INFO 5100
Application Engineering and Development

Agenda

- Array
- Runtime Complexity
- ArrayList
- Reference vs Value
- Linear Data Structure
- Key/Value Data Structure

Array

- Array helps you group values together
- Typed
- Fixed size
- Can be declared with literal or new keyword

Work with Array

```
public class Array {
   public static void main(String[] args) {
       // declare and initialize with literal
       int[] numbers = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
       System.out.println("array literal");
       for (int i = 0; i < numbers.length; i++) {
           System.out.println(numbers[i]);
       }
   }
}
```

Work with Array

```
public class Array {
   public static void main(String[] args) {
       // declare then fill in value
       int[] empty = new int[10];
       System.out.printf("empty array %s\n", empty);
       for (int i = 0; i < numbers.length; i++) {
           empty[i] = i * 2;
       }
       for (int i = 0; i < numbers.length; i++) {
           System.out.println(empty[i]);
       }
   }
}
```

Work with Array

```
public class TwoDArray {
   public static void main(String[] args) {
       int[][] matrix = {{1, 2, 3, 4, 5}, {6, 7, 8, 9, 10}};
       for (int row = 0; row < matrix.length; row++) {
           for (int column = 0; column < matrix[row].length; column++) {
               System.out.print(matrix[row][column]);
           }
           System.out.println();
       }
   }
}
```

Array

- Watchout for index out of bound errors
- How to insert/delete elements?

Runtime Complexity

- A measurement for how your algorithms will scale based on number of input
  - O(1)
  - O(logn)
  - O(n)
  - O(nlogn)
  - O(n^2)
  - O(2^n)
  - O(n!)
- http://www.bigocheatsheet.com/

ArrayList

- Array can be hard to work with sometimes
- ArrayList wraps around basic Array to provide cleaner/more convenient API
- Automatically changes capacity
- Stores Object types

Work with ArrayList

```
public class ArrayListExample {
   public static void main(String[] args) {
       java.util.ArrayList<String> countries = new ArrayList<>();
       countries.add("United States");
       countries.add("France");
       countries.add(0, "Spain");
       for (String country : countries) System.out.println(country); System.out.println(1);
       countries.remove(0);
       for (String country : countries) System.out.println(country);
   }
}
```

Importing Packages/Classes

- By default, Java automatically imports java.lang.* packages/classes
- You can use full class path name to replace importing

ArrayList

- https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html
- http://developer.classpath.org/doc/java/util/ArrayList-source.html
- ArrayList has an initial capacity (overridable)
- ArrayList automatically copies to a double* sized Array

Amortized Runtime Complexity

- Add normally takes O(1)
- When resizing, it takes O(m) time
- Amortized runtime complexity is O(1)

Work with ArrayList

```
import java.util.ArrayList;

public class AutoBoxing {
   public static void main(String[] args) {
       ArrayList<Integer> numbers = new ArrayList<>();
       numbers.add(1);
       numbers.add(2);
       for(int number : numbers) System.out.println(number);
   }
}
```

Auto Boxing

- To make Primitive types to work with APIs supporting Object types
- Every Primitive type has a corresponding Object type
- https://docs.oracle.com/javase/7/docs/api/java/lang/Integer.html

Auto Boxing

```java
import java.util.ArrayList;

public class AutoBoxing {
   public static void main(String[] args) {
       // Autoboxing pitfall
       System.out.println(127 == 127);
       System.out.println(128 == 128);
       System.out.println(Integer.valueOf("127") == Integer.valueOf("127"));
       System.out.println(Integer.valueOf("128") == Integer.valueOf("128"));
       System.out.println(new Integer(2000) == new Integer(2000));
   }
}
```

Reference vs Value

- When comparing two variables behavior of Object type reference and Primitive type value are different
- Java stores primitive literal value in memory
- Java stores reference to memory address in memory for Object types

Compare Equality

```
public class Equality {
   public static void main(String[] args) {
       Dog d1 = new Dog("James");
       Dog d2 = new Dog("James");
       System.out.println(d1 == d2);
       System.out.println(d1.equals(d2));
   }
}
```

Override equals method

- Default behavior of equals is to compare reference
- You can optionally override the behavior of equals function

Compare Equality

```
public class Dog {
   // omit repeated code

   @Override
   public boolean equals(Object o) {
       if (o == this) return true;
       if (!(o instanceof Dog)) return false;
       Dog d = (Dog)o;
       return this.name.equals(d.name);
   }
}
```

Linear Data Structure

- Besides Array, ArrayList, there are other important Linear Data Structures
  - Linked List
  - Stack
  - Queue
  - Heap

Linked List

- In Array, inserting elements to the beginning of the Array incur huge performance impact
- LinkedList is a loosely coupled data structure

Stack

- Stack is a special Data Structure which enforces element behaviors
  - Push
  - Pop

Queue

- Stack is a special Data Structure which enforces element behaviors
  - Enqueue
  - Dequeue

Heap

- Stack is a special Data Structure which enforces element behaviors
    - Add - as you add, the data structure put the elements in sorted place

Key/Value Data Structure

- Key/Value Data Structure speeds up data look up