Giant

Constructor	`public Giant()`
Color	`Color.GRAY`
toString	`"fee"` for 6 moves, then `"fie"` for 6 moves, then `"foe"` for 6 moves, then `"fum"` for 6 moves, then repeat.
getMove	always infect if an enemy is in front otherwise hop if possible otherwise turn right.

```
public Giant () {
}

public Action getMove(CritterInfo info) {
}

public Color getColor() {
}

public String toString() {
}
```

YOU WILL HAVE NO LOOPS
AT ALL
ANYWHERE
IN YOUR CODE!!!

This is what we call an "event driven system."

- **Your code is not in control.**
- **For example your getMove() method will be called once every step.**

Constructor	`public Giant()`

```
public Giant () {
}
```

Only need to actually write this if you decide
you need fields and they require special initialization.

Color	Color.GRAY

```
public Color getColor() {
}
```

This one is pretty simple, just returns the specified color.

getMove	always infect if an enemy is in front otherwise hop if possible otherwise turn right.

```
public Action getMove(CritterInfo info) {
}
```

Pseudo code:

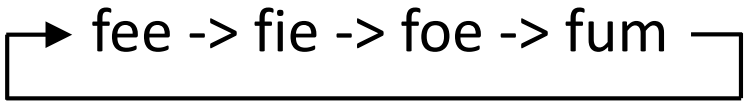
```
if enemy in front
    infect
else if open in front
    hop
else
    turn in place spinning right
```

This method is the only action which is guaranteed to be invoked exactly once by the controlling code every step of the game.

toString	`"fee"` for 6 moves, then `"fie"` for 6 moves, then `"foe"` for 6 moves, then `"fum"` for 6 moves, then repeat.

```
public String toString() {
}
```

Pseudo code:



Have 4 strings to cycle through: “fee”, “fie”, “foe”, “fum”
Switch them in order every 6 steps
…remember, no loops!
…this method will be called once every step of the game…
…you will have to keep track of which step you are on!
…have a field for step count/number
…read the spec about which method needs to do the step counting!!!!
…have some way of using step count to determine which string to return.

For tomorrow:

1. Write and test the Giant class.
2. Write and test the Lion class.