**About Java**

* Java was made by sun microsystems
* Sun microsystems was bought by oracle
* A machine uses machine code which is a set of numbers the processor understands
* A set of instructions written in a programming language is a program
* IDE (NetBeans) → [Source code → compiler→ exe (machine code)] → OS → hardware
* Java source → [java bytecode → exe] → JVM (Java virtual machine)

**Java coding**

* Programming language → program → 1. Set of precise instructions (statements) 2. A statement executes one task only 3. A statement in java uses a delimiter (;)
* Comments 1. Documentation comments (/\*\*) end with (\*/) 2. Multi-line comments explain the program or sections of the program (/\*) end with (\*/) 3. End of line comments tells what one statement does. After the statement. (//) end with nothing
* Comments are important USE THEM!

**Title block**

* Will have 1. Your name and last name 2. Student number 3. Title 4. ICS3U-A 5. Date 6. Teacher's name
* Plug into line 8-ish

**Outputting**

* Print statement and println statement

**Escape**

* \t → tab
* \n → New line
* \” → output”
* \\ → outputs \

**Format command**

* System.out.format

**Data Types**

Primitive Data Types and Reference / Abstract Data Types

Primitive Data Types – Numbers – Boolean (Java assigns it a fixed amount of space)

Numbers:

1. Integers int
2. Long Integers long
3. Short Integers short
4. Floating Numbers float
5. Double (Decimal) double
6. Characters char

Reference / Abstract Data Types:

Used to write non-numbers String

Variables

1. Unknown value (number)
2. Changing numbers

Variable – Memory location in RAM

Identifier (Name)

**Variable Identifiers:**

1. Should be a noun corresponding to its function – ex. average\_mark, or averageMark
2. An identifier cannot have white space
3. Cannot have special characters (,.\*+-=/\()[]{}) in it
4. Cannot start with anything other than an alphabet
5. Cannot use reserved words in Java

**How to use variables:**

1. Variables need to be declared:

Syntax - < data type > <identifier>;

Examples: int student\_mark; double average\_mark; char alphabet; boolean flag;

1. Assigning a value to a variable:

Syntax - <variable identifier> = <value>;

Example: student\_mark = 95;

The equal sign is the assignment operator

Constant

Declaration Syntax

final <data type> <identifier>;

Example – final double PI;

PI = 3.14;

PI = 3.142; (This is not allowed)

**Scanner**

Importing Scanner: inport.java.util.Scanner;   
Opening new scanner: Scanner input = new Scanner (System.in);

Inputting using a Scanner:   
nextInt();

nextDouble();

**Arithmetic operations**

Operation:        Symbol        Priority

Multiplication       \*                1

Division                 \                 1

Subtraction             -                2

Addition                 +                2

Modules division    %         -   e.g. 25%4=1 (only gives remainder of division)

**Steps in programing:**

1. Read the problem
2. Identify input and output to and from the program
3. Follow the pattern for building a program

**Algorithm Building**

1. Pseudo Code -> Half English Half Java
2. Flow Chart
3. (=) -> Start/End
4. /=/ -> Input/output
5. Square -> Computation
6. Square with 3 arrows -> Yes, No, Decision
7. Down Arrow -> Flow Path
8. Circle with X -> Connector (Connects one part of the flow chart with the other)

Start > double, size, total cost, material cost, LABOUR RENT > Input Size > material cost = 0.05 \* size \* size > Connector > total cost = material cost + LABOUR RENT > O/P total cost > End

**Type Casting**

Lower Precision (int)

Higher Precision (double)

Lower precision (Up Cost) – Higher prescision

Higher -> Lower down cost

Lower -> Higher

Java does this intrinsically

Examples:

int divisor, dividend;

double quotient;

divisor = 2;

dividend = 3;

quotient = dividend / divisor; //Quotient is double and dividend and divisor are int

The above is an example of Lower Precision to Higher Precision

double num1, num2;

int quotient;

num1=2.5;

num2=1.0;

quotient = num1/num2; //Quotient is int and num1 and num2 are double

Java will give an error for the example above because you are trying to make a double an int

How to force through this error:

double num1, num2;

int quotient;

num1=3.0;

num2=2.0;

quotient = num1/num2; //Quotient is int and num1 and num2 are double

= (int) num1 / (int) num2; //Turning a double into int

Syntax

(<data type>) <identifier>

Rounding:

double num, original;

int rounded;

num = 234.5;

original = num; //Original number will be stored

num = 234.5+0.5;

rounded = (int) num; /\*Leaves the decimal out so if you are adding 0.5 to something like 234.3 it will be 234.8 but Java will round it to 234 because it leaves the decimal out.

**Decision Structures (Conditional Control)**

If Statement and Else Statements

Syntax:

if (<condition>) {

statement 1;

statement n;

} else {

statement 1;

statement 2;

}

**Computer Operators**

Operators Symbol

Equal to = = (Type together)

{We use 2 equal signs as it represents comparison between two numbers as opposed to 1 equal which assigns a value to a number}

Greater than >

Less than <

Operators Symbol

Greater than equal to >=

Less than equal to <=

Not equal to !=

**Math Library**

double base, exponent, result;

input base;

input exponent;

result = Math.pow(base, exponent);

output result;

**Random Method**

double value; // Logic error if int since number is between 0-1

value = Math.random(); // ( ) brackets include of number between 0-1

Random number in a range

(high – low)

(int)( (high – low) \*Math.Random()+low)

5-15

int randomNum;

randomNum = (int) (11\*Math.random()+5); // 11 because 15-5 + 1 (10 + 1 = 11)

2-6 // 6-2 +1 = 5 (5\*Math.random()+2)

**String**

Length():

int world\_length;

string word;

word = “Name”;

word\_length = word.length();

// Word length is 4 because string (Name) has four letters

The index value is the number of letters there are but it is always 1 less because it starts at 0

Substring:   
Length():

int world\_length;

string word;

word = “Name”;

word\_length = word.length();

string part\_word;

part\_word = word.substring(1, 3);

part\_word = word.substring (1);

**Comparing Strings**

String one, two;

one = “Good day”;

two = “good day”;

if (one.equals(two)) { // Is boolean

System.out.println(“Strings are equal”);

}

else {

System.out.println(“Strings are not equal”);

}

**Methods**

\* Method needs to have a purpose

\*I/O happens inside method

\*Input happens outside while output happens inside

\* Both I/O happen outside

How to write method:   
public static void < name > () {

statement 1;

statement 2;

}

**Method Parameters**

Method parameters are inputs to the method

public static void main {

something (input 1, input 2);

}

public static void something (variable 1, variable 2); /\*If you do 2 variables here you need to do 2 when you do it in main method \*/

Syntax:

public static void <Method identifier> (<Data Type> <Parameter identifier>)

If you want another input, then do the data type and parameter identifier again with a comma

**Arrays**

* Arrays allow you to store more than one piece of data
* Arrays are used by adding [] to a type of variable

Array example:

double [ ] ave\_temp;

ave\_temp = {7.6, 5.2, 1.2, 0.5, -1.2, 3.5, 1.8}

* If you want to know where a number is in an array you use index values which is put in the square brackets such as ave\_temp[5]
* Note: First index value is always 0

Syntax:

<data type> [ ] <identifier>;

e.g.

String [ ] friends;

double [ ] ave\_temp;

int [ ] student\_marks;

* Use arrays when you want to use more then one number in a same variable
* If you read a problem and there is more then one piece of data needed use an array

If you want to restrict amount of pieces data stored then do this:

String [ ] friends;

friends = new Sting [5]’

friends = {“John”, “Tracy”, “Shivam”, “Joshua”, “Sahil” };

**Character Data Type**

Everything except functions keys (F1, F2) are character data types

char letter;

char = ‘a’; // When assigning a value to char you only use one quote not two like string

A character stores the number value from the Unicode table (Advanced ASCII)

letter2 = ‘A’;

if (letter1 > letter 2) {

System.out.println(“A comes before a in the Unicode table”);

Casting String to Array

string word;

char [ ] letters;

char alpha;

word = (“Holidays”);

letters = word.toCharArray();

letters = {H, o, l, i, d, a, y, s, !}

0 1 2 3 4 5 6 7 8

System.out.println(letters[4]); // Prints value at 4 which is d