Introduction to Programming
with Java, for Beginners

Primitive vs. References Type
The Stack and the Heap
Has A relationship

Primitive vs. Reference Types
- We’ve seen Java’s 4 primitive types: int, double, boolean, char
- Java also has reference types, for objects
  - Note: String is an object not primitive type
- Examples of reference variables:
  Dog d1; Counter c1; Player mario; String name;

Memory: Stack and Heap
- When we run Java programs, memory is allocated for variables and objects
- Understanding how this memory is managed helps us understand how Java works
- The JVM uses two kinds of memory: stack and heap
- The stack is used to store variables of primitive type:
  - When created in the DrJava interactions pane
  - During method calls
- The heap is used to store objects

How the Stack Works

```
> int x;
0
STACK

> x = 5;
5
STACK

> double min = 0.5;
> boolean done = false;
```

DrJava Interactions
Stack
Reference Type

- In Java, no variable can ever hold an entire object
  - One variable can only contain one thing
  - Object consists of multiple of data/state and hence stored on heap

- The term reference is used because it refers to a memory location where the object lives
  - The variable of reference type is used to access the object

- The value of reference variable is either “null” or a “heap address”
  - null means currently not pointing at any location

Value of a Reference Variable

Example:

>`Counter c1;`
>`c1 null`
>`c1 = new Counter();`
>`c1 Counter@e05ad6`

- e05ad6 is location in memory where c1 resides
  - e05ad6 hexadecimal (base 16) number
  - This location will different on your computer

- We don’t have to (and can’t) deal with these hex numbers directly
  - Convenience of using variables

How the Heap Works

<table>
<thead>
<tr>
<th>DrJava Interactions</th>
<th>Stack and Heap</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&gt; int x = 99;</code></td>
<td>![Stack and Heap Diagram 1]</td>
</tr>
<tr>
<td><code>&gt; Counter c1;</code></td>
<td>![Stack and Heap Diagram 2]</td>
</tr>
<tr>
<td><code>&gt; c1</code></td>
<td>![Stack and Heap Diagram 3]</td>
</tr>
<tr>
<td><code>null</code></td>
<td>![Stack and Heap Diagram 4]</td>
</tr>
<tr>
<td><code>&gt; c1 = new Counter();</code></td>
<td>![Stack and Heap Diagram 5]</td>
</tr>
<tr>
<td><code>&gt; c1</code></td>
<td>![Stack and Heap Diagram 6]</td>
</tr>
<tr>
<td><code>Counter@2f996f</code></td>
<td>![Stack and Heap Diagram 7]</td>
</tr>
<tr>
<td><code>&gt; c1.incrementCount();</code></td>
<td>![Stack and Heap Diagram 8]</td>
</tr>
<tr>
<td><code>&gt; Counter c2 = new Counter();</code></td>
<td>![Stack and Heap Diagram 9]</td>
</tr>
<tr>
<td><code>&gt; c2</code></td>
<td>![Stack and Heap Diagram 10]</td>
</tr>
<tr>
<td><code>Counter@4a0ac5</code></td>
<td>![Stack and Heap Diagram 11]</td>
</tr>
</tbody>
</table>

String

- A sequence of characters
- A String is a built-in Java object type
- Java provides this type because it’s used so frequently
- Examples of String creation:

>`String s1 = new String("hello");`
>`String s2 = "hello"; // commonly used shortcut`
>`s2 + " you!"
"hello you!"
>`s2 = "The result is " + 100;`
>`System.out.println(s2);
"The result is 100"`
String (contd..)

DrJava Interactions

Stack and Heap

> String c1 = new String("Hill");

27996E

HEAP

STACK

Note:
- Dr Java, does not give the heap address of String reference, this because Strings are special objects

> String c1 = new String("Hill");
> c1
"Hill"

“Has a” Relationship

- An object of type A has an instance variable which is an object whose type is B. (A “has a” B.)
- E.g: A Freshman object whose room is of reference type DormRoom

The UML diagrams below show instance variables and methods of Freshman and DormRoom object:
- UML(Universal Modeling Lanaguage) industry standard used to describe classes in OOP

DormRoom Code

```java
public class DormRoom{
    private int num;
    private String bldgName;
    public DormRoom(int n, String b){
        num = n;
        bldgName = b;
    }
    public String getLocation(){
        return num + " " + bldgName;
    }
}
```

A DormRoom on the Heap

> DormRoom room = new DormRoom(208, "Hill");
> room.getLocation()
"208 Hill"
Freshman Code

```java
public class Freshman {
    private String name;
    private DormRoom room;
    public Freshman(String n, DormRoom r) {
        name = n;
        room = r;
    }
    public String getName() { return name; }
    public DormRoom getRoom() { return room; }
}
```

A Freshman on the Heap

```java
> DormRoom room = new DormRoom(208, "Hill");
> Freshman f = new Freshman("jo", room);
> f.getName()
"jo"
> f.getRoom().getLocation()
"208 Hill"
```