COMP6700/2140 First Program

Alexei B Khorev and Josh Milthorpe

Research School of Computer Science, ANU

22 February 2017

Topics

You can never teach anybody anything. It's impossible. But we should try.
[A. A. Stepanov]

What we will do in this part:

1. What is programming?
2. *Java*'s story
3. Java as a language and as platform
4. Our First Java Program

What is Programming?

- Act of writing a structured text — a computer code
- Following well-defined and (usually) precise rules
- Converting this code into an executable
- Making sure it meets specification

The most important part (probably) is the first — creating a structured text, a *computer program*.

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

To represent a complete program written in the Java programming language, the above text should be saved in a file called `FirstSample.java`

A program is an interface between human mind and machine

What is Java?

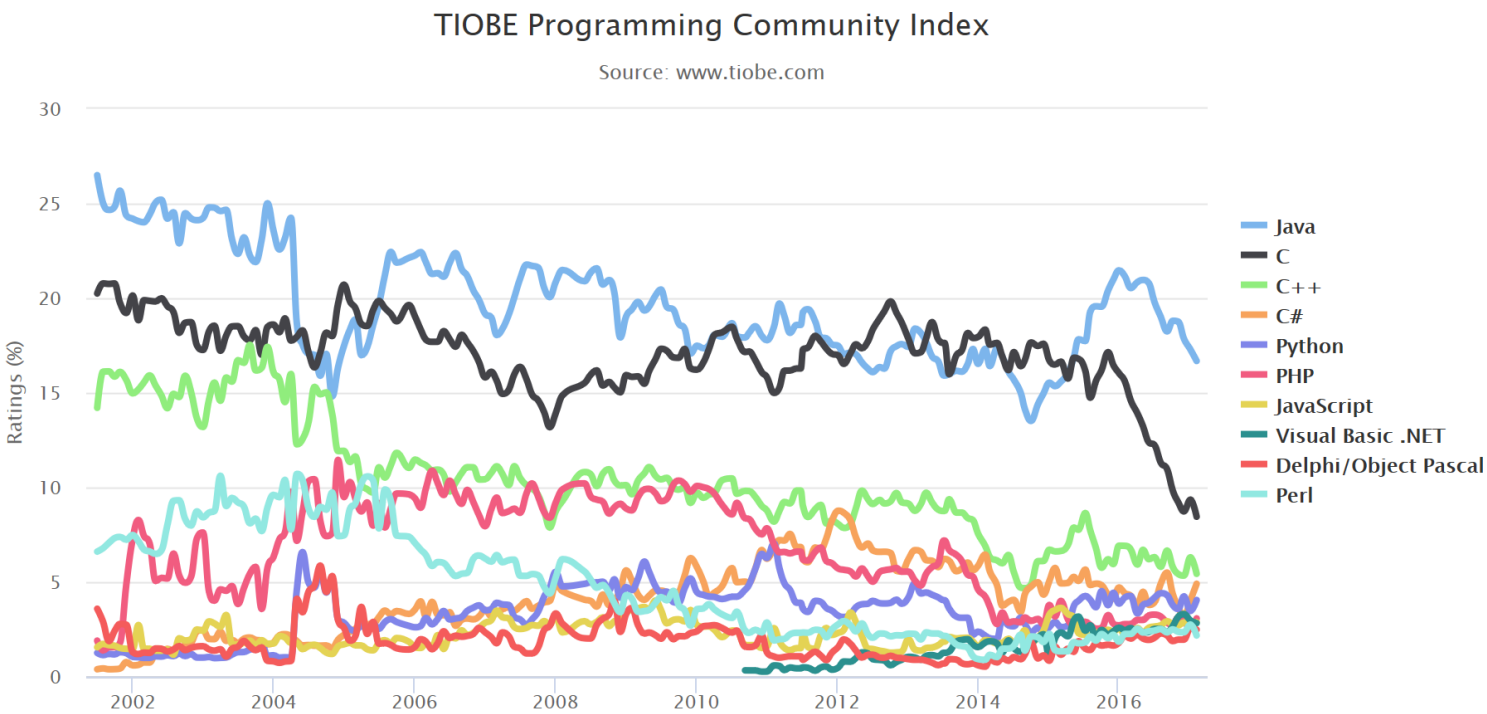
1. A programming language : Modern OO language, mature (20 years old), very well supported (extensive, often high quality libraries), actively developed (9 major releases; the last — Java SE 8 — has been finally released in March 2014, the next is expected 2017?)
2. A software development platform (language + SDK + API): includes the language itself, a rich standard library, libraries for creating all major types of software artefacts: Graphical User Interfaces(GUI), Networking, Database Interactions, Web Applications, Encryption, Text Processing and so on, and necessary tools for compiling, optimising, testing, transforming and integrating your code into working applications.
3. A software deployment and runtime environment: *JVM*: **J**ava HotSpot **V**irtual **M**achine is the core of Java software platform: everything — compiling, debugging, executing, etc. — is done by the JVM.
4. Java philosophy : *"write once, run anywhere"*: get started quickly, write less code, write better code, develop programs quicker, avoid platform dependencies, distribute easier
5. Java™ (the trade mark)

Java's current position in the industry

According to TIOBE latest data:

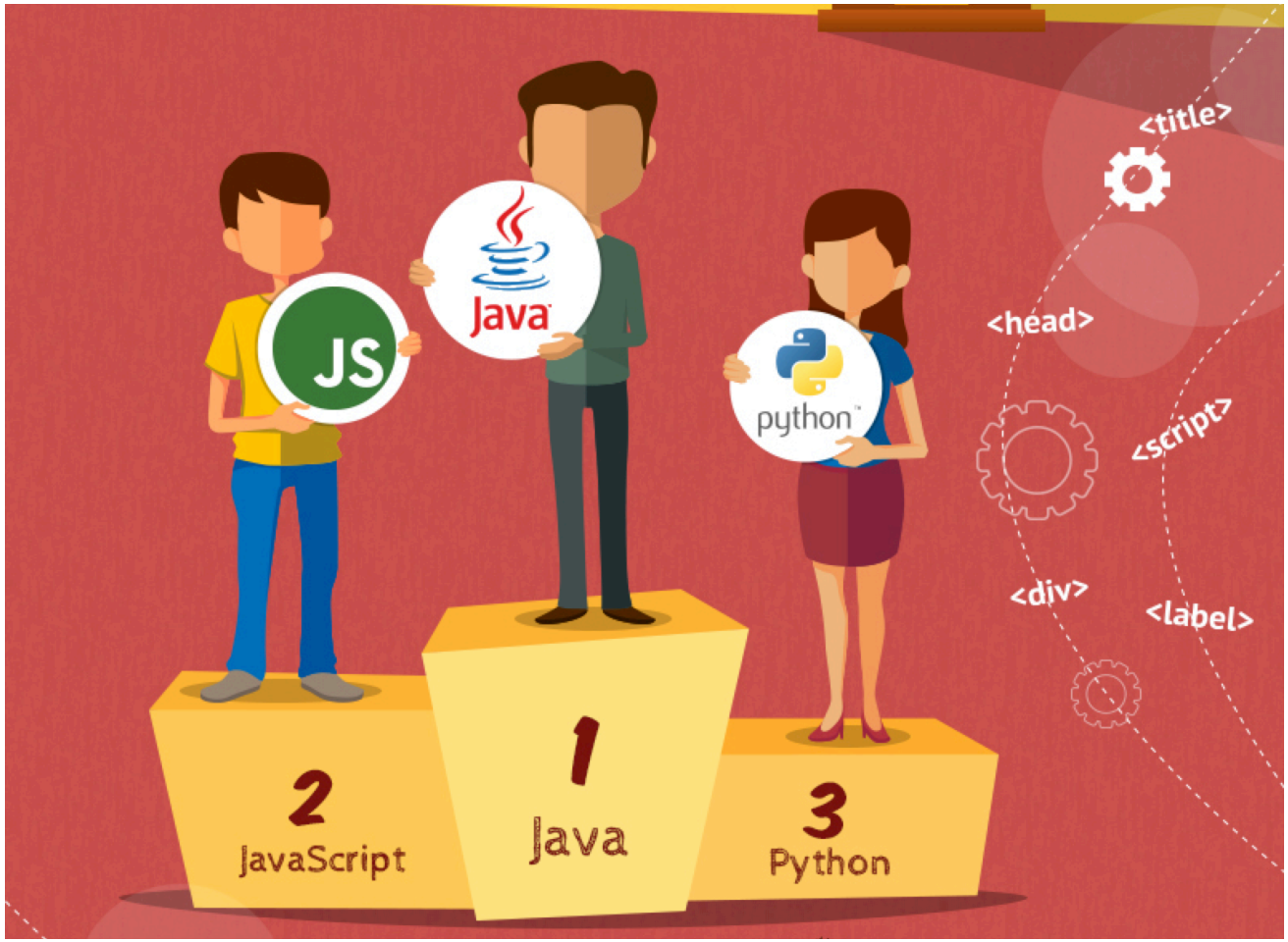
Feb 2017	Feb 2016	Change	Programming Language	Ratings
1	1		Java	16.676%
2	2		C	8.445%
3	3		C++	5.429%
4	4		C#	4.902%
5	5		Python	4.043%
6	6		PHP	3.072%
7	9	▲	JavaScript	2.872%
8	7	▼	Visual Basic .NET	2.824%
9	10	▲	Delphi/Object Pascal	2.479%
10	8	▼	Perl	2.171%

Long-Term Trends



Java is the champion!

Java has been the Programming Language Hall of Fame winner in 1997, 2005 and 2015 — the last two times after new significant language extensions.



Elements of a Computer Program

A computer program is a document, contained in a file or a collection of files, which is written in strict accordance with the rules of a programming language. The purpose of a computer program is to describe a computation. We build a complex computation as a combination of simpler ones. The language we use to achieve this should therefore provide us with:

- *primitive expressions*, which represent the simplest entities the language is concerned with
- *means of combination*, by which compound elements are built from simpler ones, and
- *means of abstraction*, by which compound elements can be named and manipulated as units.

(in words by Harold and the Sussmans:)

We seek a language which will allows us:

- Naming — *variables* (symbols which will represent data)
- Choosing — *conditionals* (decision rules which determine what actually happens during execution)
- Repeating — *looping* (means of describing a multitude of executions in simple terms)
- Grouping — *functions* (also called routines, procedures, methods)

Structure of a Computer Program

From the book "The Java Programming Language":

- A program starts as a sequence of characters contained in a file — the source code.

Structure of a Computer Program

From the book "The Java Programming Language":

- A program starts as a sequence of characters contained in a file — the source code.
- Interpreting those characters according to the rules of a given language, is a job of the special program called compiler, or interpreter.

Structure of a Computer Program

From the book "The Java Programming Language":

- A program starts as a sequence of characters contained in a file — the source code.
- Interpreting those characters according to the rules of a given language, is a job of the special program called compiler, or interpreter.
- Some characters will represent the names of *variables*, others will be special keywords used by the language, still others will be *operators* or "punctuation" characters used to separate the other elements.

Structure of a Computer Program

From the book "The Java Programming Language":

- A program starts as a sequence of characters contained in a file — the source code.
- Interpreting those characters according to the rules of a given language, is a job of the special program called compiler, or interpreter.
- Some characters will represent the names of *variables*, others will be special keywords used by the language, still others will be *operators* or "punctuation" characters used to separate the other elements.
- These textual constructs form the *lexical elements* of the program. The lexical elements must be identified as keywords, comments, literals, variables, operators, or whatever else is appropriate...

Java — Compiled or Interpreted?

The program is only the beginning; it needs to be executed on the computer hardware. Depending on implementation, a programming language can be

- **compiled** — a special program, called a *compiler*, translates the *whole program* into the machine code. Machine code is a set of instructions executed *directly* by the processor; such execution is fastest.
- **interpreted** — a special program, called an *interpreter*, reads the program text, statement by statement, and executes them successively. This eliminates the need for a separate compilation phase. The program can be run on any platform which has the required interpreter, but the code executes (much) slower.

Java is a hybrid: `javac` produces a bytecode, which is like an assembly code, but architecture neutral. Java code compiled on a *Linux* machine will run on a *Windows* machine *without* recompilation (this was important for the applet technology — the server could send a required applet which the client browser could execute immediately). The Java interpreter (`java` command) executes Java bytecode. The same bytecode runs on any platform with a JVM. Modern JVMs, like *HotSpot* perform *Just in Time Compilation* to convert frequently-executed bytecode to machine code.

Types of Programming Errors

The path from an empty file to a correctly functioning program can be tortuous. Things can (and usually do) go wrong, even when it appears that everything is right. Remember, a program code is for human understanding as much as for the machine.

At the beginning, your programs inevitably will be:

- *Syntactically incorrect* — grammar violation: wrong or undefined symbols, sub-blocks grouped in wrong way. "Yoda speak", incomplete sentences. Syntax errors are revealed at compile time.

Types of Programming Errors

The path from an empty file to a correctly functioning program can be tortuous. Things can (and usually do) go wrong, even when it appears that everything is right. Remember, a program code is for human understanding as much as for the machine.

At the beginning, your programs inevitably will be:

- *Syntactically incorrect* — grammar violation: wrong or undefined symbols, sub-blocks grouped in wrong way. "Yoda speak", incomplete sentences. Syntax errors are revealed at compile time.
- *Semantically incorrect* — impossible to assign a meaning to a syntactically correct sentence. "Colourless green ideas sleep furiously."

Types of Programming Errors

The path from an empty file to a correctly functioning program can be tortuous. Things can (and usually do) go wrong, even when it appears that everything is right. Remember, a program code is for human understanding as much as for the machine.

At the beginning, your programs inevitably will be:

- *Syntactically incorrect* — grammar violation: wrong or undefined symbols, sub-blocks grouped in wrong way. "Yoda speak", incomplete sentences. Syntax errors are revealed at compile time.
- *Semantically incorrect* — impossible to assign a meaning to a syntactically correct sentence. "Colourless green ideas sleep furiously."
- *Logically wrong* — produce results which cannot be true. "John is married bachelor." This is a "famous" *GIGO* principle — Garbage-in–Garbage-out.

Types of Programming Errors

The path from an empty file to a correctly functioning program can be tortuous. Things can (and usually do) go wrong, even when it appears that everything is right. Remember, a program code is for human understanding as much as for the machine.

At the beginning, your programs inevitably will be:

- *Syntactically incorrect* — grammar violation: wrong or undefined symbols, sub-blocks grouped in wrong way. "Yoda speak", incomplete sentences. Syntax errors are revealed at compile time.
- *Semantically incorrect* — impossible to assign a meaning to a syntactically correct sentence. "Colourless green ideas sleep furiously."
- *Logically wrong* — produce results which cannot be true. "John is married bachelor." This is a "famous" *GIGO* principle — Garbage-in–Garbage-out.
- *Poorly written* — "There were two cars on the road: one was green, and another was turning left." OK for computer to execute, but pain for programmers to read and use. This is called a *code smell* (if code smells, it needs *refactoring*).

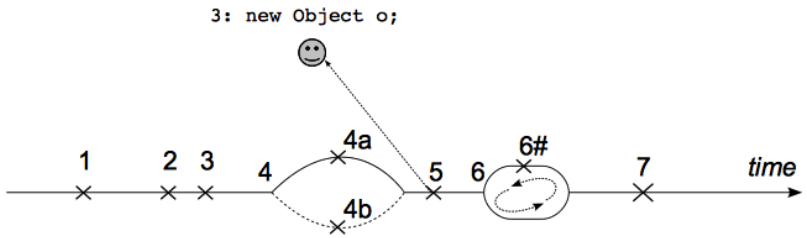
Threading

A (single-threaded) computer program is a sequence of instructions (*statements*) which are executed *sequentially*, one after another when the program is run. Many modern programs are *multi-threaded*. Their structure is (significantly) more complex. We shall have a glimpse of multi-threaded programming later.

Program as text

```
   public static void main(...) {
1      statement1;
2      statement2;
3      ObjectA o = new ObjectA();

4      if (cond)
4a        statementA;
       else
4b        statementB;

5      o.method();

6      while (loopCond) {
6#        repeatStatement;
       }

7       finalStatement();
}
```

Program as process



Our First Object (let's ignore it)

- Although the most simple Java program uses objects, we will not pay attention to them, instead focusing on the program run-time structure.
- First application will have only one *class* (one cannot write Java code without classes altogether).
- Our simple Java program is similar to a program written in a procedural programming language (C, Fortran, ...). The focus of our interest will be the fundamental building blocks of Java, which you will have to use when practicing writing simple programs in the labs.
- Java is cluttered with object oriented terms, it is not possible to avoid mentioning them (particularly when you need to call library methods in other classes). For the moment, consider those elements of syntax which look obscure as a linguistic pattern that you just copy and that you will try to understand later on.

Essential Elements of Java Code

Loosely speaking, the world of a Java program is populated with:

- Data (values and references) which are actual information carriers; the program uses this information and generates a new one
- Types and Variables (each variable has certain type)
- Declarations, Assignments and Initialisation statements
- Operators (act on data and variables)
- Control Flow Keys
- Class/Object Fields and Class/Object Methods
- Arrays and Strings (special reference types, used very often)
- Comments (irrelevant for program behaviour, but crucial for human reader)

We shall start with the bare essentials: studying the syntax and semantics of the core language. We will type code into an editor, compile it and run the byte code interpreter. Our simple program will read and write text to a terminal (shell) window (or, using simple pop-up windows).

Our First Java Program — 1

- Type the following text (never mind the white space) into the editor window

```
1 public class FirstSample {
2     public static void main( String[] args ) {
3         System.out.println( "Hello World" );
4     }
5 }
```

  (the line numbering is fictitious, it's **not** part of the code!)
- Save it as `FirstSample.java`; the name of the file **must** match the name which follows the word "class" on the first line — Java is case-sensitive!)
- Type `javac FirstSample.java` in your terminal (don't forget to hit "return" button, ↩)
- If you entered the code correctly the compiler will produce the `FirstSample.class` file
- Run (execute) the FirstSample.class file by typing `java FirstSample` ↩ . You should see the following output:
  `Hello World`

After that the program will exit (terminate).

Our First Java Program — 2, Code Anatomy

- Line 1: public is an *access modifier*. It determines what other parts of the program can have access to the class (or class members).
- Line 1: class defines a code grouping. Everything in Java lives inside classes.
- Line 1: FirstSample is the name of our class.
  - Note capitalisation (this is not a syntax rule, but a coding style convention).
  - Must begin with a letter and then can have any combination of letters and digits. Length unlimited.
  - Cannot use a Java reserved word.
  - Convention is that class names are nouns with initial capital letters (again — **Java is case sensitive**).
- The file for the source code **must** be of the same name as the public class with the extension `.java` added (`FirstSample.java`)
- Line 1 (the last symbol): Left hand curly bracket (or brace) defines the beginning of a code block. Matching { and } is important, the indentation is not (but the code layout is important for readability).

Our First Java Program — 3, Code Anatomy (*cont.*)

- Line 2 : When you compile a class and run it, execution always starts at the beginning of the method main.
  - You **must** have a main method in order for the code to execute
  - The main method must be `public`, `static` and `void`
- Line 2 (String[ ] args) : The brackets contain the argument list for `main` — the data passed to this method.
  - `main` **must** have a single argument of type String[ ] (array of strings).
  - This complete line is typed at least once in every Java program (and is annoying for toy code like this):
    - `public static void main( String[] args )`
- Line 3 : This sentence is the body of the method and is a Java *statement*.
  - In Java, all statements must end with a semi-colon character `;`
  - Can be multi-lined. There are no limits on the length of a statement.
- Line 3: The System.out object is contacted and asked to use its `println(...)` method. The syntax for method calls is `o.mname(parameters)`, where `o` is the name of the object (or class).

Our First Java Program — 4, Code Anatomy (*concl.*)

- Line 3: The single argument of `System.out.println` is a string constant (enclosed in double quotes).
- Line 3: Finally, the word `static` in the `main` method declaration is a *scope modifier*, meaning that the method belongs to the whole class, and does not require an instance of the class to be called (unlike not-*static* methods which need an instantiated object to be called). Other modifiers exist in Java, and we shall meet them in due time.

```
/** The very first Java program with added comments
 *  @author abx
 *  @version 1.0 */
public class FirstSample {
    /* we need a main method to run it as a standalone app */
    public static void main( String[] args ) {
        // to say "G'day mate" is more authentic
        System.out.println( "Hello World" );
    }
}
```

Our First Java Program — 5

What we can add to the code without changing its behaviour :

A very important part of every code in every language are *comments*. They are totally irrelevant to the compiler, but they mean a great deal to a human who writes and reads the code. Comments do not affect the executable code at all but can increase the readability of the source. You can add as many comments as you like but too many comments can make the code unreadable as well. Look at good code examples to get a feel for appropriate comments. There are three kind of comments in a Java code :

- One-line comments : everything from `//` to the end of line.
- Block comment : everything between the pair of `/* ... */`
- *Doc comments* : `/** ... */` ; they may contain special *javadoc tags*, which begin with `@` symbol. We shall study the use of Doc comments later; they are used to generate the class documentation automatically using `javadoc` command

Block and Javadoc comments do not nest!

Try adding some comments of these kinds to the above code — neither compilation, nor execution should change.

Families of Programming Languages

Several hundred programming languages exist. Why so many? How do they differ apart from syntax features (which may or may not be accidental)? There are several programming paradigms and styles:

- **Imperative** — defines a program as a step-by-step description of changes in computational environment
- **Declarative** — defines the indented result of computation (and leaves for language implementation to obtain it)
- **Functional** — defines computation as a (set of) function which transforms an input into an output, without side effects
- **Procedural** — (related to imperative) focus on defining a program as a sequence of *procedures* (also called *functions*) to be carried on data; *algorithms* are of primary interest
- **Object-oriented** — structures code as a set of objects which tightly couple data with the code that transforms it

Java is a procedural *and* object-oriented language. The new features added in Java SE 8 (in 2014) allow for some (limited) form of the functional programming style. Java is a multi-paradigm programming language.

Modern Java

Java provides support for creating robust distributed systems. The design principles which Java strives to implement are:

- **Simplicity** (?, was until *generics* were introduced in 2005, or λ-expression in SDK8) — a programming language has to be easy to learn, use and understand (*eg*, Java includes an automatic *garbage collection* that simplifies programming)
- **Robustness** — to achieve security and safety, the language has a *strong type system* to catch more errors during compile time; use of *pointers* (memory addresses) is not allowed
- **Portability** — the programs need to be runnable on wider range of computer architectures *with the same result* (run-time bytecode interpreters; performance is worse than C/C++)
- **Internet compatibility** — to allow the bytecode be accessed by a remote client incl. other programs (Java has a set of APIs that facilitate the connection using the standard Internet protocols)
- **Concurrency** — Java has the ability (in its language features, object model, JVM, and its API) to create applications which can interact with many systems at the same time by running separate *threads* of execution and controlling data access from those threads.

Further Information

Hortsmann's *Core Java for the Impatient*, Ch. 1.1

Context is Everything

It can only be the thought of verdure to come,
which prompts us in the autumn
to buy these dormant white lumps of vegetable matter
covered by a brown papery skin,
and lovingly to plant them and care for them.

It is a marvel to me
that under this cover they are labouring unseen
at such a rate within to give us
the sudden awesome beauty of spring flowering bulbs.

While winter reigns the earth reposes
but these colourless green ideas sleep furiously.

C. M. Street

http://www.linguistlist.org/issues/2/2-457.html#2