Lesson 9-1: ArrayLists

Computer Science 46A: Introduction to Programming

San José State University

Announcements

- Homework?
- Lab this Friday 3/28
- Nothing due all of next week.
    - Mar31-Apr4 is Spring Break. Enjoy.
    - Next reading assignment will be due Apr 8
- What will we be covering next class?
    - Install Lockdown browser on your machine
    - You are allowed to bring 1 sheet of paper – front and back

Learning Objectives

By the end of this lesson, you should be able to:

1. Declare an ArrayList and use common ArrayList methods
2. Use an enhanced for loop to access ArrayList values in succession

Loops: A review

- A loop is a code block that that is repeated many times
- In Java, there are three different implementations of loops:
  - While loop – if a condition is met, keep doing something
  - For loop – do something a certain amount of times
  - Do loop – do something, and then keep doing it if a condition is met

Poll Everywhere: Question 1

Which of the following loops will print “1, 2, 3, 4, 5,”?:

A)
```
for (int i=1; i<6, i++)
{
    System.out.print(i+“, ”);
}
```

B)
```
for (int i=0; i<5, i++)
{
    System.out.print(i+“, ”);
}
```

C)
```
for (int i=0; i<=5, i++)
{
    System.out.print(i+“, ”);
}
```

D)
```
for (int i=1; i<5, i++)
{
    System.out.print(i+“, ”);
}
```

PollEv.com/narayanbalasubramanian644

Loops update the iterator variable

- In all of our previous code, we have been working with single strings, objects, numbers, etc
- Consider the iterator variable i in the following code

```
for (int i=1; i<6, i++)
{
    System.out.print(i+“ , ”);
}
```

Each time we go through the while loop, we change the value of I

- What if you wanted to store multiple values for i at once?
- Answer: Use an ArrayList

ArrayLists – an introduction

- An ArrayList can be used to store a collection of objects
- Consider the following ArrayList called myNumbers:

myNumbers				
1	**2**	**3**	**4**	**5**

- The ArrayList stores five integers into memory
- ArrayLists can only store one type of object
  - .e.g Strings, Rectangles, etc
- To create ArrayLists, first import java.util.ArrayList

ArrayList Syntax

- To declare an ArrayList, use the following syntax:

```
ArrayList<type> [name] = new ArrayList<type>();
```

- For example:

```
ArrayList<Circle> myCircles = new ArrayList<Circle>();
```

myCircle

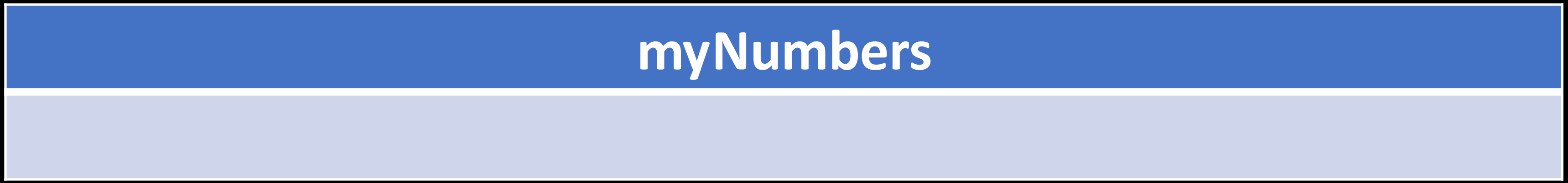
The declaration constructs the ArrayList – next we need to fill it

ArrayLists with Primitive Types

- Recall: primitive types are ints, doubles, chars, longs, boolean, etc
- ArrayLists cannot technically contain primitive types
    - Arraylists contain objects
- To use primitive types in ArrayLists, we use a “wrapper” class
- For example:

```
ArrayList<Integer> myNumbers = new ArrayList<Integer>();
```

Wrapper Class



Wrapper Classes for Primitive Data Types

Primitive Data Type	Wrapper Class
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double
boolean	Boolean
char	Character

Wrapper classes must be used to place primitive data types in ArrayLists

Common ArrayList methods: add

- `add([new value]);`
  - The add method will append new values to the end of the ArrayList

myNumbers				
1	2	3	4	5

- `add(int index, [new value]);`
  - If an index is provided, the value is inserted into the list
  - All other values get shifted to the right

See example code in `MyNumbers`

Common ArrayList methods: set and remove

- `set(int index, [new value]);`
  - The set method will change the value at a given index to a new value
- `remove(int index);`
  - The remove method will remove the value at a given index
  - The remove method also returns the given element
    - In other words, you can store the element in a new variable as output from remove

myNumbers				
1	2	3	4	5

See example code in `MyNumbers`

Common ArrayList method: size

- `size();`
  - Gets the total amount of elements that are in the ArrayList

myNumbers				
1	2	3	4	5

See example code in `MyNumbers`

Looping over ArrayList values:

- Values in an ArrayList can be accessed in succession using an "enhanced for loop"
- Syntax:

```
for (<type> [var name] : [ArrayList name])
{
    // enter code here
}
```

See example code in `MyNumbersLoop`

ArrayLists with Objects

- We create ArrayLists for any type of objects we are using
- For example:

```
ArrayList<String> names = new ArrayList<String>();

names.add(“Mike”);
names.add(“Wood”);
System.out.print(names);
```

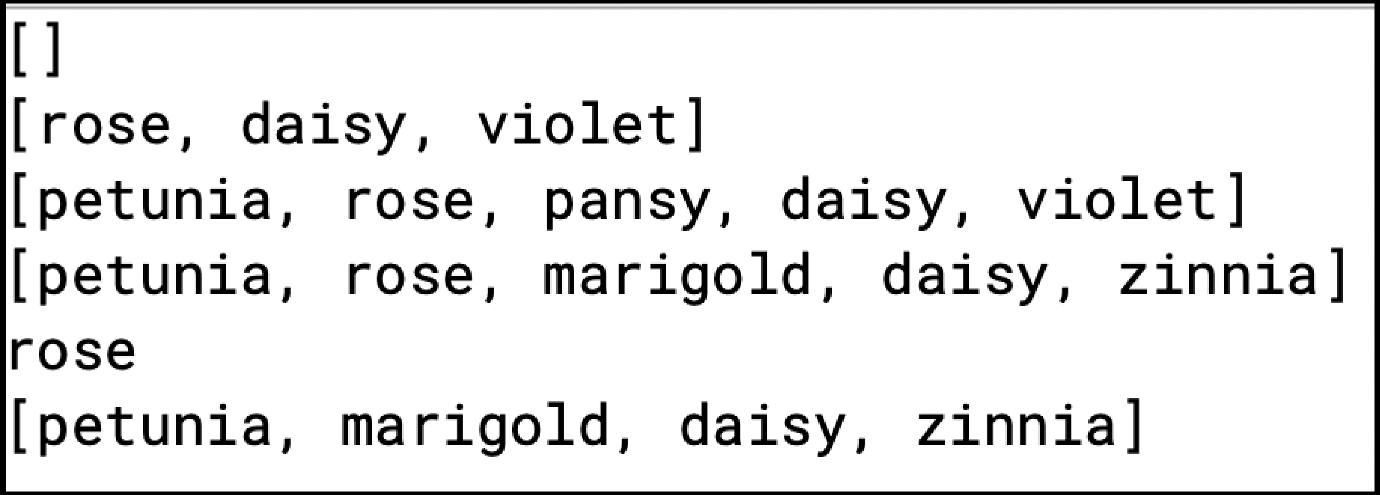
- Output:

```
      [“Mike”, “Wood”]
```

See another example in **`StockListApp`**

Participation Exercise 9-1a: `Flowers`

Goal: Use an ArrayList called garden to store a collection of flower names and experiment with ArrayList methods



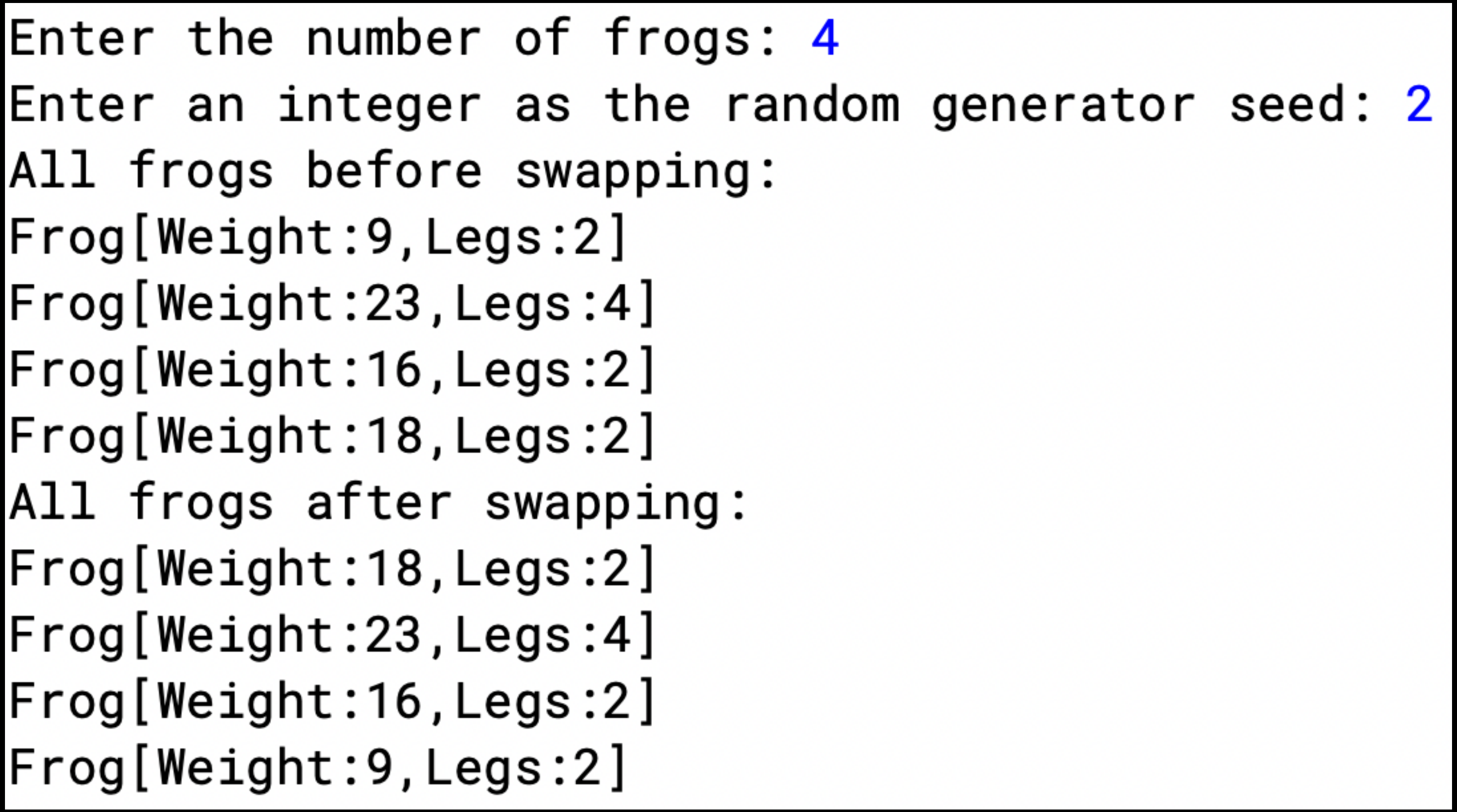
```
[]
[rose, daisy, violet]
[petunia, rose, pansy, daisy, violet]
[petunia, rose, marigold, daisy, zinnia]
rose
[petunia, marigold, daisy, zinnia]
```

Output of the `Flowers` script

Codecheck Link: HERE and on Canvas

Participation Exercise 9-1b: FrogListApp

Goal: Use an ArrayList to store a set of objects of class Frog (provided)



```
Enter the number of frogs: 4
Enter an integer as the random generator seed: 2
All frogs before swapping:
Frog[Weight:9,Legs:2]
Frog[Weight:23,Legs:4]
Frog[Weight:16,Legs:2]
Frog[Weight:18,Legs:2]
All frogs after swapping:
Frog[Weight:18,Legs:2]
Frog[Weight:23,Legs:4]
Frog[Weight:16,Legs:2]
Frog[Weight:9,Legs:2]
```

Output of the FrogListApp script

Codecheck Link: HERE and on Canvas