**2.1.2 Autumn II –Java List Introduction**

**Introduction**

In this assignment we will complete the Autumn scenario using Java list functions and the enhanced loop. If you know Python, you will see that the enhanced loop is equivalent to the for loop in Python and that lists function in much the same way. This loop makes it easier to catalog every object of a given type in the world and then do something to them.

Concepts covered are:

* More Java library classes
* Lists
* Generic type
* for each loop (enhanced loop)
* Keywords: **null, List**

**Materials**

* Computer with Greenfoot

**Part I: Object References**

1. Open up your completed Autumn scenario from the last assignment.
2. You already know how to communicate with a single Actor or object, now we are going to learn how to interact with several of them at once. Sometimes we want to do something to every object of a given class, or every object in a certain range (like broadcasting in Scratch). First, when a user clicks the mouse, we will change image of all the leaves.

|  |
| --- |
| * The Leaf class has a method called changeImage() that will allow you to easily switch images. Try this out by placing a Leaf object in the world and calling the method. * Let’s take a look at the API for the World class. Look for any methods that give us access to objects in the program. Find one that will allow you to find multiple objects and remember it. Hint: Methods that return object references are usually called **get methods**. |

1. Once you find the method, we need a quick discussion of how to actually use it. The method in question uses java.lang.Class as a parameter and returns java.util.list. This means it takes a class as a parameter and returns a List object. Recall that to access a class we need to use the name of the class followed by the **.class** keyword extension. For example: Actor**.class**.

In order to be able to use List objects we need to import them just like we do for Array objects. Lists are part of the Java standard library. We need to add an import statement to the top of the Block class before the class header. Once you add the import you can create List objects in the code as needed.

A list is actually not a class at all but something different called an **interface**. A Java interface acts much like a class but objects cannot be instantiated from the interface *directly*. This means you cannot use the **new** keyword to make a list! We will explore this in later assignments. For now you can treat the List interface like other classes.

|  |
| --- |
| * Look up Interface List in the Java standard API.   **To check the standard Java AP while running Greenfoot, go to *Help > Java Library Documentation*.** |

You will see that the entry for lists has this at the top:

Interface List<E>

The <E> means that a list needs a **generic type**. This is an additional data type specified after the list is declared. **A generic type argument must be an object.** This is similar to an array, which needed a type as part of the declaration. If I am going to create a list of Actor objects for example, I would make a declaration like this:

List<Actor> myList = someMethodToGetActors();

Now we are ready to start out code to get all of the leaves.

|  |
| --- |
| * Open the source code for the Block class. * Add an import statement to the top of the code:   **import** java.util.List;   * This class has a method to check for mouse clicks; we will write our code here. Find it. * Inside the method create a new list to store Leaf objects. Use the world method you found earlier to store the Leaf objects in the list. |

1. Now we need to go through every item on our list and change its image. Sounds a lot like Python in concept. To do this we will use a **for each loop** (also called an enhanced loop). The syntax looks like this:

for (ElementType variable : collection)

{

code;

}

Example:

List<String> students = someMethodToGetStudents();

for (String firstName : students)

{

reverseFirstName();

}

|  |
| --- |
| * Write your for each loop. Test your code. |

1. Let’s improve the program.

|  |
| --- |
| * Write code in Block that turns every Apple object 90 degrees when one is hit by the block. * Write code in Block that moves every Pear object to the right when one is hit by the block. * Modify your code so that only images of leaves in the left half of the screen change image when you click the mouse. |