Voortgezet Programmeren

Lecture 0: Introduction

Tommi Tervonen

Econometric Institute, Erasmus University Rotterdam

Course learning objectives

After this course, you should be able to

- understand the main concepts in object-oriented programming
- design and implement programs in Java

This course is designed for ones who completed Programmeren (FEB22012) before this year

Why Java?

- Open (+-)
- Free
- Fast
- Portable
- Object-oriented
- Gets you a job

Course organization

- 6 lectures
  - Theoretical contents
  - Provide background for the exercises
- 6 exercise sessions
  - 6 large exercises done in pairs
  - Come to exercises to ask questions and get help with your code
- 6 exercise lectures (first one = this introduction)
  - Office hours for the teaching assistants (+me) to be around to give detailed answers to grading
  - Example answers to the exercises will be posted online, but there is always more than 1 correct answer

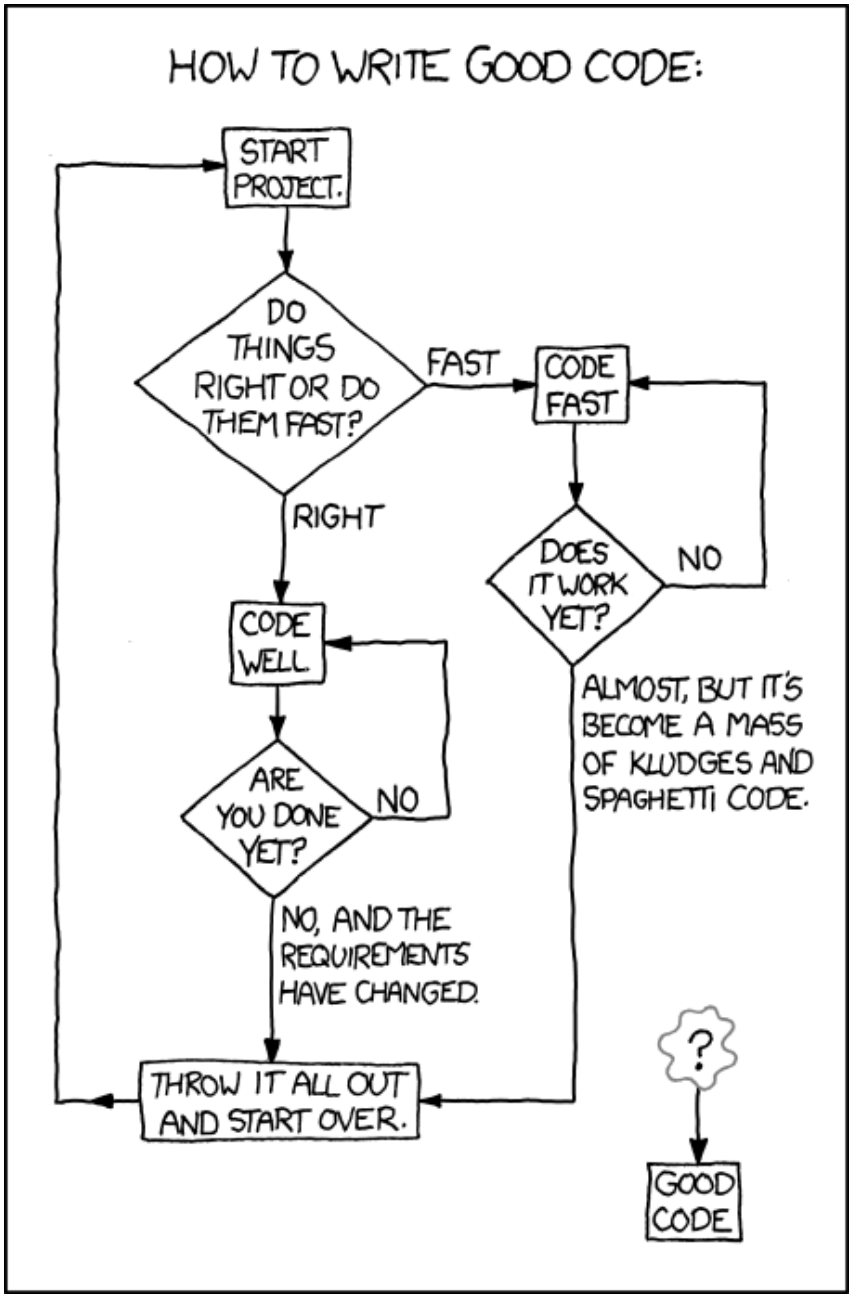
Study load

- 4 ECTS = 112h
- 6 lectures = 12h
- 6 exercise sessions = 12h
- 6 exercise lectures = 6h
- Exam = 4h
- $\Rightarrow$ Independent programming 78h = 13h/w

Grading

- Exercises: 50% (each $8.\bar{3}\%$)
  - Done in pairs
  - Exercises will be published in BB after Monday's lecture
  - Strict deadline on Sundays @ 23.59
  - Submission via BB: *only* the source files in a ZIP. Include a comment in all files with your names and student numbers
  - Incorrect submission format = 0 points
  - Non-compiling code = 0 points
  - Crashing code = 0 points
  - Not adhering to good programming practices = max 6 points
- Written exam: 50%
  - Essay questions

Making the exercises



- Don't underestimate the importance of theory

```
if(stuck()) {
   askHelp() || fail();
}
```

Plagiarism

- Do not submit anything you haven't written yourself
- Do not submit anything that is not your idea
- The teaching assistants will not give you answers in the tutorials: they will merely help you find the answer
- "But I could've solved this problem myself, it was just faster to google the solution"
- All suspected plagiarism will be reported to the examination board

Course staff

Tommi Tervonen	Lectures & exercises	H10-23	-
Alexander Hogenboom	Exercises	H10-21	ETC1+3
Frederik Hogenboom	Exercises	H10-21	ETC2+6
Charlie Ye	Exercises	H10-13	ETC4+5

- Also: you! Participate in course discussion forums in BB to get and provide help with the exercises
- TAs grade exercises and give feedback during "question time"

Required knowledge

Inleiding programmeren:

- Variables and methods
- Program flow
- Decisions and branching
- Control structures
- Bitwise operators
- Arithmetic operators
- Scoping

Lectures

L0 Introduction
- Practicalities
- Programming paradigms
- Compiled languages
- Introduction to types

L1 Elementary concepts in OOP
- Objects and Classes
- Variables and Methods
- Memory allocation and garbage collection

L2 Programming with Java
- Decisions, iteration
- Arrays
- Errors and Exceptions

L3 Programming by contract
- Data hiding
- Side effects
- Pre- and post-conditions
- Static variables and methods
- Unit testing

Lectures

L4 Interfaces and polymorphism

- Interfaces
- Casting
- Polymorphism
- Inner classes

L5 Inheritance

- Inheritance hierarchies
- Overriding
- Subclass construction
- Polymorphism and inheritance

L6 Java Collections Framework

- Collections
- Lists
- Sets
- Maps

Literature

- Lectures = main exam material
- Horstmann: Java Concepts (6th ed.), Wiley
- All course material is posted in `http://smaa.fi/tommi/courses/prog3/`
- If you don't know how computers work: LN-TT-22012-1 (`http://smaa.fi/tommi/courses/prog2/ln-tt-22012-1.pdf`)

Software

- JDK v6+
- Exercises must compile & run with Sun JDK with JRE 1.6.0_26-b03 (default in Ubuntu with sun-java6-jdk package)
- The exercise sessions will be guided with Eclipse (`eclipse.org`)

Q?

"The effective exploitation of his powers of abstraction must be regarded as one of the most vital activities of a competent programmer."

E.W. Dijkstra

Programming paradigms

- Programming paradigms refer to the philosophy behind designing programming languages
- When you know to program with 1 language of a paradigm, others of the same paradigm are easy to learn (mostly just syntax)

Programming paradigms

1. Procedural / imperative paradigm (C, Pascal, Matlab, R, Fortran, Algol, Python)
2. Object-oriented paradigm (Java, Smalltalk, C++ partially)
3. Declarative paradigm, including
   - Functional programming (ML, Lisp, Haskell, Erlang, Scala, Scheme)
   - Logic programming (Prolog)

OO vs Procedural

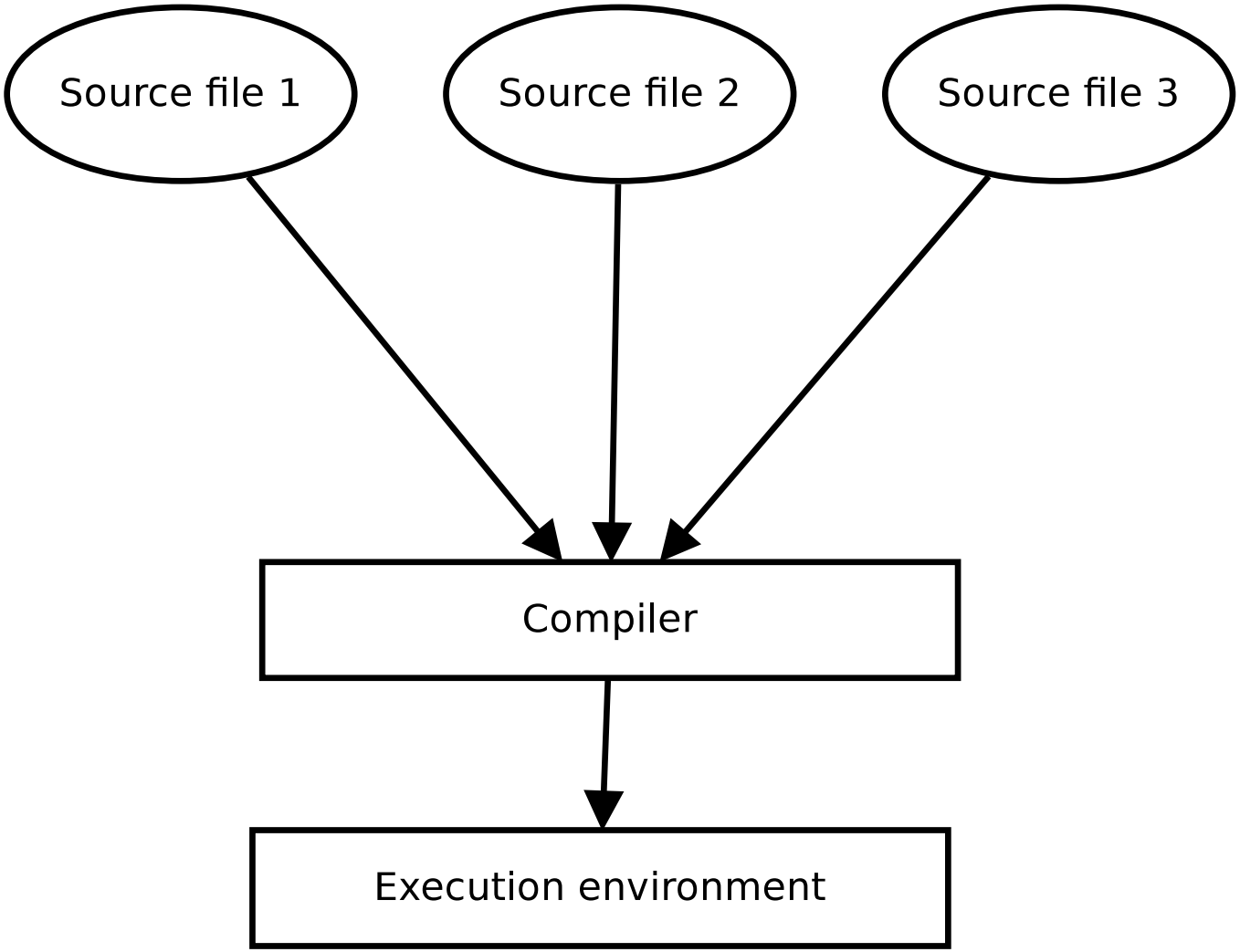
Object-oriented	Procedural
Design classes that communicate	Design global methods
Abstract Data Types	Data structures
Suitable for large programs	For "small" programs
Access control in language	Programmer has full access

- Both are part of imperative paradigm: control flow consists of *statements* that change the state of the program
- x = 2;
- Imperative paradigm makes program correctness hard to prove, as $x = 2 \neq x \leftarrow 2$

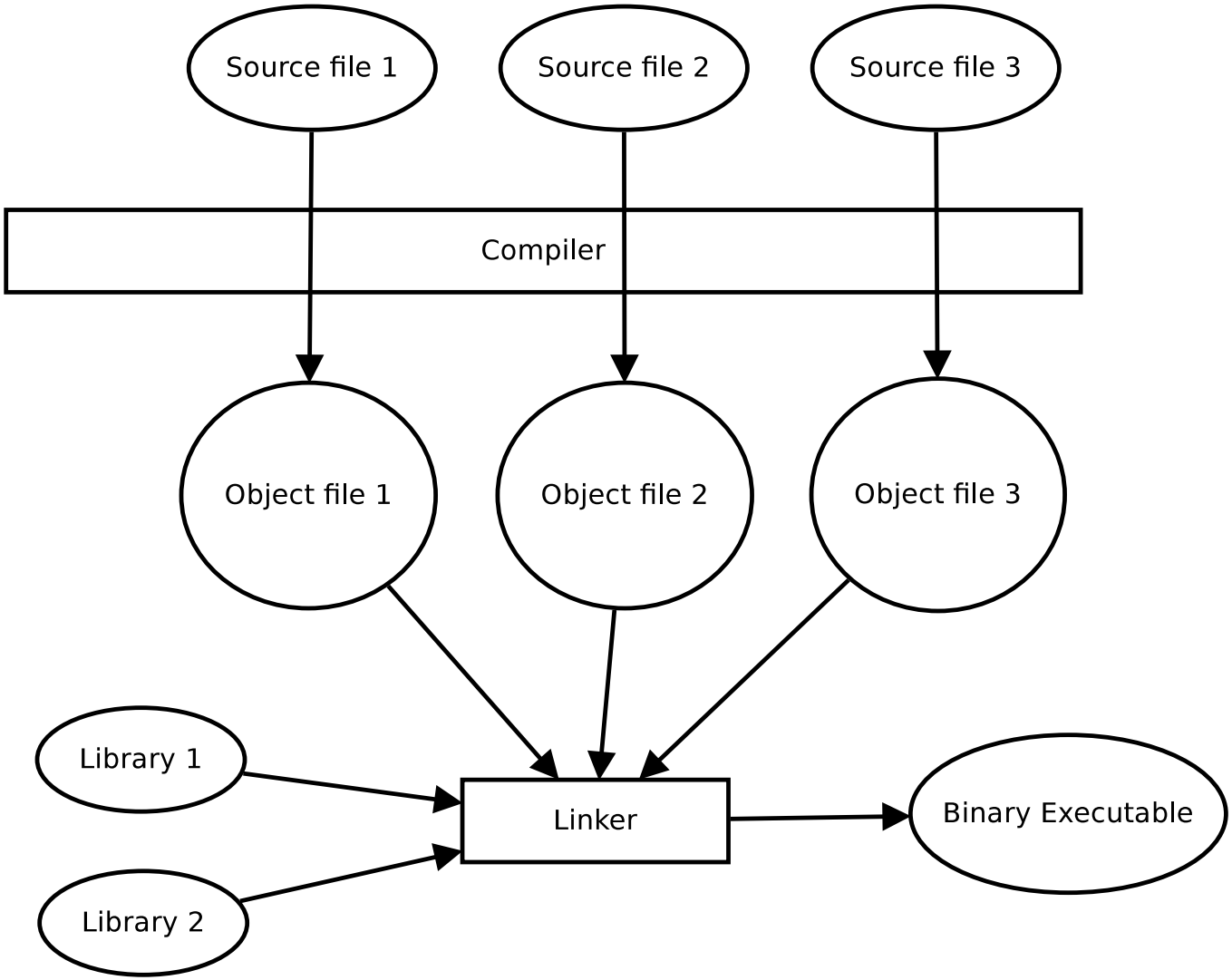
Compilation of languages

- Before source code can be executed, it needs to be *compiled* into an executable format
- The compilation can be made
  1. Completely in advance to a binary executable (fast)
  2. Partially in advance to bytecode to be executed in a virtual machine (Java, quite fast and portable)
  3. Run-time (slow but allows easy "modify & execute" cycles)

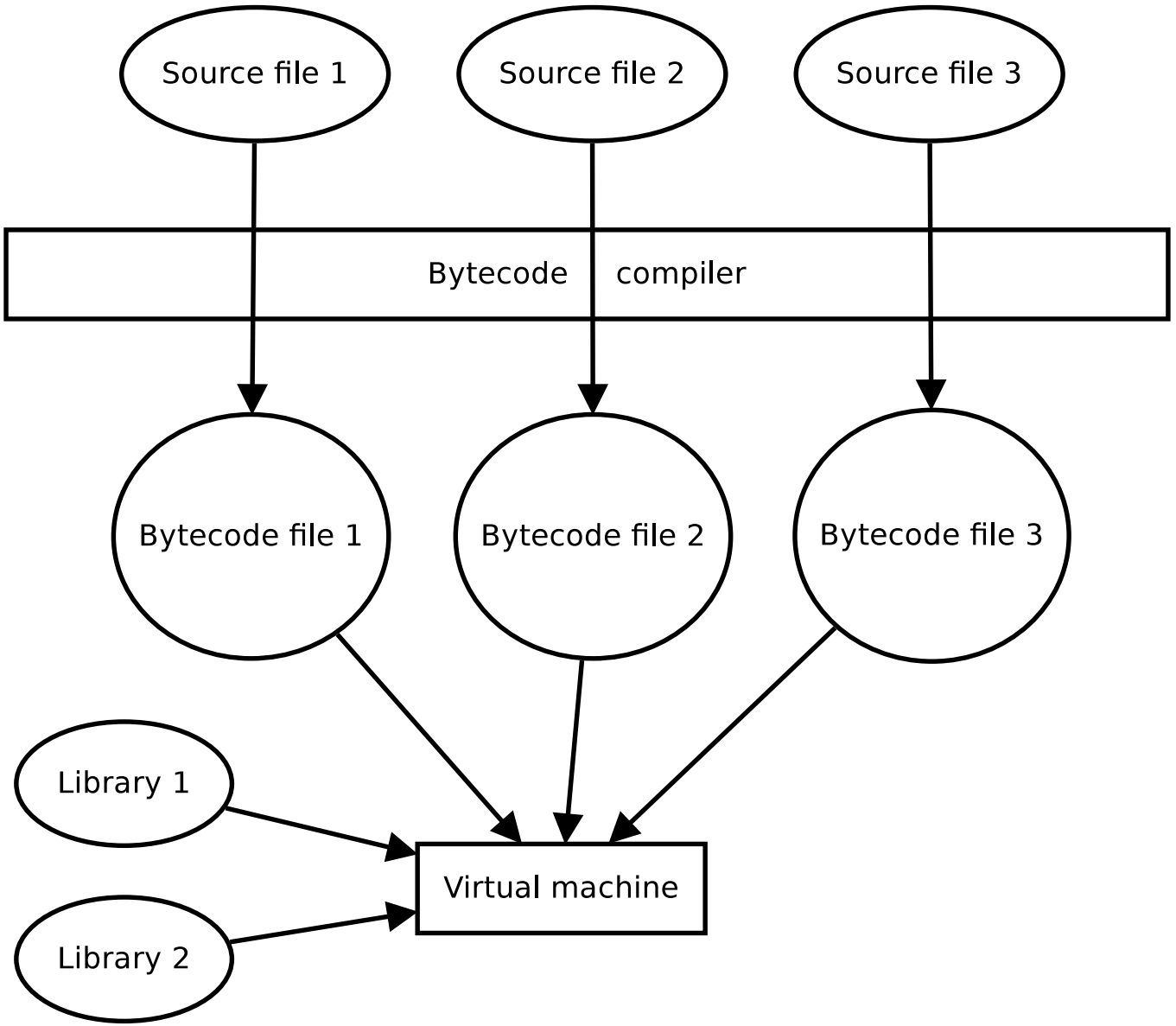
Runtime compiled languages (e.g. Matlab)



Fully compiled languages (e.g. C)



Bytecode compiled languages (e.g. Java)



Introduction to types

- Typing systems form the core of programming languages - they allow construction of abstractions
- Differences in electric currency $\rightarrow$ bits $\rightarrow$ numbers $\rightarrow$ characters $\rightarrow$ objects

Strong and weak typing

Weak typing : a single variable can be assigned varying types of values

```
y = 3; % ok - no type declaration required
y = 't'; % ok
```

Strong typing: each variable has a type associated with it

```
int x = 2; // ok
x = 3; // ok
x = ''s''; // compilation error
```

Typing in Matlab

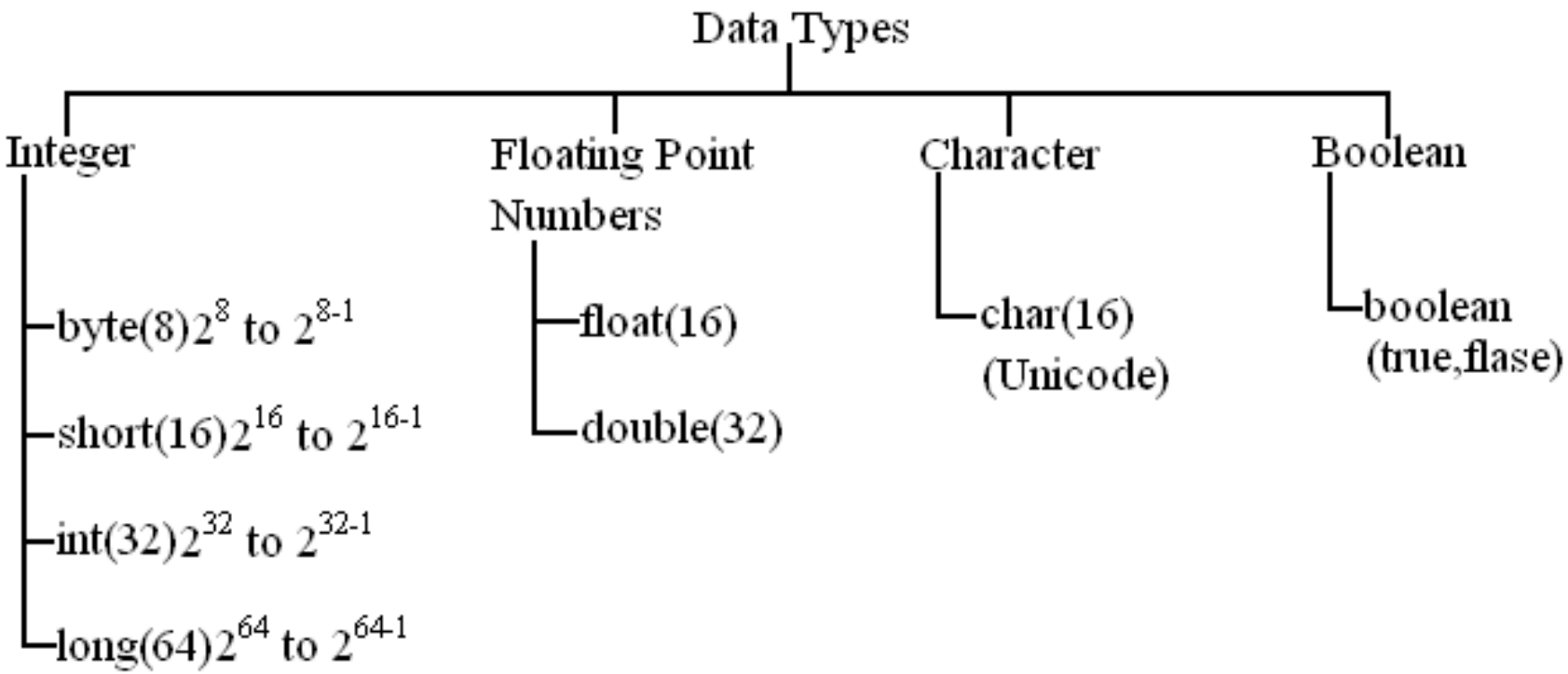
- Matlab is a weakly typed language, and the following are valid expressions:

```
x = 1;
y = '1';
z = x + y;
```

- Now z = ?

Typing in Java

- **Primitive** and **object** types



- Variables must be declared (int age;)
- Non-trivial conversions must be **type cast** (example)

Until next week

1. Read `http://docs.oracle.com/javase/tutorial/getStarted/intro/index.html`
2. Download and install JDK
3. Download, install, and familiarize yourself with Eclipse
4. Make sure you can compile and execute code in terminal (=command prompt in m$ terminology)
5. Make independently exercise 0 (not graded)