Constructors

Recall:

- In an object, Java automatically initializes any integers to 0, boolean fields to false, and reference fields to the value null.

Constructor Methods

- Constructor methods are instance methods used to initialize objects at the time the objects are declared.

- If we do not wish to use the default initializations, we can write our own constructors to initialize our objects to whatever we want.

Example

- In our class *fraction*, we can add the following constructor.

```
class fraction{
        private int num;
        private int den;

        public fraction ()
        {
                num = 0;
                den = 1;
        }

}
```

- To create and initialize an object using our constructor in the main method, type:

```
fraction f = new fraction ();
```

- The preceding line will create object f of type fraction and initialize the numerator to 0 and the denominator to 1

Notice:

1. The name of the constructor is the same name as the class (i.e. fraction).
2. It is implied that a constructor will return an object.
3. The constructor is an instance method. Thus you can refer to variables in the class directly (i.e. num, den).
4. The call does not use dot notation (i.e. the call is made by *f = new fraction ()* and not *f = new f.fraction()*) .

Overloading Constructor Methods

- We can have more than one constructor method for an object.
  - The constructor that is being called is dependent on the parameter/s being sent. (like previous methods)

- In our class *fraction*, we can **add** the following constructor.

```
public fraction (int n, int d)
{
    num = n;
    den = d;
}
```

- To create and initialize another object using our new constructor, type:

```
fraction e = new fraction ();
fraction f = new fraction (1,2);
```

- The preceding line will create an object *e* of type fraction and initialize num to 0 and den to 1.
- It will also create another object *f* of type fraction and initialize num to 1 and den to 2

Another Example

- In our class *fraction*, we can add yet another constructor.

```
public fraction (fraction tempf)
{
    num = tempf.num;
    den = tempf.den;
}
```

- To create and initialize objects using our new constructor, type:

```
fraction g = new fraction (1,2);
fraction f = new fraction (g);
fraction h=g;
```

- The preceding code first creates a new object *g* and initializes its values to 1 and 2.
- It will also create another object *f* and initialize its values to the ones in g.