About Lab 10

CRITTERS!

The basic idea of the Critter lab is simple: you make a critter by creating a subclass of the class critter.Critter. The only functionality a critter has is through the 5 methods:

```
getChar(self)
getColor(self)
getMove(self, info)
getStats(self)
fightOver(self, kill, oppFight)
```

The critter_main.py program runs a tournament by gathering up all of the critter subclasses in the current folder, and making 25 objects of each class at random locations on a grid. As these objects move around, if two try to occupy the same square on a grid they "fight"; the winner gets the square and the loser becomes less healthy. If a critter becomes too unhealthy it dies and is removed from the board.

Here are more details about the methods:

- getChar(self) is called whenever the system needs to know what character is used to represent the critter. It should return a single character, such as "B".
- getColor(self) is called when the system needs to know the critter's color. It should return something of the form

  color.<color name in capital letters>

  such as color.BLACK

  color is the Python color module; your file should import it.

- getStats(self) is called only when the critter is created. It returns a pair of values

  attack, defense

  that are used in fighting. The attack and defense values can't add up to more than 100.
- getMove(self, info) is probably the most complicated method in most critters. It needs to return one of the 5 directions ["NORTH", "SOUTH", "EAST", "WEST", "CENTER"], where "CENTER" means to not move.

  We'll talk more about getMove later.

Fights occur when 2 critters try to occupy the same square. One critter becomes the attacker and the other defender. If the attacker's attack stat is greater than the defenders defense stat (the stats are set in the getStats() method), the attacker wins; otherwise the defender wins. If the attacker wins the difference between the attacker's attack stat and the defender's defense stat is subtracted from the defender's health. If the defender's health goes below 0, the defender dies and is removed from the board.

Notice that only the defender is harmed in a fight; nothing bad happens to the attacker even if the attacker loses.

If the defender's last move before a fight is "CENTER" (i.e., stay still), the defender's defense stat is doubled for the fight.

The Chameleoturtle's strategy is to not move whenever any opponent is near. It's defense stat is 50, which doubles to 100. This means the Chameleoturtle can never be killed; it is like a rock that moves when nobody is around.

When two critters try to occupy the same square, the system decides which is the attacker and which is the defender. If one critter is moving and the other isn't, the moving critter becomes the attacker. If both are moving the system randomly chooses one to be the attacker.

Most of the strategy for critters involves the getMove(self, info) method. That info parameter has information about squares on the board that are near self's location.

info.getColor(a, b)

returns the color of the critter a squares to the right and b squares above self. For example,

info.getColor(0, 1)

tells you the color of the critter one square above you.

Note that your critter will crash if $a^2+b^2>9$.

There are other ways to use this info parameter in getMove(self, info):

- info.getHealth(a, b) tells you the health of the critter (a, b) squares away
- info.getStats(a, b) tells you the (attack, defense) stats of that critter
- info.getChar(a, b) tells you its character
- info.getColor(a, b) tells you its color
- info.getType(a, b) tells you that critter's class name, as a string (e.g., "Lion") Critters can change what character and color they show, but they can't change their class name.

Both info.getType(a, b) and info.getChar(a, b) return "." for an empty square. A pacifist critter might use this to look for non-confrontational squares to move into.

For example, Chameleoturtle looks at every square it can (every value of a and b where $a^2+b^2<=9$); if it finds any critter that could possibly be an enemy it stops moving. It can't be killed while it is stationary, so this is a pretty safe strategy.

The critter's fightOver(self, kill, oppFight) is called by the system at the end of each of the critter's fights. If your critter survives the attack it can use this information to prepare itself for future fights. The oppFight parameter has the same methods as the info parameter of getMove(self, info) but without arguments; they refer the opponent:

oppFight.getColor() is the opponent's color

oppFight.getChar() is the opponent's character

and so forth

For example, Chameleoturtle does its chameleon thing by taking on the character and color of the last opponent it fought. Here is its fightOver( ) method:

```
def fightOver(self, kill, oppFight):
    self.char = oppFight.getChar()
    self.color = oppFight.getColor()
```

and those variables self.char and self.color are returned by the Chameleoturtle's getChar(self) and getColor(self) methods.

Your critter earns points by

- Killing other critters
- Surviving. Each living critter gets a point at the end of every 100 turns.
- Grabbing a "Point Cache". These show up as numbers on the grid. To capture a point cache two of your critters need to attack it in succession -- with no attacks from other critters in between. To profit from this you might look for ways to get your critters to cooperate.

One way to get critters to communicate is through class variables. For example, if one of your critters finds a point cache at 20, 15 it could set a class variable to [20, 15]. Any time that variable is set your critters might try to move towards that location. Whoever "kills" the point cache could reset the class variable to [ ].