Introduction to Programming
with Java, for Beginners

Comparing Strings
Null Pointer Exception
"Has a" Relationship

Primitive and Reference Types Recap
- We've seen Java’s 4 primitive types: int, double, boolean, char
- Java also has reference types, for objects
- Examples of reference variables:
  Dog d1; Counter c1; String s;
- The value of reference variable is
  - Either “null” or a “heap address”
  - null means currently not pointing at any location

Comparing Strings
- If the == operator is used
  - Java compares the addresses where the String objects are
    stored, not the letters in the String
  - For example:
    > String a = "hi";
    > String b = "hi";
    > a == b
    > false

- Use the String class’ equals method to compare two
  Strings for equality
  > a.equals(b)
  > true
  > b.equalsIgnoreCase("Hi")
  > true

Null Pointer Exception
- null is a legal value for any kind of object
  - i.e. Person p, Counter c, Player mario
- null can be assigned, tested, and printed
- But if you try to use a field or method of null, you get a
  NullPointerException i.e. you try to
  access some object that has not been created
  > p.getName()
  > mario.getStrength()
“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 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;
    }
}
```

A DormRoom on the Heap

```java
> 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;}
}
```

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

More methods to Freshman

```java
public class Freshman{
    // ...
    public void changeRoom(DormRoom r){
        room = r;
    }
    public String address(){
        return room.getLocation();
    }
    public boolean hasARoom(){
        if(room != null)
            return true;
        else
            return false;
    }
}
```

More Interactions

```java
DormRoom r = new DormRoom(176, "McNair");
f.changeRoom(r);
f.getRoom().getLocation()  // "176 McNair"
f.address()  // "176 McNair"
f.hasARoom()  // true
DormRoom rr;  // rr is null
f.changeRoom(rr);
f.hasARoom()  // false
f.getRoom().getLocation()  // Error – Why?
```