

Introduction

Jan. 3, 2000

Cosc 1030 is about:

- Software development
- Data structures
- Sorting/searching

In this course we'll deal mostly with data structures.

Course webpage is at <http://www.cs.yorku.ca/course/1030/>

The assignments are available @ ariel in the /cs/course/1030/ directory.

Java is:

“simple, object oriented, network savvy, interpretive, architecture neutral, portable, high performance, multithreaded, dynamic language” secure.

Portable: The code can be transported to any computer and it will run correctly due to the java virtual machine.

network savvy: to be able to open to, and communicate with other computers as easily as on the terminal it is run on.

Programs vs applets

Programs big: Applets small. Applets get downloaded and executed on web browsers. We will be concentrating on programs in this course.

Version:

1.2.

Java Development environments:

Sun JDK (Java Development Kit)
Sun Java Workshop,
IBM Visual Age for Java,
Code Warrior,
MS J++

We use the JDK in ariel.

```
javac - compiler, source code → bytecode
java  - executes java bytecode
javadoc - html documentation from the source code.
jdb   - java debugger.
```

Yay!: hello world example ☺

Comments

```
/* this is a huge comment */
/** this is a javadoc comment */
// this is a one line comment
```

Data Types

2 types:

- primitive
 - Byte
 - Short
 - Int
 - Long
 - Float
 - Double
 - Char
 - Boolean
- } integer types
- decimals
- longer decimals
- letters
- true/false

Reference

- Arrays
- String

Etc...

Variables

Must start with a letter (includes `_` and `&`)

Followed by letters or digits

Any length

Unicode

Can't use keywords (i.e. `for`, `while`)

e.g.: `int x;`

Final variables

These can not be changed after initialized.

e.g. `final double pi = 3.14;`

Operators

Arithmetic `+`, `-`, `*`, `/`, `% (mod)`, `++`, `--`

Relational: `>`, `<`, `>=`, `<=`, `==`, `!=`, `instanceof`

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

`&&` and `||` are short circuit, meaning if the condition is nullified before both terms are evaluated, the second term is not consulted.

Assignment =

e.g. `value = 42;`

shortcuts: `value = value + 7;`

is the same as `value += 7;`

Control flow (decisions)

If

If (expression) statement; or if long blocks are needed,

```
If (expression)
{ statement
}
if (expression)
{ statement
}
else
{ statement
}
```

SWITCH:

Switch is used to replace a series of if, elseifs. For example, if you have 47 different cases, don't use 47 elseifs... use a switch.

See switch example ☺

WHILE:

```
While (expression) // as long as <expression> is true
{ statement
}

do
{ statement
}
while (expression)
```

see while example ☺

FOR

```
For (initialization; termination; increment)
{ statement
}
```

e.g.

```
int sum;

for (int i = 0; i < 10; i++)
{ sum += I;
}
```

Style

Style dates back to original programming languages and makes code readable. E.g. use good variable names, indent to keep code structured...

Branching

`break;`
used to exit structures like loops and switches.

[More examples ☺](#)

Arrays

(note: arrays are a reference type)

e.g. `int[] arrayOfInts;` // arrayOfInts is of type of `int[]` (an array of ints)
e.g. `int[] arrayOfInts = new int[20];` // makes an array with indices from 0 - 19

In general:

`elementType[] arrayName = new elementType[arraySize];`

Arrays can be listed as well...:

`Int[] numbers = {3, 7, 26, 4, 672, 24};`

`array.length` refers to the length of array.

Arrays can be multidimensional:

[See array examples ☺](#)