ORACLE ACADEMY

Lesson 5: Using Randomization and Understanding Dot Notation and Constructors

Try It: Practice Activities

Objectives

- Create randomized behaviors
- Define comparison operators
- Create if-else control statements
- Create an instance of a class
- Recognize and describe dot notation

Vocabulary

Identify the vocabulary word for each definition below.

	A technique that allows a class to use a method from another class or object. The dot between the class/object name and the method name indicates that the method comes from a different class or object.
	A special kind of method that is automatically executed whenever a new instance of the class is created.
	A keyword that indicates that a new object is being created.
	Symbols that compare two random values in a method.

Project

1. Continue the project from Section 3 Lesson 4.
2. Open the Project named FrogFlyL4. From the Scenario Menu, choose “Save as” and use the name FrogFlyL5. This will keep a copy from Lesson 4 in case there is a need to refer to it later.
3. Compile the Greenfoot project.
4. In the FrogWorld constructor, add one instance of a Fly object in any suitable location.
5. Compile the project and check that the Fly appears in the World.
6. Open the code editor for the Fly subclass.
7. Delete the if statement for turning the Fly with the Left arrow key.

8. Add a new if statement so that the Fly will turn left 30 degrees 70% of the time.
9. Add a new else statement to the if above so that the Fly will turn right 30 degrees the other 30% of the time.
10. The Greenfoot delay() method pauses execution. It is often useful in the act() method, which runs continuously. To see the animation more clearly, pause the animation for 20 milliseconds. Add the following statement to both the if and the else (just below the turn() method):

    Greenfoot.delay(20);
11. Compile your Greenfoot project.
12. Test your project. The Fly should have random behavior. 70% of the time it will turn left. 30% of the time it will turn right.
13. Does your project work as it should? Test and debug. Recompile as necessary.
14. Save your project and exit.

Optional Practice

1. Create a new scenario with three Actor subclasses. Write code in the act() method of each subclass that will turn() the objects in a random direction and move() the objects a random distance.
2. In your scenario's world constructor, write code that automatically creates two new instances of an Actor class when the world is initialized. Add another Actor subclass that will move as the up and down arrow keys are pressed. (Up Key=Forward, Down Key=Backward) If the “t” key is pressed, the object should turn 90 degrees. Practice moving the objects using the up, down, and t keys.