Java Classes and Objects

Classes and objects are used to model data that are not primitive

Eg. Complex numbers have to be coded as objects. A complex number (x + y*i) where x and y are numbers and i is the sqrt(-1). Complex numbers can be added, subtracted, and multiplied.

We can implement a class Complex to implement this type:

Complex x,y,z;

// x, y, z are not objects. They are variables to reference an object. As of now they are null.

System.out.println(x.toString());

// This will crash the program. Since x is null and the .operator is used to x, throws a NullPointerException

Whenever you use .operator on ask: "could the thing before the .operator be null? If so, implement a protection plan.

x = new Complex(2, 3);

y = new Complex(1,2);

// new calls the constructor to create the new object, now x and y are references to the new objects

z = x;

// If a variable is assigned to an object with =, the variable is moved to the object on the right of the =. Now both z and x are references to the same object, they are aliases.

Object semantics shows up with = and ==

Complex x, y;

Creates two variable to reference Complex objects, null now.

x = new Complex(1,2);

x is moved to reference object [1 + 2i]

System.out.println(x.toString());

prints 1 + 2i

y = x;

y is moved to reference the same object x is [1 + 2i]

System.out.println(y.toString());

prints 1 + 2i

y.plusplus();

changes the object y is referencing to [1 + 2i] -> [2 +2i]

System.out.println(x.toString());

prints [2 + 2i] since x and y are referencing the same object

y = new Complex(3,2);

y is now referencing a new object [3 + 2i]

System.out.println(x.toString());

prints [2 + 2i] since x is still referencing the same object

x == y;

false

x = new Complex(3,2);

x is now referencing a new object [3 + 2i]

x == y;

false. Even though x and y respective objects have the same values, they are not the same object

x.equals(y);

true. A method must be used in order to check if the objects have the same value.

For objects, == tells whether the objects are aliases.

What is you want to add two objects?

You must use a method:

a) *c = a.addOn(b);*

This makes c = a+b. This method belongs to object a

b) *c= Complex.add(a, b);*

This makes c = a+b. This method does not belong to any object, but its owned by the class (static method)