Comments

```
// Single line comments.

/*
     Multi-line comments.
*/
```

```
/**
 *   JavaDoc comments.
*/
```

Primitive Data Types

Type	Content	Range of values	Size & format
char	single character	Any Unicode character	16-bit Unicode
boolean	**Boolean value**	**true or false**	**Unspecified?**
byte	integer	−128 to 127	8-bit 2's complement
short	integer	−32,768 to 32,767	16-bit 2's complement
int	**integer**	**−2,147,483,648 to 2,147,483,647**	**32-bit 2's complement**
long	integer	About −9e18 to about 9e18	64-bit 2's complement
float	rational	About −3.4e38 to about 3.4e38	32-bit IEEE 754
double	**rational**	**About −1.8e308 to about 1.8e308**	**64-bit IEEE 754**

Most commonly used types shown in **bold** and *a*e*b* means $a \times 10^b$.

Keywords and Reserved Words (cannot be used as variable names)

Reserved words: `true`, `false` and `null`.

Keywords:

`abstract`	`default`	`goto` *	`package`	`synchronized`
`boolean`	`do`	`if`	`private`	`this`
`break`	`double`	`implements`	`protected`	`throw`
`byte`	`else`	`import`	`public`	`throws`
`case`	`enum` ***	`instanceof`	`return`	`transient`
`catch`	`extends`	`int`	`short`	`try`
`char`	`final`	`interface`	`static`	`void`
`class`	`finally`	`long`	`strictfp` **	`volatile`
`const` *	`float`	`native`	`super`	`while`
`continue`	`for`	`new`	`switch`	

* not currently used, ** added in 1.2, *** added in 5.0

Arithmetic Operators

`+`, `-`, `*`, `/` and `%` (modulo)

Assignment and Increment Operators

`=`, `+=`, `-=`, `*=`, `/=`, `%=`, `++` and `--` (When applied to objects, = assigns the reference. The `clone()` method is typically used to duplicate object contents. Note, =+ and =− don't generate errors.)

Comparison Operators (take arithmetic "arguments," "return" *true* or *false*)

`==`, `!=`, `<=`, `>=`, `<` and `>` (Be careful, `==` compares object references not contents. To compare object contents, generally use `equals()` or `compareTo()` methods.)

Boolean Operators (take boolean "arguments," "return" *true* or *false*)

`&&` (and), `||` (or) and `!` (not)

Logical (Bitwise) and Shift Operators (generally used with integer types)

`&` (and), `|` (or), `^` (xor), `~` (compliment), `<<` (shift left), `>>` (shift right with sign extension), `>>>` (shift right without sign extension)

Selection Constructs

A *block* is a single statement or more than one statement enclosed in braces, i.e. {}.

```
if (boolean expression)
  then block

if (boolean expression)
  then block
else
  else block
```

```
if (boolean expression 1)
  block 1
else if (boolean expression 2)
  block 2
else, etc
  block 3, etc.
```

In nested *if statements* without additional braces, *else* clauses always associate with the "nearest" unsatisfied *if*.

```
switch (selector) {
    case constant1: statement(s);
    case constant2: statement(s);
    etc.
    default: statement(s);
}
```

The *selector* must be an integer type (includes *char*). Use `break` statements to prevent "fall-through." The "default" clause is optional.

Repetition Constructs

```
while (condition)
  block

do
  block
while (condition)
```

```
for (initialize; test; increment)
  block
```

While it is not necessary good practice, `break` statements can be used to exit and `continue` statements can be used to cycle (i.e., return to the *condition* or *test*) loops.

Exceptions & Assertions

```
try {
 statement(s);
} catch (ExceptionType e){
 statements(s);
} finally {
 statements(s);
}
```

Multiple catch clauses can be specified. The *finally* clause is optional.

Use `throw new ExceptionClass(messageString);` to create and throw an exception. Tag methods that may throw exceptions as follows: `visibility type methodName throws ExceptionClass(s) { … }`. Multiple exception classes should be separated with comas. The difference between checked and unchecked exceptions appears relaxed and throws clause appears optional in most (all?) cases as of Java version 1.5/5.0.

`assert boolean expression;` or `assert boolean expression: message;`.

Assert creates and throws an AssertionError if the *boolean expression* is false. If provided, the message is passed to the AssertionError constructor.