Lakehead University
Department of Computer Science
CS 2477 Object Oriented Programming
Assignment # 3
Instructor: Dr. Jinan Fiaidhi

Q1: Define a class named Document that contains a member variable of type String named text that stores any textual content for the document. Create a method named toString that returns the text field and also include a method to set this value.

Next, define a class for Email that is **derived from** Document and includes member variables for the sender, recipient, and title of an email message. Implement appropriate accessor and mutator methods. The body of the email message should be stored in the inherited variable text. Redefine the toString method to concatenate all text fields.

Similarly, **define a class for File that is derived from Document** and includes a member variable for the pathname. **The textual contents of the file should be stored in the inherited variable text.** Redefine the toString method to concatenate all text fields.

Finally, create several sample objects of type Email and File in your main method. Test your objects by passing them to the following subroutine that returns true if the object contains the specified keyword in the text property.

```
public static boolean ContainsKeyword(Document docObject,
String keyword)
{
        if (docObject.toString().indexOf(keyword,0) >= 0)
                return true;
        return false;
}
```

Q2: The goal for this programming project is to create a simple 2D predator-prey simulation. In this simulation the prey are ants and the predators are doodlebugs. These critters live in a world composed of a 20x20 grid of cells. Only one critter may occupy a cell at a time. The grid is enclosed, so a critter is not allowed to move off the edges of the world. Time is simulated in time steps. Each critter performs some action every time step.

The ants behave according to the following model:

- Move. Every time step, randomly try to move up, down, left or right. If the neighboring cell in the selected direction is occupied or would move the ant off the grid, then the ant stays in the current cell.
- Breed. If an ant survives for three time steps, then at the end of the time step (i.e. after moving) the ant will breed. This is simulated by creating a new ant in an adjacent (up, down, left, or right) cell that is empty. If there is no empty cell available, then no

breeding occurs. Once an offspring is produced an ant cannot produce an offspring until three more time steps have elapsed.

The doodlebugs behave according to the following model:

- Move. Every time step, if there is an adjacent ant (up, down, left, or right) then the doodlebug will move to that cell and eat the ant. Otherwise the doodlebug moves according to the same rules as the ant. Note that a doodlebug cannot eat other doodlebugs.
- Breed. If a doodlebug survives for eight time steps, then at the end of the time step it will spawn off a new doodlebug in the same manner as the ant.
- Starve. If a doodlebug has not eaten an ant within the last three time steps, then at the end of the third time step it will starve and die. The doodlebug should then be removed from the grid of cells.

During one turn, all the doodlebugs should move before the ants.

Write a program to implement this simulation and draw the world using ASCII characters of "o" for an ant and "X" for a doodlebug. Create a class named Organism that encapsulates basic data common to both ants and doodlebugs. This class should have an overridden method named move that is defined in the derived classes of Ant and Doodlebug. You may need additional data structures to keep track of which critters have moved.

Initialize the world with 5 doodlebugs and 100 ants. After each time step prompt the user to press enter to move to the next time step. You should see a cyclical pattern between the population of predators and prey, although random perturbations may lead to the elimination of one or both species.

In this program you should show clearly the concept of **polymorphism and abstract class.**

Q3: Write a class named PhoneBook that has fields for a person's name and phone number. The class should have a constructor and appropriate accessor and mutator methods. Then write a program that creates at least five PhoneBook objects and stores them in an ArrayList. Use a loop to display the contents of each object in the ArrayList.

Due date: March 11.