ANNOUNCEMENTS

- Pick up your Exam 1 papers (before/after class or in Adrienne’s office hours)
- Lab 1 assessment this week – you need to report at the time given in the email sent on Friday 2/5.
- Lab 2 will begin week of 2/15 in recitation.
Methods

1. Calling methods
   - Asking objects to perform some action
     - e.g. move, turn, eat
   Syntax:
     method Name (argument list);

2. Defined methods
   - Give Java source code for what the method is supposed to do.
   Syntax:
     Method header (signature)
     public returnType methodName (parameter list)
Method body - enclosed in & including \& 3
- calls to other methods
- if statements

GREENFOOT CLASS DOCUMENTATION

- Available from inside Greenfoot
- Information about classes built-in that you may want to use/have used in the past
  - Actor
  - World
  - Greenfoot
Constructors are special methods that are called every time an object is created – they set up the initial state of our objects.

Explicit constructors (ones that you can see in the source code) look like this:

```java
public NameOfClass()
{
}
```

A constructor has the same name as the name of the class.

It does not have a return type.

If there is no explicit constructor in the source code for a class, Java provides an implicit one that you do not see in the source code, but is inserted at compile time.
Looking at the constructor of CarWorld, we can see a method call that looks like this:

\[ \text{super}(x, y, z) \]

Here, we are not calling a method called super, but rather super is a keyword that indicates the superclass. In this case, we are calling the superclass’ constructor.

We can add objects to the world when it is created by calling the `addObject` method from the world.

Example

```
addObject(new Car(), 34, 56);
```

Note that we need to create a new Car object to add by using the expression `new Car()`. This expression creates an object and calls the constructor of that object.

The numbers that follow are the x and y coordinates of where we would like the object to be in the world.
If we want to store information in our programs, we use a variable.

Variables are places to store information/data.
Instance variables store information important to the entire class.

Syntax:
```
private <type> <identifier>
```

who has access
what kind of values are stored
name of variable

public vs. private

Anyone has access (any part of the program) only accessible from within the class
After we declare the instance variables, it is good practice to give it an initial value.

We would give an instance variable and initial value in the constructor of the class.

Example

```java
_barrelsHit = 0;
```

Note that this expression uses the assignment operator (=) and takes the values on the right hand side and assigns them to the variable on the left hand side.

If we create direct sub-classes of Actor, we do not have a “move” or “turn” method.

The Feb 12 version of the scenario shows how to begin to teach Actors how to move on their own.