Unix/Vim Help Session
TOMORROW 6PM 306 Soda

DEFINING CLASSES

Fields: Variables stored in objects.
aka instance variables.
amanda.age → field
amanda.introduce() → method call
```java
class definition

class Human {
  public static int numberOfHumans;
  public int age;
  public String name;

  public void introduce() {
    System.out.println("I'm " + name + " and I'm " + age + " years old.");
  }

  public void copy(Human original) {
    age = original.age;
    name = original.name;
  }

  public Human(String givenName) {
    age = 6;
    name = givenName;
  }

  public void change(int age) {
    String name = "Tom";
    this.age = age;
    this.name = name;
  }

  public Human() {
    numberOfHumans++;
  }

  public int kids = Human.numberOfHumans/4;
```

`amanda.copy(rishi);`

`amanda.change(11);`

`int kids = Human.numberOfHumans/4;`
Each Human object can have different values of age, name.

```java
Human amanda = new Human("Amanda");

Human amanda = new Human();
amanda.age = 6;
amanda.name = "Amanda";

amanda.introduce();
```

Output: I'm Amanda and I'm 6 years old.
Default constructor takes no parameters, does no initializing.

```java
public Human() {
    this.age = 0;
    this.name = "Untitled";
}
```

Override default constructor

The "this" Keyword

"amanda.introduce()" implicitly passes an object (amanda) as a parameter called "this".

**IMPORTANT: You CANNOT change the value of "this"!**

```java
this = amanda;  // Compile-time error!
```
The "static" Keyword

Static field: a single variable shared by a whole class of objects.
Also called class variables.

System.in & System.out are static fields.

Static method: Doesn't implicitly pass an object as a param.

```java
public static void printHumans() {
    System.out.println(numberOfHumans);
}
```

main() is always static.

**IMPORTANT: In a static method, THERE IS NO "this"! referencing "this" → compile-time error!**