Due Date: April 4, 2005
Points: 20

Introduction

As we talked about in class, you are going to implement a simulation of a Fox hunting down a bunny. The diagram to the left should remind you how the simulation works. For each step of the game both the bunny and the fox get a chance to look around and decide what their next move should be. After both have decided what to do they each make their move. After they move you should check to see if the bunny and fox are both in the same position. If that is the case, the fox eats the bunny and the game is over.

In class we designed an interface for the board, it does not need to be as fancy as the board shown to the left, but the basic functionality should be close. For example, you should be able to stop, run, step, reset and replay a game.

The methods we defined for the Board class are:

- `__init__(self,size)` Create the board and randomly populate the board with shrubberies. (5–10% of the cells on the board should have shrubs)
- `look(self,x,y)` Look returns a list. The elements of the list are tuples: (type, x, y) Where type must be one of ‘F’, ‘B’ or ‘S’ where ‘F’ and ‘B’ stand for the appropriate creatures, and ‘S’ stands for a shrubbery. Yes, you must bring me a shrubbery. Creatures can look in eight different directions but cannot see through other creatures or shrubberies.
- `move(self,oldx,oldy,newx,newy)` Moves a creature on the board from old location to new location.
- `addCreature(self,type,x,y)` This method allows you to tell the board the initial position for the Fox and Bunny.
- `draw(self)` draw the squares on the board to show the locations of all creatures and shrubberies.

You should start this project by implementing the Board class. You can interactively create a board, place creatures, and move them around. You can also interactively test the move and look functions. If you know your board is working properly you will find it much easier to write and debug the creature classes.
Creature Classes

The intelligence in this simulation is built into the two creatures, the fox and the bunny. How intelligent you make each of them is up to you. We will have a contest to see who has made the smartest bunny. The following diagram shows the inheritance hierarchy for the Creature classes. It is extremely important for you to follow the interfaces that we have defined. I should be able to take a Fox class and insert my Fox class in your program and the program should still run.

```
Creature
  __init__(self,x,y)
  getX()
  getY()
  x
  y
  newx
  newy

Bunny
  __init__(self,x,y)
  planNextMove(self,board)
  makeNextMove(self,board)

Fox
  __init__(self,x,y)
  planNextMove(self,board)
  makeNextMove(self,board)
```

Main Loop

One of the goals in this project is to give you another look at using polymorphism in classes, therefore the main loop of your program should look something like the following:

1. create the board.
2. create the Fox and Bunny and append them to a list of creatures.
3. while not done:
   4. call planNextMove for each creature on the list
   5. call makeNextMove for each creature on the list
   6. check to see if the creatures are on the same square
4. print out the winner and the number of turns it took for the fox to catch the bunny