Introduction to Programming
with Java, for Beginners

"Has a" Relationship
Null, NullPointerException
Call by Reference

References Recap
- The value of reference variable is
  - Either "null" or a "heap address"
  - null means currently not pointing at any location
- Counter c = new Counter();
  - new Counter() allocates space for the actual Counter on the heap, and initializes it
  - c actually allocates space to hold a reference to a Counter
    - c is placed on the stack and existent as long as
      - Method does not return (if declared in a method)
      - Dr. J Interactions pane is not reset (if declared in Interactions pane)
- The assignment makes c refer to the new Counter

"Has a" Relationship or Composition
- 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 show instance variables and methods of Freshman and DormRoom object:
- UML (Universal Modeling Language) 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;
    }
}
```

> DormRoom room = new DormRoom(208, "Hill");
> room.getLocation()
"208 Hill"
A DormRoom on the Heap

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

ESE112

Freshman Code

> 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

Note on Operators

- Dot operator and parentheses for method calls have same precedence
- Associativity  L to R
- f.getRoom().getLocation() is equivalent to (f.getRoom()).getLocation()
More Interactions I

> DormRoom rr;
> Freshman f = new Freshman(“Tom”, rr);
> f.getRoom().getLocation()
// Error - Why?

- If you try to access a datafield/method of reference variable whose value is null, then you get a `NullPointerException`
  - i.e. you try to access some object that has not been created
  - E.g. Person p;
    p.getName() //Gives a runtime error
  - Runtime error because p.getName is valid syntax i.e. it compiles but does not execute

Null in general

- **null** is a legal value for any kind of reference variable
- Example:
  - Person p, Counter c; Player mario
  - p, c, and mario are null
- **null** can be assigned, tested, and printed
  - Example: if(mario == null)

More Interactions II

> DormRoom room = new DormRoom(208, "Hill");
> Freshman f = new Freshman(“jo”, room);
> DormRoom r = new DormRoom(176, "McNair");
> f.changeRoom(r);
> f.address()
 "176 McNair”
> f.hasARoom()
true

Reference Variables as parameters

DormRoom myRoom = new DormRoom(1, “Hill”);
f.changeRoom(myRoom);

void changeRoom(DormRoom r){
    room = r;
}

- When parameters of reference type are inputs, the entire object is not copied
- Only the reference is copied
- This means that `myRoom` and `r` refer to the same object!
- Changes made to the object referenced by `r` remain changed when the method returns
- This is known as call by reference