*me, a C/C++ developer learning Java for the first time

Java for dummies

*the laptop

int x;

int foo[] = new int[100];

int foo[] = new int[100];
foo =

What have you done

int foo[] = new int[100];
foo = new int[50];
Highlights

- Overload equals

classy x;
classy y;
y=x; // equals operator
Review: Copy constructor

To avoid double deleting (crashes program) or multiple pointers looking at the same spot...

We have to redefine the copy constructor if we use dynamic memory

The copy constructor is another special constructor (same name as class):

```cpp
Dynamic();
~Dynamic();
Dynamic(const Dynamic &d);```

```cpp`
Review: Copy constructor

When exactly does a copy constructor run?

1. You use an '==' sign with classes
2. You call-by-value a class as an input to a function (i.e. do not use &)
3. You return an inputted class to function

(see: copyCout.cpp)
Copy constructor: arrays

How would you copy a dynamically created array inside a class?

```cpp
class rng{
private:
  double* array;
public:
  rng();
  rng(const rng &original) //write me!
};
rng::rng()
{
  array = new double[10];
  for(int i=0; i < 10; i++)
  {
    array[i] = rand()%100; //0-99
  }
}
```

What if this was a normal array?

(see: copyArray.cpp)
Copy constructor vs. '=='

There is actually two ways in which you can use the '==' sign...

1. The copy constructor, if you ask for a box on that same line:
   ```cpp
classy x;
classy y = x; // copy constructor
```

2. Operator overload, if you already have a box when using '==':
   ```cpp
classy x;
classy y; // y gets box
y=x; // equals operator
```

(See: copyVsEquals.cpp)
What is the difference between copy and '=='?

“copy” is a constructor, so it creates new boxes.

'=' is changing the value of an existing box (but the idea of not sharing is the same).

The “proper” way to implement this is nuanced... see code comments if you care (See: overloadEquals.cpp)
Review: Copy constructor

This means we should modify our list of when copy constructor is called to:

1. You use an '=' sign with classes **on a declaration line**
2. You call-by-value a class as an input to a function (i.e. do not use &)
3. You return an inputted class to function
When using pointers in a class or dynamic memory, you should create:

1. Deconstructor
2. Copy constructor
3. Overload '==' operator

Typically the built-in functions are not sufficient if you use a “new” or '*'
Simple card game

Let's make a small card game based on hearthstone (simplified)
Simple card game

Rules:
- Cards **attack** and **defense** values
- Defender draws one card
- All cards always attack
- Defender selects which card to defend with
- Both attack & defend cards lose life equal to opposite's attack

(See: simpleStoneV2.cpp)