CS100M Lecture 17

- Previous Lecture:
  - Loops
  - String objects
  - Intro to OOP

- Today’s Lecture:
  - Intro to OOP
  - Class definition: instance variables & methods

- Reading: Sec 4.0-4.2

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*

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

Class Definition

```java
public class class-name {
  
  declaration (and initialization)

  constructor

  methods

  }
```

October 28, 2003
Class definition: declarations

class Interval {
  private double base; // low end
  private double range; // interval range
}

- **Declarations** in a class define **fields** *(instance variables)* of the class
- Each class is a **type**. Classes are *not* primitive types.

Declarations Revisited

- Syntax:    **type** name;
- Examples:  **int** count;
  Interval in1;
  Interval in2;

- Instance variables have default initial values
  - int variables: 0
  - Non-primitive (reference) variables: **null**
- Value **null** signifies that no object is referenced

Object instantiation

- An expression of the form **new** class-name() computes a reference to a newly created object of the given class
- Examples:
  Interval in1; //declaration
  in1 = new Interval(); //instantiation
  //Combined declaration & instantiation
  Interval in2 = new Interval();

Objects are referenced

```java
public class Client {
    public static void main(String[] args){
        Interval in1;          //declaration
        in1 = new Interval();  //instantiation
        //Combined declaration & instantiation
        Interval in2 = new Interval();
        System.out.println(in1.base + in1.range);
    }
}
```

Manipulating references

```java
public class Client {
    public static void main(String[] args){
        Interval in1;
        in1 = new Interval();
        Interval in2;
        in2 = in1;
        System.out.println(in2.base + in2.range);
    }
}
```

Accessing fields

```java
public class Client {
    public static void main(String[] args){
        Interval in1;
        in1 = new Interval();
        Interval in2;
        in2 = in1;
        System.out.println(in1.base + in1.range);
    }
}
```
Class definition: methods

class Interval {
    private double base;  // low end
    private double range; // interval range

    // Getter method
    public double getEnd() {
        return base + range;
    }
}

Methods

A method is a named, parameterized group of statements

```
modifier return-type method-name ( parameter-list ) {  
    statement-list
}
```

return-type void means nothing is returned from the method

There must be a return statement, unless return-type is void

Calling an instance method

public class Client {
    public static void main(String[] args){
        Interval in1;  
        in1 = new Interval();
        double x;
        x = in1.getEnd();
    }
}