Topics
- Declaring a Variable
- Assignment
- Initialization
- Input
- Conditional Statements
- Conditional Statement Flow Charts
- Logical Operators

Declaring a variable
A variable is an identifier that refers to a location in the computer’s memory where a value can be stored for use by a program. Variable names must be a legal identifier. The name can be of any length. They should usually start with a lowercase letter and have a meaningful name. If the name is several words concatenated together, all words after the first should have the first letter capitalized.

The general form is:
<data type> <variable name>;

or
<data type> <variable name1>, <variable name2>, …, <variable namen>;

The data type, for simple programs, will be one of the primitive data types: byte, short, int, long, float, double, char or boolean or a String.

For example:
int age;
double radius;
double area;
int result;
String output;
String message;

Assignment
After a variable is declared, you can assign it a value using the assignment operator (=). The variable should always be on the left hand side of the equals. The general format for assignment is:
<variable> = <value>;

For example:
age = 23;
radius = 3.2;
area = radius * radius * Math.PI;
result = x + y + z;
output = “Some text”;
message = “Acceptable”;

NOTE: The equals symbol is always used for assignment, never for comparison.

Initialization
Declaration and assignment can be combined into one step, typically called initialization. This is because you are declaring a variable and giving it an initial value.

The general form is:
<data type> <variable name> = <value>;}
For example:
int age = 23;
double radius = 3.2;
double area = radius * radius * Math.PI;
int result = x + y + z;
String output = “Some text”;
String message = “Acceptable”;

Input
For a detailed explanation, see the Lecture 3 notes. Input, whether through the Scanner class or JOptionPane, consists of two steps. The user must be prompted to do enter some value (and that result stored in a String with JOptionPane) and then the result must be stored in a variable that can be manipulated later. With JOptionPane, the input must be converted into a number. With the Scanner class, you must store the result into the appropriate data type.

JOptionPane
Integers:
String number;
int number1;

// read in number from user as string and convert to int
number = JOptionPane.showInputDialog(null, "Enter first integer");
number1 = Integer.parseInt(number);

Real numbers:
String number;
double number1;

// read in number from user as string and convert to int
number = JOptionPane.showInputDialog(null, "Enter first integer");
number1 = Double.parseDouble(number);

Scanner
Integers:
int number1;

// set up console input
Scanner scanner = new Scanner(System.in);

// read in numbers from user
System.out.print("Enter number 1 (an integer): ");
number1 = scanner.nextInt();

Real numbers:
double number1;

// set up console input
Scanner scanner = new Scanner(System.in);

// read in numbers from user
System.out.print("Enter number 1 (an integer): ");
number1 = scanner.nextDouble();

Using Conditional Statements
Use simple if statements when executing code for some special condition. If you want to perform a specific action based on a condition, use a simple if statement. This type of statement will allow you to execute code for only that condition. If you have two pieces of code, one that is executed if some condition is met and one that is executed if it is not met, use an if-else statement. This will allow you to execute one block of code for a certain condition and a completely different block of code for all other cases.
The *if-else if* statement has the ability to check multiple conditions and execute blocks of code based on which condition is met. Once a block of code is executed, the flow of control jumps to after the end of the *if* block (after the last brace). In the special case where you may want to test a condition across a number of conditional statements (divisibility of a number across the first 10 numbers), then you can have simple *if* statements one after the other.

```java
if (quizscore >= 7) {
    temp = "Student passed";
} else {
    temp = "Student failed";
}
```

```java
if (quizscore >= 7) {
    temp = "Student passed";
} else if (score >= 90.0) {
    grade = 'A';
} else if (score >= 80.0) {
    grade = 'B';
} else if (score >= 70.0) {
    grade = 'C';
} else if (score >= 60.0) {
    grade = 'D';
} else {
    grade = 'F';
}
```

<table>
<thead>
<tr>
<th>Simple (one-way) if statement</th>
<th><em>if-else</em> statement (two-way if)</th>
<th><em>if-else</em> if statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>int result = 3735;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>if (result % 2 == 0) {</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
|     System.out.println("divisible by 2");
| }                             |                                 |                        |
| if (result % 3 == 0) {        |                                 |                        |
|     System.out.println("divisible by 3");
| }                             |                                 |                        |
| if (result % 4 == 0) {        |                                 |                        |
|     System.out.println("divisible by 4");
| }                             |                                 |                        |
| if (result % 5 == 0) {        |                                 |                        |
|     System.out.println("divisible by 5");
| }                             |                                 |                        |
| if (result % 6 == 0) {        |                                 |                        |
|     System.out.println("divisible by 6");
| }                             |                                 |                        |
| if (result % 7 == 0) {        |                                 |                        |
|     System.out.println("divisible by 7");
| }                             |                                 |                        |
| if (result % 8 == 0) {        |                                 |                        |
|     System.out.println("divisible by 8");
| }                             |                                 |                        |
| if (result % 9 == 0) {        |                                 |                        |
|     System.out.println("divisible by 9");
| }                             |                                 |                        |

Multiple simple (one-way) *if* statements, every one of these conditions is evaluated.
Conditional Flow Charts
See the charts below for the flow of programs with conditional (if) statements.

One-Way if statement
Simple if statement

Two-Way if statement
If-else statement
Nested if statements
Converted to if-else if

(grade >= 90)?
letterGrade = 'A';

(grade >= 80)?
letterGrade = 'B';

(grade >= 70)?
letterGrade = 'C';

else
letterGrade = 'F';

System.out.println(letterGrade);

There can be as many conditions as necessary in this section.
Logical and Comparison Operators
Java has four Boolean operators: logical not (!), logical and (&&), logical or (||), and logical exclusive or (^). Java also has six comparison (relational) operators. The result of a comparison operator is a boolean (true or false). The comparison operators in Java are: less than (<), greater than (>), equal to (==), less than or equal to (<=), greater than or equal to (>=), and not equal to (!=). The comparison operators are used to perform tests that have boolean results. Those results can then be combined using the Boolean operators.

Testing if a variable is within a range can be done as shown:
```java
if (result > 0 && result < 12) {
    System.out.println("in range");
} else {
    System.out.println("out of range");
}
```

A value is in a range if it is greater than one value AND lower than another value. This boolean condition checks that the number satisfies both of those conditions.

You could also reverse this check to see if the value was out of range as shown:
```java
if (result < 0 || result > 12) {
    System.out.println("out of range");
} else {
    System.out.println("in range");
}
```

A value is out of range if it is greater than one value OR lower than another value. This boolean condition checks that the number satisfies one of the conditions, the other, or neither.

One can use the if-else if statement to categorize values into different levels:
```java
if (result < 0) {
    System.out.println("very low");
} else if (result >= 0 && result < 12) {
    System.out.println("low");
} else if (result >= 12 && result < 25) {
    System.out.println("medium");
} else if (result >= 25 && result < 35) {
    System.out.println("high");
} else {
    System.out.println("very high");
}
```

To compare equality, you must use the == operator, NOT the = operator. A single equals is an assignment statement, which will evaluate to true if the value is able to be store in the value. == compares the values to see if they are equal. Use the != operator to indicate not equal to.

```java
message = "Your guess (" + guess + ") is ";
if (guess == answer) {
    message += "correct!!";
} else if (guess != answer) {
    message += "incorrect";
} else if (guess < answer) {
    message += "low";
} else if (guess > answer){
    message += "high";
} else {
    message += "invalid";
}
```