project
└ package
  └ class

Unit 1 - Intro to Java

(02) - Java Basic Syntax

HelloWorld.class

```
public class HelloWorld{
    public static void main(String[] args){
        System.out.println("Hello World");
    }
}
```

Comments - /**/ , //

Javadoc Comments

```
/**
 * @author name
 * @param name description
 * @return description
 * @version number
 */
```

Output

System.out.print() ← no newline, output vars using '+' } "~~" + x + "~~"
System.out.println() ← same as ↑ but automatic newline }
System.out.printf() ← format string. Add '\n' for newline.
↳ System.out.printf("format string", parameter(s));

↑ %d integer
%f double
%s string

printf width
%Wd - W chars wide (right)
%-Wd - left aligned
%Wf - W chars wide(.)

printf precision
%.Df - rounded to D digits
%W.Df

(03) - Primitive Data & Definite Loops

Primitive types: int, double, char, boolean
3 3.5 'h' true

String ← capital letter because class
↖ an array of chars

```
String food="cookie";
char firstLetter = food.charAt(0); //'c'
food = food.toUpperCase(); //"COOKIE"
int lengthFood = food.length(); //6
```

Casting
(char) ('a'+2) is 'c'

Variables - conventions
↳ camelCase (helloThere)
→ No spaces
→ No starting with digit
→ No reserved words

Declaring:
dataType varName = initialVal;
Constants: final double PI = 3.1459;
↳ Always written in capitals
→ Initialized when declared
→ Keyword - final

(04) - Java Buffered Reader & Scanner

Buffered Reader (no issues regarding consecutive lines, \n whitespace is taken care of

```
import java.io.*;
public class MyFile{
    public static void main(String[] args) throws IOException{
        BufferedReader cin = new BufferedReader(new InputStreamReader(System.in));
        System.out.print("Enter your name: ");
        String myName = cin.readLine();
    }
}
```

* You can only input STRINGS *
* You can PARSE into other types:
↳ Integer.parseInt()
`int toNum = Integer.parseInt(stringNum);`
↳ Double.parseDouble()
`double toNum = Double.parseDouble(stringNum);`

Scanner (if nextLine(); after previous nextLine() - no issues
if nextLine after next()/nextInt/nextDouble() - you have to read in previous line ws.

```
import java.util.*;
:
Scanner console = new Scanner(System.in);
System.out.print("How old are you? ");
int age = console.nextInt();
```

↳ Methods
nextInt()
nextDouble()
next() ← char
nextLine() ← Str

(06) - Methods & Casting

Methods

```
public static returnType name(type param, ...){
    statement(s);
    ...
    return expression; //of type returnType
}
```

To call:

```
returnType variable = name(value, ...);
```

Casting (type) expression

```
double result = (double) 19/5;   //3.8
int result2 = (int) result;      //3
int x = (int) Math.pow(10,3);    //1000
double result = 19/5;            //3.000
double result = 19.0/5;          //3.8
```

Java Math

- Math.abs(value);
- Math.ceil(value);
- Math.floor(value);
- Math.log10(value);
- Math.min(v1, v2);
- Math.max(v1, v2);
- Math.pow(base, exp);
- Math.random(); //double b/w 0, 1
- Math.round(val); //nearest int
- Math.sqrt(val);
- .sin() .cos(v), .tan(v)
- .toDegrees(v), toRadians(v)

Constants:

```
Math.E   //2.7182818...
Math.PI  //3.1415926...
```

(07) - Loops & Ifs

For loops - same as C++
If statements - same as C++
Nested loops - i = row, j = col
Scope - the part of the program where a variable exists

Shorthands - +=, -=, *=, /=, %=
Inc/Dec - ++, --
Relational - ==, !=, <, >, <=, >=
Logical - &&, ||, !

(08) - Logic, while, do while, random numbers

While:

```
while(test){
    ...
}
```

Do-while:

```
do{
    ...
}while(test);
```

Sentinel val - 999, 'y/n', -1

Short-Circuit Evaluation: Java stops evaluating a test if it knows the answer (&& stops if anything false, || if anything true)

De-Morgan's Law: a&&b $\xrightarrow{\text{neg}}$!(a&&b) = !a || !b
a||b $\xrightarrow{\text{neg}}$!(a||b) = !a && !b

Random Numbers

Math.random();

```
int aNumber = (int)(Math.random()*6)+1;  //1-6   (inclusive)
            = (int)(Math.random()*6)+2;  //2-7
            = (int)(Math.random()*3)+7;  //7-9
            = (int)(Math.random()*100)+1; //1-100
```

The range ends & includes at b+(a-1)

```
(~~()*b) + a
(~~(b)) + a
```

Random Class (java.util.*) — so max-1 inclusive

- nextInt() - Random int
- nextInt(max) - Random int in range [0, max)
- nextDouble() - Random real n in range [0.0, 1.0)

```
Random rand = new Random();
int n = rand.nextInt(20)+1;  //1-20
      = rand.nextInt(8)+23;  //23-30
      = rand.nextInt(5)*2+4; //4-12 but even only
```