the **move** and **attack** of the animals by making use of **inheritance** and **polymorphism**.
  - You can check it by avoiding code duplication as much as possible.
  - **Economy is valuable in coding: the easiest way to ensure a bug-free line of code is not to write the line of code at all.**
- **40% marks** are given to the **functionality** of your program.
  - You may add **additional classes, instant variables and methods** to the class.
  - Your program output must be **identical** to what is described in this document, with the exception of the trailing spaces at the end of each line of output.

- **20% marks** are given to your **JavaDoc**. A complete JavaDoc includes documentation of every classes, member fields and methods that are not private. JavaDoc for the main method may be omitted.

Submission:

Please submit all source files (`*.java`) in a single compressed file (in `.zip` or `.7z`) to Moodle. **Late submission is not allowed**.

Do not submit .*class* file.

-- END --