- Previous Lecture:
  - **`for`** loop
  - Defining a class—writing methods
- Today's Lecture:
  - Review
  - Defining a class—constructors
- Assigned reading:
  - T Sec 3.1.3

Loops

- Infinite loop
  - Bad! Program execution is stuck in the loop (until memory is exhausted)
- Definite iteration
  - A loop that ends after a definite no. of passes—you know how many times loop body executes
- Indefinite iteration
  - A loop that ends! But it is hard to tell at the start of program execution how many times loop body executes

Class Definition

```
public class class-name {

    declaration (and initialization)

    constructor

    methods

}
```


Constructor

- A *constructor* is used to initialize fields in objects
- Each class has a default constructor
- Can define your own constructor:

```
modifier class-name ( parameter-list ) {
    statements-list
}
```

- Use `public` as the modifier for now
- An instance method that has *no* return type

Default constructor

- Automatically inserted by Java
- An explicitly defined constructor replaces the default one

```
/** Constructor for Interval */
public Interval() {}
```


Overloading the constructor

- Two methods with the same name in the same class are said to be overloaded
- The method *signatures*, not names, are different
- The signature of a method is its name and its parameter types (including the order of the types)

Signatures of `Interval` constructors

Interval(double, double)

Interval()

Interval(double)


```
class Interval {
  private double base;  // low end
  private double width; // interval width

  public Interval(double b, double w) {
    base= b;
    width= w;
  }
  /** Interval with base 0, width w */
  public Interval(double w) {
    Interval(0,w);
  }
}
```



```
class Interval {
  private double base;  // low end
  private double width; // interval width

  public Interval(double b, double w) {
    base= b;
    width= w;
  }
  /** Interval with base 0, width w */
  public Interval(double w) {
        this(0,w);
  }
}
```


Constructor invocation

`new` ***class-name*** **(** *expression-list* **)**

- Above expression yields a reference to a *new* object of the given *class-name*
- The defined (or default) constructor is invoked on the new object created by **`new`**