# Assignment 5 Snarks and Grumpkins

Due Date: Wednesday Dec 9
Weight: 6%
This assignment should be completed individually.

## Description

*Snarks* and *grumpkins* are, possibly mythical, creatures that live beyond The Wall. Snarks prey upon grumpkins. Your task is to create a 2 dimensional simulation of snarks and grumpkins and to observe how the population of each fluctuates over time.

These creatures live in a square, fixed-size grid world ($20 \times 20$ is a good size). Each cell in the grid can be unoccupied or occupied by either a single snark or a single grumpkin.

Time passes in this world in discrete steps. Each creature performs some series of actions every time step. Actions include moving, eating other other and breeding, creating baby creatures.

During a time step, a creature may move to an adjacent cell. In this world, adjacent mean to the left, right, above or below, no diagonals. The edge of the world is like a giant wall. If they try to move past it they are blocked and remain in the same cell.

This system is an example of two ideas. One is simulation of predator-prey populations. Population levels fluctuate in a non-linear fashion so they are interesting to study. This system is also what is known as a cellular automata. These are 1, 2 or 3 dimensional automata that march through time in discrete steps according to well defined transition rules. A fascinating, and probably the best known, example of 2D cellular automoata is Conway's Game of Life http://www.bitstorm.org/gameoflife/. I spent many hours avoiding my Phd thesis trying out new Life patterns.

### *Grumpkins*

At each time step a grumpkin does two things, move, then breed, in that order.

1. *Move.* Select a random adjacent cell. If the target cell is occupied or is beyond the edge of the world, the grumpkin remains in place, other wise it moves to that cell.

2. *Breed*. If the grumpkin survives for three consecutive time steps, it is able to breed.

   A new grumpkin will be born in a randomly chosen empty adjacent cell.

   If there is no empty cell available, no breeding occurs.

   Once an offspring is produced, the grumpkin cannot produce an offspring until three more time steps have elapsed.

***Snarks***

Snarks are predators and have a slightly different behaviour. During each turn the snark will move and possibly eat, then possibly breed, then possibly starve, in that order.

1. *Eat*. If there a grumpkin in an adjacent cell the snark will move there and eat the grumpkin. If there is more than one grumpkin to eat, the snark will choose the first grumpkin it finds in this order: right, down, left, up. Snarks cannot eat other snarks.
2. *Move*. If the snark did not move to eat a grumpkin, it moves in the same fashion as the grumpkin does.
3. *Breed*. If a snark survives for eight time steps it will breed in the same fashion as the grumpkin.
4. *Starve*. If a snark has not eaten in the last three time steps, at the end of the third timestep it will starve and die.

Dead snarks or grumpkins are simply removed from the game.

Write a program to implement this simulation.

The grid world should be 20 x 20. When drawing the world, snarks should be represented using a Greek capital Delta (U+0398) and grumpkins should be drawn with a Greek capital Theta (U+0394).

Initially your world will have 5 snarks and 100 grumpkins, randomly placed on the grid.

You should prompt the user for how many steps to perform (default 1) before displaying the next configuration. At the bottom of each printout there should be a line giving the current number of snarks and grumpkins.

All of the snarks act before the grumpkins. Only one snark or grumpkin can act at a time. Go through the grid from top left to bottom right and have each snark perform its actions, then repeat for the grumpkins.

### Implementation Details

- I have provided an abstract class, *Creature* and an interface *Predator* that you must use, without editing.
- I have provided a first draft of an class *Game* to implement the grid world. Start with this and add code as needed.
- Random integers can be generated with Java's random number generator.

  ```
  import java.util.Random;

  ...

  Random rand = new Random();
  int x = rand.nextInt(10);
  ```

  `s` will be a random integer between 0 and 9, inclusive.
- Remember that this assignment is meant to be an example of using polymorphism so use polymorphism when you can.
- Unicode characters can be included in a Java string by entering the four digit hex code, escaped with a `\u`.

  e.g. `String str = "This is a Greek letter pi \u03C0";`

### What to Hand In

1. Rename your BlueJ project folder to `<LastName>_<FirstName>_Asg5`.
2. Submit your work to the COMP 1502 submit folder.

### Grading

A detailed grading rubric will be provided.

### Outcomes

Once you have completed this assignment you will have:

- Used inheritance via classes and interfaces;
- Used polymorphism;
- Practiced your object-oriented design skills;

- Used both the state and behaviour of objects;
- Created relationships (dependency, aggregate) between objects;
- Used multiple instances of a class, each with its own state;
- Used code that was written by someone else and provided to you.