



The University of the West Indies, St. Augustine
COMP 2603 Object Oriented Programming 1
2020/2021 Semester 2
Lab 9

In this lab, we will create various Collection objects and examine the Java APIs. Previous topics such as Polymorphism, Inheritance, and Interfaces still apply to this lab.

Part 1: LinkedList

1. Create a new project in BlueJ called **Lab9**. Bring up the **Collection** interface in Google.
2. Create a **Plant.java** class in your project with the following code:

```
public class Plant{
    private String name;

    public Plant(String name){
        name = name;
    }

    public String toString(){
        return "Plant Name: " + name;
    }
}
```

3. Create a main class called **Greenhouse** in your project. Include the statement **import java.util.*;** at the top of your main class. You will be creating collection objects which require use of this package.
4. In the **Greenhouse** main method:
 - a. Create a new **LinkedList** object called **vegetables** which holds **Plant** objects: **Collection <Plant> vegetables = new LinkedList<Plant>();**
What would happen if we omitted <Plant> from this step? Why do we use it then?

Answer:

Is the **vegetables** object polymorphic? How do you know?

Answer:

TIP: Use a Google search to find out more about a class

Google keywords:

Java
Class-name

- b. Write code to print out the details of the plants in the **vegetables** collection if the collection is not empty. If the collection is empty, then print out “No plants were found in the vegetables collection”
- c. Create the following **Plant** objects and add them to the **vegetables** collection.

Plant Object	Plant Name
p1	Large Tomato
p2	Small Tomato
p3	Potato

- d. Print out the details of the **vegetables** collection to verify the objects were added in step (c).
- e. Type the following code:

```
if(vegetables.contains(p1) ) // line1
    System.out.println("Large Tomato Plant found");
else
    System.out.println("No Large Tomato Plant found");
```

What was printed? Why?

Answer:

- f. Add **p1** to the **vegetables** collection a second time. Print out the **vegetables** collection. Was **p1** added? What type of **Collection** is **vegetables**? Does it allow duplicate objects?

Answer:

- g. Replace line 1 in step (e) with the following:
- ```
if(vegetables.contains("Large Tomato Plant"))
```
- What happens to the output now? Explain why this happens.

Answer:

- h. Create a new **Plant** object **p4** with its type as “Small Tomato”. Add it to the **vegetables** collection and print out the **vegetables** collection. Was **p4** added? Suppose we do not want duplicates in the **vegetables** collection. How can we make this work for a **LinkedList**? What does the **Plant** class need to have?

Answer:

5. In the **Plant** class, write an **equals( )** method that checks equality based on the **Plant name**. In other words, two **Plant** objects are equal if they both have the same **name**. The **equals( )** method has the following signature:

**public boolean equals(Object obj)**

6. Create a new **Plant** object **p5** with its type as “Small Potato”. Add it to the **vegetables** collection and print out the **vegetables** collection. Are there still duplicates even though we supplied an **equals( )** method to discriminate between objects? Why?
7. Create a new **Plant** object **p6** with its type as “Small Potato”. Don’t add it to the **vegetables** collection though.
- a. Type the following code and run the program:

```
if(vegetables.contains(p6)) // line1
 System.out.println("Small Potato Plant found");
else
 System.out.println("No Small Potato Plant found");
```

Small potato plant found is printed. Why did this happen?

Answer:

- b. Comment off the **equals( )** method in the **Plant** class. Repeat step (a) above. No Small Potato Plant found is printed. Why did this happen now?

Answer:

8. Add p6 to the **vegetables** collection and print out the collection. You should have:

Plant Name: Large Tomato  
Plant Name: Small Tomato  
Plant Name: Potato  
Plant Name: Small Tomato  
Plant Name: Small Potato  
Plant Name: Small Potato

9. Try to retrieve the 4th element in the **vegetables** collection directly. Is there a method in the Collection interface to make this work?

Answer:

10. Cast the **vegetables** collection to its dynamic type. Examine the List interface API.

- a. Identify least three of the methods native to the List interface:

Answer:

- b. Try these out on the **vegetables** collection.
- c. Create a new **Plant** object **p7** with its type as "Lettuce" . Write code to insert p7 at position 2 but add the element that is originally at position 2 to the end of the **vegetables** collection. Examine the API method `set(int index, E element)` for this task.
- d. Add three more random Plant objects of your choice.
- e. Remove the first element in the **vegetables** collection and insert it at the middle of the **vegetables** collection.
11. Create a new ArrayList that only contains half of elements in the **vegetables** collection. Use the Collection API method for this.