CMSC 131

Fall 2018

Primitives vs. References

There are two kinds of variables:

- Primitives
- References to objects

Let's look at the memory diagram (Stack and Heap) for declaring these local variables:

```
int x = 7;
double y = 3.4;
Dog z = new Dog();
Scanner s = new Scanner(System.in);
```

References

How many Dog objects are created by this statement:

```
Dog a, b, c;
```

Creating Strings is Unique

Two ways to do (essentially) the same thing:

```
String x = “hello”;
String x = new String(“hello”);
```

Taking out the Garbage

Let's talk about the garbage collector by considering the memory diagram for this:

```
String s = new String("hello");
s = new String("goodbye");
s = new String("whatever");
```

Assignment with References (Aliasing)

First consider the memory diagram for this:

```
int x = 7, y = 12;
y = x;
```

Now consider the memory diagram for this:

```
String x = “blue”, y = “orange”;
y = x;
```

Aliasing occurs when two variables refer to the same object.

Can we make copies of objects

1. There is a special method called *clone*. (Next semester…)
2. Using a *copy constructor* (later this semester)

```
String x = “hello”;
String y = new String(x);        // invoking “copy constructor”
```

More details about constructors later…

== vs. equals

Let's draw memory diagrams and consider:

```
String a = "cat";
String b = a;
String c = new String("cat");
```

Are these true or false?

```
a == b
a == c
a.equals(b)
a.equals(c)
```

What does equals really check?
What does == really check?

Let's write a typical Java class…

Example: Student.java, Driver.java

- Instance variables
- Instance methods
  - void sayHello()
  - void spendToken()
  - void acceptTokens(int numTokens)

(To be continued next class…)