American University of Armenia
CS 120 Intro to OOP
Spring 2019

Homework Assignment 4, Part 2

In Part 1 of this homework, you produced an `ArrayLife` class that used a 2D array for representing a Game of Life board. That representation is quite general because it can be used to represents boards of any size (within sensible limits). At the same time, its memory usage is not very efficient. It consumes a **byte** of memory to store the state of each cell that is represented as a `boolean` value.

Let us notice that a **bit** of memory would be sufficient to store the state of one cell. For example, the value 0 may correspond to the state *dead* and the value 1 to the state *alive*. Thus, we can store the states of up to 64 cells using only **8 bytes = 64 bits** of memory.

1. **(35 points)** Produce a `PackedLife` class that represents a Game of Life where the underlying board is represented by a single variable of type `long`. This class should be fit for boards of size 8-by-8 or smaller. Thus, your class should look like:

```
public class PackedLife {

  private int width;
  private int height;
  private long world;
  private Pattern pattern;

  // ...
}
```

   Your class should pack the 0/1 state values of the cells into the different bits of the `long` variable named `world` according to the following rules. Let us assume that the board has size 6-by-6, for example. Then the state of the cell at position $(0, 0)$ should be stored in the **least significant bit** of `world`: the one corresponding to 2^0 and determining whether the number is even or odd. The state of the cell at position $(1, 0)$ (column 1 of row 0) should be stored in the second least significant bit corresponding to 2^1. The state of the cell at position $(5, 0)$ (column 5 of row 0) would then be stored in the bit corresponding to 2^5, position $(0, 1)$ (column 0 of row 1) to 2^6, position $(1, 1)$ to 2^7, etc.

   Hint: The expression `1L << (row * width + col)` evaluates to the corresponding power of 2 for the position `(col, row)`.

   Similar to the class `ArrayLife` from Part 1, your class should include the constructor and the methods with the following headings:

   - `PackedLife(String format)`: constructor that takes the format string as a parameter
   - `boolean getCell(int col, int row)`: accessor for a specific cell
   - `void setCell(int col, int row, boolean value)`: mutator for a specific cell
   - `void print()`: printing the state of the whole board
   - `int countNeighbours(int col, int row)`: counting the number of neighbours alive

- `boolean computeCell(int col, int row)`: determining if the cell will be alive or dead in the next generation, based on the rules of the game
- `void nextGeneration()`: updating the game board to the next generation
- `void play()`: while the user inputs the character `'s'`, printing the game board and advancing to the next generation; stops when the user inputs `'q'`

For instance, the `getCell` method implementation may look like:

```
public boolean getCell(int col, int row) {
  if (row < 0 || row >= height) {
    return false;
  }
  if (col < 0 || col >= width) {
    return false;
  }

  if (((world >>> (row * width + col)) & 1L) == 1L)
    return true;
  else
    return false;
}
```

Note that your methods may need to invoke each other or you may need to define some extra methods. For each of the methods, determine and add the correct access modifier and explain your choices in a brief report (1–2 paragraphs).

2. **(5 points)** In your report, explain in what situations you would use the `ArrayLife` class for the representation of the Game of Life and in what situations you would use the `PackedLife` class. List the advantages and disadvantages of both classes.

3. **(10 points)** The `initialise` method in the class `Pattern` from Part 1, works with an array of `boolean`s thus cannot be used with the class `PackedLife`. Overload this method to initalise a variable of type `long`:

```
public long initialise() {
  //TODO: produce a value of type long representing the state of
  //      'world' as expressed by the contents of the field 'cells'.
}
```

In your report, explain why the overloaded method cannot have the heading

`public void initialise(long world)`

but rather has the heading

`public long initialise()`.

Add a `main` method to your `PackedLife` class and test it.