Review

Arrays

- An array is a list of values with the same type.
- Size of an array is established at the time of creation and cannot be changed.

```
//Declare and create an array with 5 elements:
String[] names = new String[5];
```

- Limitations of an array:

```
//Declare and initialize an array with 5 elements:
String[] names = { "Kirk", "Nathan", "Paige", "Hui", "Peter" };
```

Bonnie	Nathan	Paige	Hui	Peter

  - You can modify an element:

```
names [0] = "Bonnie"; //Modify the element at index 0
```

  - But you cannot change the length of an array (i.e., remove or add an element)

Array vs. ArrayList

Compare two program codes below, which prints a list of Strings. Where do you see the difference of array and ArrayList?

Array

```
public class AmazingTeachers {
    public static void main(String args[]) {

        //Declare and initialize an array
        String [] names = {"Kirk", "Nathan", "Paige", "Hui","Peter"};

        //Print out the array
        for (int i=0; i<names.length; i++){
        System.out.print(names[i]+ " ");
      }
    }
}
```

Code: **HERE**

ArrayList

```
import java.util.ArrayList;

public class AmazingTeachers {
    public static void main(String[]args){

        //Declar and create an ArrayList
        ArrayList<String>names = new ArrayList<String>();

        //Add items to the ArrayList
        names.add("Kirk");
        names.add("Nathan");
        names.add("Paige");
        names.add("Hui");
        names.add("Peter");

        //Print out the ArrayList
        System.out.print(names);
    }
}
```

Code: **HERE**

Declare & Create an ArrayList

ArrayList:

- An ArrayList is a resizable array.
- Array is a Java language built-in feature, and `ArrayList` is a class that needs to be imported from the `java.util` package:
  - ```
    // To import just the ArrayList class:
    import java.util.ArrayList;
    ```
  - ```
    // To import everything in package including ArrayList:
    import java.util.*;
    ```

- `ArrayList` cannot hold primitive types like `int` and `double`; they can only hold objects like `String` and the wrapper classes `Integer` and `Double`.
  - Wrapper classes: A way to use primitive data type as object type.
  - `ArrayList<Type> <ArrayListName> = new ArrayList<Type>();`
    - `ArrayList<String> names = new ArrayList<String>();`
    - `ArrayList<Integer> nums = new ArrayList<Integer>();`

- Quick practice: **HERE**

Which of the following is the correct way to create an `ArrayList` of integers?

A. `ArrayList[int] numbers = new ArrayList();`

B. `ArrayList<String> numbers = new ArrayList();`

C. `ArrayList<int> numbers = new ArrayList<int>();`

D. `ArrayList<Integer> numbers = new ArrayList<Integer>();`

Elements in an ArrayList

Array

- Elements in an array can be directly accessed.

```
//Declare and initialize an array with 5 elements:
String[] names = {"Kirk", "Nathan", "Paige", "Hui", "Peter"};
```

ArrayList

- Elements in an ArrayList cannot be directly accessed.
- Elements can only be accessed by using methods.

```
//Declare and create an ArrayList object named names
ArrayList<String> names = new ArrayList<String>();

//Add elements to an ArrayList
names.add("Kirk");
names.add("Nathan");
names.add("Paige");
names.add("Hui");
names.add("Peter");
```

Print out an ArrayList

Array

```
String[] names = {"Kirk", "Nathan", "Paige", "Hui", "Peter"};
for (int i = 0; i < names.length; i ++) {
    System.out.print(names[i] + " ");
}
```

ArrayList

```
ArrayList<String> names = new ArrayList<String>();
System.out.print(names);
```

- Quick practice: **HERE**

ArrayList Size & Other Methods

Array

```
//Given an array names
String[] names = { "Kirk", "Nathan", "Paige", "Hui", "Peter" };

//Length of an array: 5
names.length;
```

ArrayList

```
//Given an ArrayList names
ArrayList<String> names = new ArrayList<String>();
//[Kirk, Nathan, Paige, Hui, Peter]

//Size of an Array: 5
names.size();
```

Kirk	Nathan	Paige	Hui	Peter

ArrayList Methods:

```
//How will these methods modify the ArrayList??
```

- `names.add("Ed");`
- `names.add(3, "Ed");`
- `names.set(5, "Alex");`
- `System.out.print(names.get(0));//Return “kirk”`

What will print when the following code executes?

```
ArrayList<Integer> list1 = new ArrayList<Integer>();
list1.add(1);
list1.add(2);
list1.add(3);
list1.add(2, 4);
list1.add(5);
System.out.println(list1);
```

A. [1, 2, 3, 4, 5]
B. [1, 4, 2, 3, 5]
C. [1, 2, 4, 3, 5]
D. [1, 2, 4, 5]

Traversing: Array vs. ArrayList

- While loops, for loops, and for-each loops can all be used to traverse an `ArrayList` just like an array.
- Compare the codes below, which calculate the sums of integers. Where do you see the difference?

Array

```
int[] nums = {20, 30, 40, 50};
```

```
int sum = 0;
for (int i = 0; i < nums.length; i ++) {
  sum = sum + nums[i];
}
```

```
System.out.print(sum);
```

ArrayList

```
ArrayList<Integer> nums = new ArrayList<Integer>();
nums.add(20);
nums.add(30);
nums.add(40);
nums.add(50);
```

```
int sum = 0;
for (int i = 0; i < nums.size(); i++){
    sum = sum + nums.get(i);
}
```

```
System.out.print(sum);
```

Project, Practice & Assessment

Five-star spicy project: Pascal's triangle

Submit your work to our **Gallery Walk**. :)