Unit 9

Classes and Methods
- the `return` Statement
- Overloaded Methods

return Statement

- A method, unless void, returns a value of the specified type to the calling method.
- The return statement is used to immediately quit the method and return a value:

return *expression*;

The type of the return value or expression must match the method's declared return type.

This is a great way to double-check your work on exams and quizzes….

return Statement (cont'd)

- A method can have several return statements; then all but one of them must be inside an if or else (or in a switch):

```
public someType myMethod (...)
{
    ...
    if (...)
        return <expression1>;
    else if (...)
        return <expression2>;
    ...
    return <expression3>;
}
```

return Statement (cont'd)

- A boolean method can return true, false, or the result of a boolean expression:

```
public boolean myMethod (...)
{
   ...
   if (...)
      return true;
   ...
   return n % 2 == 0;
}
```

return Statement (cont'd)

- A void method can use a return statement to quit the method early:

```
public void myMethod (...)
{
   ...
   if (...)
      return;
   ...
}
```

No need for a redundant **return** at the end

return Statement (cont'd)

- If its return type is a class, the method returns a reference to an object (or null).
- Often the returned object is created in the method using new. For example:

```
public Fraction inverse ()
{
    if (num == 0)
        return null;
    return new Fraction (denom, num);
}
```

- The returned object can also come from a parameter or from a call to another method.

Overloaded Methods

- Methods of the same class that have the same name but different numbers or types of parameters are called *overloaded* methods.
- Use overloaded methods when they perform similar tasks:

```
public void move (int x, int y)   { ... }
public void move (double x, double y)   { ... }
public void move (Point p)   { ... }

public Fraction add (int n)   { ... }
public Fraction add (Fraction other)   { ... }
```

Overloaded Methods (cont'd)

- The compiler treats overloaded methods as completely different methods.
- The compiler knows which one to call based on the number and the types of the parameters passed to the method.

```
Circle circle =  new Circle(5);
circle.move (50, 100);
Point center =
            new Point(50, 100);
circle.move (center);
```

```
public class Circle
{
    public void move (int x, int y)
     { ... }

    public void move (Point p)
     { ... }
    ...
```

Overloaded Methods (cont'd)

- The return type alone is not sufficient for distinguishing between overloaded methods.

```
public class Circle
{
    public void move (int x, int y)
     { ... }

    public Point move (int x, int y)
     { ... }
    ...
```