Announcements

• Project #2 has been posted
• Exam #1 is on Monday 10/8
Recall: Objects have State and Behaviors

More examples:
- String
- Scanner
- Dog
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:

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

Defines a kind of object
• Instance variables (for the state)
• Instance methods (for the behaviors)

Let’s write a class together:
Example: Dog.java, Driver.java
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:

```java
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:

```java
String s = new String(“hello”);
s = new String(“goodbye”);
s = new String(“whatever”);
```
Assignment with References (Aliasing)

First consider the memory diagram for this:
```java
int x = 7, y = 12;
y = x;
```

Now consider the memory diagram for this:
```java
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)

\[
\begin{align*}
\text{String } &\text{ } x = \text{“hello”;} \\
\text{String } &\text{ } y = \text{new String}(x); \quad \text{// invoking “copy constructor”}
\end{align*}
\]

More details about constructors later…
== vs. equals

Let’s draw memory diagrams and consider:

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

Are these true or false?

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

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