Lecture 15

Topics covered till this point:

I. Primitive types

- byte, short, int, long

- double, float

- boolean

- char

II. Constructed types

- string

III. Methods

- parameter passing

THE TERTIARY OPERATOR

We have studied unary operators (eg. ++, - (negation)), binary operators (eg. +, -, /, * ...). The ternary / tertiary operator is an operator that takes three arguments / expressions and returns the result after evaluating two expressions. It is also known as the conditional operator.

Syntax:

<expression1> ? <expression 2> : <expression 3>

The conditional operator ? : uses the boolean value of one expression to decide which of two other expressions should be evaluated. The conditional operator is syntactically right-associative (it groups right-to-left), so that a?b:c?d:e?f:g means the same as a?b:(c?d:(e?f:g)). The conditional operator has three operand expressions; ? appears between the first and second expressions, and : appears between the second and third expressions. The first expression must be of type boolean or Boolean, or a compile-time error occurs.

At run time, the first operand expression of the conditional expression is evaluated first; the resulting boolean value is then used to choose either the second or the third operand expression:

* If the value of the first operand is true, then the second operand expression is chosen.

* If the value of the first operand is false, then the third operand expression is chosen.

The chosen operand expression is then evaluated.

Eg.,

Consider the following if-else construct:

```
if (a > 0)
  x = a;
else
   x = (-a);
```

The tertiary equivalent:

```
x = (a > 0) ? a : (-a);
```

CLASSES

Class is a programming construct used to group related fields(variables) and methods(functions). Classes provide modularity and structure in a computer program. Code for a class should be relatively self-contained. Collectively, the properties and methods defined by a class are called members.

For eg.

Consider the class for points in two dimensions:

```
class point {
        double x,y;     //   Fields / variables.
        double distanceFromO() {   // distance from origin.   (method)
                return Math.sqrt(x*x + y*y);
        }
}
```

Classes are examples of user defined types created from inbuilt ones. The point class builds on two doubles (x,y).

Accessing a class variable

In order to access you use the dottend representation:

<class object>.<field / method>

For eg.

p.x; // Access the member variable x

p.distanceFromO(); // Call the distanceFromO function.

Classes used as members for other classes

Classes can be used as members in other classes also.

For eg. Consider the class triangle:

```
class triangle {
        point a,b,c;     //  Using earlier defined type point
        double getAverageDistance() {    //  Compute average distance.
                return ((a.distanceFromO() + b.distanceFromO() + c.distanceFromO()) / 3.0);
        }
}
```