Previous Lecture:
- Iteration—while, for

Today’s Lecture:
- String objects
- Intro to OOP
- Reading: Sec 2.6, 4.0, 4.1

String (LL Sec 2.6)
- An object!
  - Made up of primitive data type char
  - Think of it as a fancy array of chars
- Creating Strings:
  - String s1 = “Hello”;
- Concatenate using +:
  - String s2 = s1 + ” class”;
  - int x = 100;
  - s2 = s2 + x;
  - System.out.println(s2);
  - System.out.println(“CS”+99+1);

Count down v.2
```java
int t = Keyboard.readInt(); // time left
String mes; // message to be printed
for (; t>0; t--) {
    mes = “T-” + t + “ second”;
    if ( t!=1 )
        mes = mes + “s”;
    System.out.println(mes);
}
System.out.println(“Take-off!”);
```

Terminology and concepts
- **Object**: contains variables (fields, instance variables) and methods
  - **Variables**: “state” or “characteristics” e.g., name, age
  - **Methods**: “behavior” or “action” e.g., yell, bounce
- **Class**: blueprint (definition) of an object
  - No memory space is reserved for object data

Class definition vs. object instantiation
- If you want make a whole lot of cookies, you may want to
  - Make a cookie cutter—define the class
  - Stamp out the cookie— instantiate an object

    Making a cookie cutter ≠
    Getting a cookie

We have used different classes already:
- Keyboard
- Math
- System, String
Above classes provide various services
(related services are grouped in same class)
Implementation details of the class are hidden from the client (user)
Suppose we’re dealing with rectangles...
(We write code in method main as we used to)
"Data" may be two coordinates: \((x_1,y_1,x_2,y_2)\)

```
// draw a rectangle
drawline(…)
drawline(…)
drawline(…)
drawline(…)
```

(Note: This page and the following pages do not show actual/complete code. We’ll have more detailed discussion next lecture.)

Why not group the characteristics and needed operations for "rectangle" together?
If we create some rectangle “template,” then we can easily get rectangles (including the needed calculations) when we want

```
public class Rect {
    // attributes
    private double left;
    private double right;
    …

    // drawRect method
    …
    // area method
    …
    // perimeter method
    …
}
```

```
public class UseRect {
    public static void main (string[] args) {
        // create a rect
        Rect r1 = new Rect();
        // calculation on r1
        r1.area();

        // create another rect
        Rect r2 = new Rect();
        r2.drawRect();
    }
}
```

OOP ideas
- Aggregate variables/methods into an abstraction (a class) that makes their relationship to one another explicit
- Objects (instances of a class) are self-governing (protect and manage themselves)
- Hide details from client, and restrict client’s use of the services
- Allow clients to create/get as many objects as they want

Variables
- **TWO main types of variables:**
  - **Primitive type**
  - **Reference to object**
- Some variables with different properties:
  - **Local**: live and die inside a method
  - **Instance variable**: owned by and accessed through individual instances (objects)
  - **Static variable**: class variable shared by all instances—only one copy in a class