Introductory Java, part 1

J1

- Imperative programming
- Java syntax
- Types and expressions
- Functions

Programming paradigms

- What is a program?
  - A set of definitions and expressions to evaluate (*functional*).
  - A sequence of instructions for the computer to carry out (*imperative*).
  - A set of rules and facts, and a query to prove (*logic*).
  - Code organised into packages / modules / classes / methods / procedures (*at scale*).
- Almost any practical programming language combines features of several paradigms.

	Java	Haskel	Python	C	C++
Imperative	√	×	√	√	√
Statically typed	√	√	×	√	√
Object-oriented	√	×	√	×	√
Class-based	√	×	~	×	√
Memory-safe	√	√	√	×	×

- **Imperative** languages describe computation as a series of statements that transform state.

  But, most practical languages allow for a mix of functional and imperative programming.

	Java	Haskel	Python	C	C++
Imperative	✓	×	✓	✓	✓
Statically typed	✓	✓	×	✓	✓
Object-oriented	✓	×	✓	×	✓
Class-based	✓	×	~	×	✓
Memory-safe	✓	✓	✓	×	×

- **Statically typed** languages declare/infer the type of variables and expressions, ensuring that certain errors do not happen at run time.

  But, many statically typed languages (Java, for example) allow explicit run time type casting.

	Java	Haskel	Python	C	C++
Imperative	√	×	√	√	√
Statically typed	√	√	×	√	√
Object-oriented	√	×	√	×	√
Class-based	√	×	~	×	√
Memory-safe	√	√	√	×	×

- **Object-oriented** languages allow data, and code that operates on it, to be structured into "objects".

	Java	Haskel	Python	C	C++
Imperative	√	×	√	√	√
Statically typed	√	√	×	√	√
Object-oriented	√	×	√	×	√
Class-based	√	×	~	×	√
Memory-safe	√	√	√	×	×

- **Class-based** languages allow (or require) objects to be instantiated from a template, called “class”.

  Java is strict about this: all code lives in some class.

	Java	Haskel	Python	C	C++
Imperative	√	×	√	√	√
Statically typed	√	√	×	√	√
Object-oriented	√	×	√	×	√
Class-based	√	×	~	×	√
Memory-safe	√	√	√	×	×

- **Memory-safe** languages restrict use and modification of data according to how it was created.

- Does this mean Java is “best”? No.
- So why Java?
  - Allows you to practice several paradigms.
  - Widely used, for example in
    - servers & enterprise software; and
    - android apps.
  - Fairly well supported:
    - portable;
    - rich standard (and 3rd party) library.
- What other programming languages are there? `https://www.rosettacode.org/`, for example.

Syntax

- The syntax of a programming language defines the rules for what is a valid program.
- Programming language syntax is *precise* – any mistake can result in a program that cannot be compiled or run.
- Tools (IDE, compiler) will help you get it right – use them!
- Java is *case sensitive*: upper and lower case letters are distinct.

Coding
Hello world

```java
// package name must match source file path
package comp1110.lectures.J01;

// class name must match source file name
// (without .java)
public class HelloWorld {

  public static void main(String[] args) {
    // program starts here!
  }

}
```


Reminder

- Oracle's Java tutorial:
  `http://docs.oracle.com/javase/tutorial/java/index.html`
- Java SE documentation:
  `https://docs.oracle.com/en/java/javase/17/docs/api/index.html`
- Keep a quick reference handy:
  `http://introcs.cs.princeton.edu/java/11cheatsheet/`
- We're not expecting you to memorise all of Java up front!

Types

- Java is a **statically typed** language.
- What is a type?
  - A set of possible values (*mathematical*).
  - A type gives an interpretation to data (and a size).
  - In code, a type is a declaration of intent.

The two kinds of types in Java

- Fixed set of primitive types:
  - Integers (`byte`, `short`, `int`, `long`).
  - Decimal numbers (`float`, `double`).
  - `boolean` (values `true` and `false`).
  - Character (`char`).
  - The type of nothing (`void`).
- Everything else is classes.

Expressions

- A unit of code that evaluates to a single value.
  - Literals (`0`, `27.3`, `"Abc"`, `true`)
  - Variables
  - Compound expressions, using
    - Operators (`+`, `-`, `*`, `/`, `==`, `<=`, `<`, `...`)
    - Method (function) calls
    - Conditional expressions (J5)
  - Object creation (O1)

- Every expression has a type.
- The type of an expression determines what operators or methods can be applied to it (and sometimes what they do – this is called *polymorphism*).
- The operator or method determines the type of the result.

Variables

- Associates a name with a value.
  - Declare and initialise `int total = 4 + 5;`
  - Declare `int total;`
  - Reassign `total = total - 1;`
- Variables hold the state that is transformed by statements in an imperative program.
- Make code easier to read by using variables to name intermediate steps in a complex expression.

[Coding Percentage]

Numeric types: Integer

- Why there are four integer types?

	size	range		
`byte`	1	$-(2^7)$ to $2^7 - 1$	(-128 to 127)	
`short`	2	$-(2^{15})$ to $2^{15} - 1$	(-32768 to 32767)	
`int`	4	$-(2^{31})$ to $2^{31} - 1$		`17`
`long`	8	$-(2^{63})$ to $2^{63} - 1$		`17L`

- The default type of an integer literal is `int`.
- Bounded integer types can *overflow*.

Numeric types: Decimal

- The `float` and `double` types are floating-point representations of decimal numbers.
- Limited *range and precision*:
  - Min/max value: $\pm 1.79 \cdot 10^{308}$.
  - Smallest non-zero value: $2.22 \cdot 10^{-308}$.
  - Smallest value > 1: $1 + 2.22 \cdot 10^{-16}$.

  (for `double`; `float` is smaller).
- Special values $\pm$ `inf` (infinity) and `NaN` (not a number).
- The default type of a decimal literal (`0.0`, `3.14`) is `double`.

Strings

- Type `String` is not primitive (it's a class).
- Special syntax for string literals: text in double quotes (") creates a new `String` object:

```
String recipient = "world";
```

- Concatenate with +:

```
String message = "Hello "+ recipient;
```

- Every type (primitive and class) in Java has a method (`toString`) that creates a "printable representation" of values of the type.
  This is invoked automatically in certain cases.

Functions

- In programming, a function (also known as “procedure”, “subroutine”) is a piece of a program that is given a *name*, and can be *called* by that name.
- Use of functions promote *abstraction* (“what, not how”) and help break a complex problem into smaller parts.
- To encapsulate computations on data, functions may have parameters and a return value (but also “side effects”).

“static methods”

- Functions/procedures in Java (like many other object-oriented languages) are called methods.
- For now, we will declare all methods `static`, meaning, roughly, that they depend only on their arguments.

- (static) method definition in Java:

```
static int percent(int a, int b) {
  // method body
  return ...;
}
```

  - keyword `static`
  - return type (here, `int`)
  - method name (here, `percent`)
  - list of parameters, each with type and name, in parentheses
  - method body, in braces (`{ }`)
  - If return type is not `void`, the method body must include a `return` statement.

- method call

```
int v = percent(x + y, total) * 2;
```

  - method name followed by arguments in parenthesis;
  - argument expressions much match parameters in type and number.
- The function call is itself an expression, of the method's return type.

Coding
Percentage & Units

Testing

- We write functions to be general, so how do we know they are, in general, *correct*?
- Test each function in isolation ("unit testing").
  - Document assumptions.
  - Test a *variety* of cases.
- What are "edge cases"?
  - Typical (numeric) examples: values equal to/less than/greater than zero; very large and very small values; values of equal and opposite signs; etc.
- More about (unit) testing later in the course.

preview (O1): Non-static methods

- Non-static (also known as “instance”) methods are called on an object, and can access (depend on, and modify) that object, as well as their arguments.
- Instance method call in Java:

```
String s = "Hello hello";
char c = s.charAt(s.length() / 2);
```

  - an expression that yields an object;
  - a dot (“.”);
  - the name of the method, followed by arguments in parentheses.