

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;

ESE112

2/12

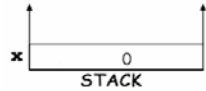

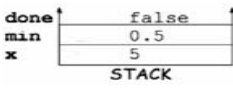
Memory: Stack and Heap

- 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*

ESE112

3/12

How the Stack Works

DrJava Interactions	Stack
> int x;	 <p>A diagram of a stack frame. On the left, a vertical line is labeled 'x'. On the right, another vertical line is labeled 'STACK'. A horizontal line connects these two lines, and inside this rectangle, the number '0' is centered.</p>
> x = 5;	 <p>A diagram of a stack frame. On the left, a vertical line is labeled 'x'. On the right, another vertical line is labeled 'STACK'. A horizontal line connects these two lines, and inside this rectangle, the number '5' is centered.</p>
> double min = 0.5; > boolean done = false;	 <p>A diagram of a stack with three frames. On the left, three vertical lines are labeled 'done', 'min', and 'x' from top to bottom. On the right, a vertical line is labeled 'STACK'. Three horizontal lines connect these lines, forming three stacked rectangles. The top rectangle contains 'false', the middle one contains '0.5', and the bottom one contains '5'.</p>

ESE112

4/12

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

ESE112

5/12

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 differ on your computer
- We don't have to (and can't) deal with these hex numbers directly
 - Convenience of using variables

ESE112

6/12

How the Heap Works

DrJava Interactions	Stack and Heap
<pre>> int x = 99; > Counter c1; > c1 null</pre>	
<pre>> c1 = new Counter(); > c1 Counter@2f996f</pre>	
<pre>> c1.incrementCount(); > Counter c2 = new Counter(); > c2 Counter@4a0ac5</pre>	

ESE112

7/12

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"
```

ESE112

8/12

String (contd..)

DrJava Interactions	Stack and Heap
<pre>> String c1 = new String("Hill");</pre>	

Note:

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

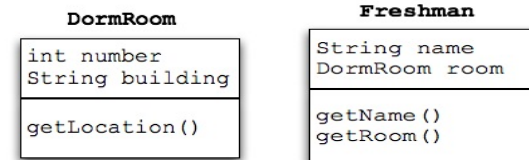
```
> String c1 = new String ("Hill");
> c1
"Hill"
```

ESE112

9/12

"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

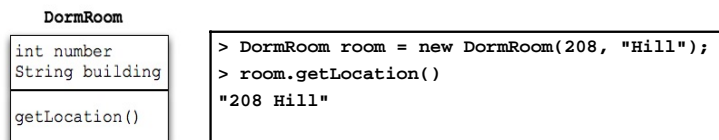


- The *UML diagrams* below show instance variables and methods of Freshman and DormRoom object:
 - UML(Universal Modeling Lanaguage) industry standard used to describe classes in OOP

ESE112

10/12

DormRoom Code



```

public class DormRoom{
    private int num;
    private String bldgName;

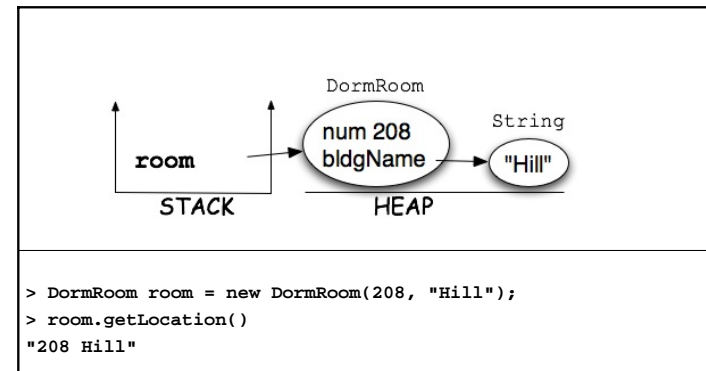
    public DormRoom(int n, String b){
        num = n;
        bldgName = b;
    }

    public String getLocation(){
        return num + " " + bldgName;
    }
}
  
```

ESE112

11/12

A DormRoom on the Heap



ESE112

12/12

Freshman Code

Freshman

```
String name  
DormRoom room
```

```
getName()  
getRoom()
```

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

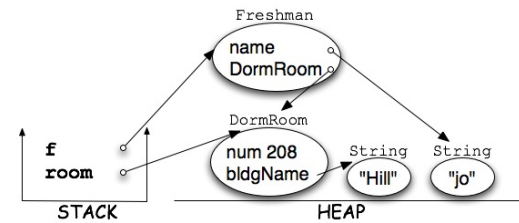
```
public class Freshman{  
    private String name;  
    private DormRoom room;  
  
    public Freshman(String n, DormRoom r){  
        name = n;  
        room = r;  
    }  
  
    public String getName(){ return name;}  
    public DormRoom getRoom(){ return room;}  
}
```

ESE112

13/12

A Freshman on the Heap

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



ESE112

14/12