Com S 228 Spring 2022

Project 1: Wildlife of the Plain

Due at **11:59pm, Thursday, Feb 24**

1. Problem Description

This project simulates interactions among different forms of life in a plain. The plain is represented by an $N \times N$ grid that changes over a number of cycles. Within a cycle, each square is occupied by one of the following five life forms:

Badger (`B`), Fox (`F`), Rabbit (`R`), Grass (`G`), and Empty (`E`)

An Empty square means that it is not occupied by any life form.

Below is a plain example as a 6×6 grid.

```
F5 E  E  F0 E  E
B3 F1 B0 R0 G  R0
R0 E  R2 B0 B2 G
B0 E  E  R1 F0 E
B1 E  E  G  E  R0
G  G  E  B0 R2 E
```

Both row and column indices start from 0. In the example, the (1, 1)th square is occupied by a 1-year-old Fox. It has a 3×3 neighborhood centered at the square:

```
F5 E  E
B3 F1 B0
R0 E  R2
```

The (0, 0)th square `F5` (a 5-year-old Fox) has only a 2×2 neighborhood:

```
F5 E
B3 F1
```

Meanwhile, the (2, 0)th square `R0` (a newborn Rabbit) has a 3×2 neighborhood:

```
B3 F1
R0 E
B0 E
```

Generally, the ***neighborhood*** of a square is a "3×3" grid which includes only those squares lying within the intersection of the plain with a 3×3 window centered at the square. When a square is on the border, the dimension of its neighborhood reduces to 2×3, 3×2, or 2×2.

2. Survival Rules

The plain evolves from one cycle to the next one. In the next cycle, the life form to reside on a square is decided from those life forms in the ***current cycle*** who live in the 3×3 neighborhood centered at the square, under a set of survival rules. These rules are specified according to the life form residing on the same square in the current cycle. Badgers, foxes, and rabbits start at age 0, and grow one year older when the next cycle starts.

2.1 Badger

A badger dies of old age or hunger, or from a group attack by foxes when it is alone. The life form on a Badger square in the next cycle will be

a) Empty if the Badger is currently at age 4;
b) otherwise, Fox, if there is only one Badger but there are more than one Fox in the neighborhood;
c) otherwise, Empty, if Badgers and Foxes together outnumber Rabbits in the neighborhood;
d) otherwise, Badger (the badger will live on).

The new life form taking over the square, if a Fox, will have age 0 when the next cycle starts.

For example, in the following neighborhood of a Badger at age 2:

```
R0 G  R0
B0 B2 G
R1 F0 E
```

there are two Badgers (including self), one Fox, three Rabbits. Going down the rule list, neither a), b), c) applies. According to rule d), the central element (square) will still be `B` (Badger) --- just one year older --- in the next cycle. In other words, `B2` will be replaced with `B3`.

2.2 Fox

A fox dies of old age, hunger, or an attack by more numerous badgers. The life form on a Fox square in the next cycle will be

a) Empty if the Fox is currently at age 6;
b) otherwise, Badger, if there are more Badgers than Foxes in the neighborhood;
c) otherwise, Empty, if Badgers and Foxes together outnumber Rabbits in the neighborhood;
d) otherwise, Fox (the fox will live on).

The new life form, if a Badger, will have age 0 when the next cycle begins.

For example, in the following neighborhood of a Fox at age 1:

```
F5 E  E
B3 F1 B0
R0 E  R2
```

there are two Foxes, two Badgers, and two Rabbits. Rule c) applies. The central square will become `E` in the next cycle.

2.3 Rabbit

A rabbit dies of old age or hunger. It may also be eaten by a badger or a fox. More specifically, the life form on a Rabbit square in the next cycle will be

a) Empty if the Rabbit's current age is 3;
b) otherwise, Empty if there is no Grass in the neighborhood (the rabbit needs food);
c) otherwise, Fox if in the neighborhood there are at least as many Foxes and Badgers combined as Rabbits, and furthermore, if there are more Foxes than Badgers;
d) otherwise, Badger if there are more Badgers than Rabbits in the neighborhood;
e) otherwise, Rabbit (the rabbit will live on).

If the new life form is a Badger or Fox, it will have age 0 when the next cycle starts.

In the following neighborhood of a rabbit at age 2:

```
F1 B0 R0
E  R2 B0
E  E  R1
```

lives two Badgers, one Fox, and three Rabbits. Rule a) does not apply because the Rabbit is only 2-years old. Rule b) does since there is no Grass in the neighborhood. The central element (square) will be `E` in the next cycle according to this rule.

2.4 Grass

Grass may be eaten out by overcrowded rabbits. Rabbits may also multiply fast enough to take over the Grass square. In the next cycle, the life form on a Grass square will be

a) Empty if at least three times as many Rabbits as Grasses in the neighborhood;
b) otherwise, Rabbit if there are at least three Rabbits in the neighborhood;
c) otherwise, Grass.

If the new life form is a Rabbit, it will be aged 0 when the next cycle starts.

For example, if the neighborhood of a Grass is

```
F0 E  E
R0 G  R0
B0 B2 G
```

the central element will be `G` in the next cycle under rule c).

2.5 Empty

Empty squares are competed by other life forms. More specifically, the life form on an Empty square in the next cycle will be

a) Rabbit, if more than one neighboring Rabbit;
b) otherwise, Fox, if more than one neighboring Fox;
c) otherwise, Badger, if more than one neighboring Badger;
d) otherwise, Grass, if at least one neighboring Grass;
e) otherwise, Empty.

If the new life form is a Badger, Fox, or Rabbit, it will have age 0 when the next cycle begins.

For example, an Empty square in the top row has the following neighborhood:

```
F0 E  E
R0 G  R0
```

which includes two Rabbits. Thus, rule a) applies to change the central element to `R0` in the next cycle.

3. Task

You will implement an abstract class `Living` to represent a generic life form. It has three subclasses `Animal`, `Empty`, and `Grass`. The first subclass, implementing the interface `MyAge`, is abstract, and needs to be extended to three subclasses: `Badger`, `Fox`, and `Rabbit`. You also need to implement a `Plain` class which has a public member `Living[][]` to represent a grid plain.

The class `Wildlife` repeatedly simulates evolutions of input plains, either randomly generated or read from files. In each iteration, it interacts with the user who chooses how the plain will be generated, and enters the number of cycles to simulate. The iteration prints out the initial plain and the final plain.

Your random plain generator may follow the uniform probability distribution so that `Badger`, `Empty`, `Fox`, `Grass`, and `Rabbit` have equal likelihoods to occupy every square. Or you may use a different distribution, as long as no life form has zero chance to appear on a square.

Java provides a random number generator. To use it, you need to import the package `java.util.Random`. Next, declare and initiate a Random object like below

```
Random generator = new Random();
```

Then, every call below

```
generator.nextInt(5)
```

will generate a random number between 0 and 4 that corresponds to one of the five life forms.

Plain creation from an input file will weigh more than random creation in our grading. When zero or a negative number of cycles is entered by the user, your code does nothing but waits for a positive input.

A new living form, if a Badger, Fox, or Rabbit, has age 0 at its creation, whether initially by the class `plain` or later on under a survival rule. After surviving a cycle, its age increases by one.

Templates are provided for all classes. Be sure to use the package name `edu.iastate.cs228.hw1` for the project.

Below is a sample simulation scenario over three initial plains. In the first iteration, the user entered 1 for a randomly generated plain, 3 to specify the grid to be 3×3, and 1 to simulate just one cycle. The simulator printed out the initial and final plains. The second iteration simulated a randomly generated 6×6 grid over 8 cycles. In the third iteration, the user typed 2 for a file input, entered the file name "`public3-10x10.txt`", and specified 6 cycles. (The file `public3-10x10.txt` resides in the same folder containing the `src` folder.) After the third iteration, the user typed 3 to end the simulation. (Any number other than 1 and 2 could have ended the simulation.)

```
Simulation of Wildlife of the Plain
keys: 1 (random plain)  2 (file input)  3 (exit)

Trial 1: 1
Random plain
Enter grid width: 3
Enter the number of cycles: 1

Initial plain:

R0 R0 B0
G  G  E
G  E  G

Final plain:

R1 R1 B1
G  G  G
G  G  G

Trial 2: 1
Random plain
Enter grid width: 6
Enter the number of cycles: 8

Initial plain:

E  E  G  R0 B0 G
G  G  R0 B0 F0 R0
G  E  E  R0 G  E
R0 G  R0 R0 B0 R0
F0 E  R0 G  R0 F0
R0 R0 F0 F0 G  F0
```

```
Final plain:
R0 E  R0 E  B3 G
E  E  E  R0 R1 R3
R0 R0 R0 E  E  R0
E  E  E  E  R0 G
R0 E  E  R0 G  G
E  R0 R0 F1 G  G

Trial 3: 2
Plain input from a file
File name: public3-10x10.txt
Enter the number of cycles: 6

Initial plain:

B0 E  B0 E  B0 R0 E  R3 E  G
G  E  B0 E  F0 R0 E  B4 G  G
G  G  G  G  E  E  R0 E  G  G
F0 E  G  G  E  R0 R0 B0 B0 G
F0 F1 E  E  E  E  E  E  B0 E
G  G  R1 R0 R0 R0 R0 B0 B0 E
E  G  R0 R1 R2 R2 G  E  G  G
B0 B0 G  R0 R0 R0 G  B0 E  G
E  G  G  F4 R2 R0 E  G  G  G
G  G  E  E  E  G  G  G  G  G

Final plain:

B0 E  B0 E  E  R0 E  R0 R2 E
G  E  B0 R0 B4 R0 E  R0 R3 R1
G  G  R2 R0 E  R0 R0 E  E  E
G  F5 R3 R0 E  R0 R0 E  E  R0
R2 E  E  E  R0 R0 E  E  B1 G
R0 R0 R0 R0 R0 E  B1 R0 G  G
E  E  R0 E  R0 E  B1 R0 G  G
B4 E  R0 E  R0 E  E  E  R0 G
G  R2 R3 E  R0 E  R0 R3 R1 G
G  R0 R1 E  R0 E  R0 R2 R0 G

Trial 4: 3
```

Your code should print out the same text messages for user interactions.

4. Input/Output Format

The format for plains is shown in the sample runs above. Every square occupies two spaces starting with one of the letters 'B', 'E', 'F', 'G', and 'R'. If the letter is 'B', 'F', or 'R', then it is followed by a digit representing the animal's age; otherwise, it is followed by a blank. There is exactly one blank between two squares, whether represented by a letter and a digit, or a letter and a blank. No blank lines.

You may assume all input files to be ***correctly formatted***, containing `‘B’`, `'E'`, `'F'`, `'G'`, `'R'`, and digits up to 6 as the only non-blank characters. ***No*** digit will exceed the lifespan of the animal represented by its preceding letter.

5. Junit Classes

JUnit classes include `AnimalTest`, `BadgerTest`, `EmptyTest`, `FoxTest`, `GrassTest`, `LivingTest`, `PlainTest`, `RabbitTest`, and `WildlifeTest`. ***You need to implement all these classes.***

6. Submission

Write your classes and JUnit test classes in the `edu.iastate.cs228.hw1` package. Also submit the text files with your JUnit tests. Turn in the zip file ***not your class files***. Please follow the following guideline:

0. Include the Javadoc tag @author in each class .java file.

1. Navigate to your project directory and find the source files. By default, Eclipse stores these files in a directory called “src” below the project directory.

2. Select the top-level directories and/or files to be included in the archive. In most cases, all you need is the top-level directory “edu.”

3. Right-click anywhere in the selection and select Send To → Compressed/zipped file.

4. Rename the zip file to **Firstname_Lastname_HW1.zip**.

5. Open your zip file and check that its contents are correct (you have included all .java files and no .class files).