FriendlyCat looks for Pizza If It Can Avoid Mean Animals

**Short description:** Simple programming practice with method calls and if-statements.

**Objective:** Reinforce learning outcomes achieved with fatcat and little-crab scenarios.

**Level:** Beginner

**Duration:** one to two hours, depending on familiarity with little-crab and fatcat scenarios.

**Teaches:** method call, method definition, sequence of statements, if-statement, and instance variables.

**Instructions**

**Part One**

1. Open friendly-cat-start scenario. Save a Copy As friendly-cat. Close friendly-cat-start. Open your friendly-cat scenario. This is the scenario you will develop following the instructions below.
2. Go to the Cat class. (right click on Cat in the right pane, and select Open Editor) Look for the methods: ‘jumpUp’ and ‘jumpDown’. Both will use the internal method jump(). Hints:
   a. Jumping behavior is modeled after walking behavior.
   b. Study the walk() method in the Cat class and how it is used by walkLeft( ) and walkRight( ).
   c. Find the jump( ) method which has four parameters.
   d. Complete the jumpUp( ) and jumpDown( ) methods with one parameter. They use a call to the jump( ) method. In the jump( ) method call, the arguments for the two image parameters are either the pair “small-cat-fly.png” and “small-cat-fly-2.png” or the pair “small-cat-fall.png” and “small-cat-fall-2.png”.
3. In the FriendlyCat class, locate the method called keyControlMove(). This method has no return value and no parameters. You want four things to happen in this method:
   a. If the left arrow key is pressed, make the friendly cat walk left.
   b. If the right arrow key is pressed, make the friendly cat walk right.
   c. If the up arrow key is pressed, make the friendly cat jump up.
   d. If the down arrow key is pressed, make the friendly cat jump down.
   Hint: Remember that there are cat-moving methods in the superclass Cat.
4. In the FriendlyCat act() method, call your new method keyControlMove(). We want the FriendlyCat to move around when we Run the program and control its move with left, right, up, and down keys. Keep in mind that to check if a key is pressed you use Greenfoot.isKeyDown(“right”), for example. Do not forget to put a cat in the world before clicking Run to test that the friendly cat moves by pressing the corresponding keys.

**Part Two**
1. Add the Pizza class to the scenario. Make it a subclass of Actor. (Do this by right clicking on Actor and selecting “New subclass”. The “New class name” field enter Pizza and select the pizza slice image from the list of Scenario images.

2. Locate the variable in FriendlyCat to keep track of the number of pizzas a friendly cat object will eat. It is named `numberOfPizzas` and initially has a value of 0.

3. Write `lookForPizza()` method in FriendlyCat. This method is modeled after `lookForWorms()` (see http://greenfoot.org/doc/tut-3). Review the FriendlyCat’s super classes to know how to call `canSee()`, `remove()`, and `eat()`. Use these calls to write a first version of `lookForPizza()` method. Call this method in `act()`.

4. Improve your `lookForPizza()` method such that it counts the number of pizzas the FriendlyCat eats using the instance variable `numberOfPizzas` you located in Step 2. The method works as follows:
   a. `numberOfPizzas` increases by one each time a friendly cat eats a pizza (has to “see” it first and “remove” it before eating it).
   b. If the friendly cat eats three pizzas, it dances and then the pizza counter resets to 0.

5. The friendly cat now moves around, eats pizza it finds, and celebrates with a happy dance when it eats three pizzas in a row.

Part Three

1. Add a new class called MeanAnimal, which is a subclass of Animal.

2. MeanAnimals chase FriendlyCats. Model the behavior of a mean animal after the behavior of a Lobster developed and refined in little-crab scenarios. The `act()` method in MeanAnimal class have method calls that make a mean animal:
   a. turn when it reaches the end of the world
   b. turn randomly
   c. move, and
   d. look for FriendlyCat instances. If it catches a poor FriendlyCat, the mean animal celebrates with fanfare.

3. Write the definitions of the methods `turnAtEdge()`, `randomTurn()`, and `lookForFriendlyCat()`.

4. The friendly cats must be cautious and aware of mean animals if they want to live eating-pizza happy lives.