COMP8710 Advanced Java for Programmers

Lecture 1
Introduction & recap basic Java

Yang He

Topics

- Introduction
- Module overview
- Java versions
- Recap basic Java
  - Variable
  - Selection
  - Loop

About this module

- It is not an introduction to programming, or to Java
  - Try COMP8810/COMP8820 instead
- It does assume some experience with Java
- It starts with some basics about Java, but rapidly accelerates the pace
- It generally starts you on a subject, without giving you the full story to that subject. You are expected to do your own reading, via textbooks or online resources
- It aims to make you more reflective, object-oriented programmers

Our expectation

- You should have written substantial Java programs (at least 100 lines of code)
- You should have used loops, classes, instantiated objects, methods, collections, inheritance, recursion, etc.
- If not, it's better for you to drop this module and take COMP8810/COMP8820 instead
  - Please email the CEMS UG and PGT cemsugandpgt@kent.ac.uk

Teaching schedule

- Intensive teaching in weeks 1-6 (KV weeks 8-13)
- Each week we have
  - 3 lectures
  - 1 practical class (2-hr)
    - Task driven programming exercises
    - Supervisor provides assistance
- Detailed teaching schedule is available on Moodle

  https://moodle.kent.ac.uk/

Please check your timetable for details about your practical classes

Teaching resources on Moodle

- Teaching schedule
- Module announcements
- Discussion forum
- Lecture slides and recordings
- Class exercises
- Assignments
- Other useful resources

Please check the module Moodle page regularly!

Assessment

- 100% on coursework
- In total 2 assignments, 50% each

Reminder

- Attend all lectures and classes
  - Do ask if you are not sure or need help
  - If you are unable to attend, you should report your absences
- Submit your coursework before the deadline
  - If you need an extension with a valid reason, you should apply for an extension before the deadline

Detailed instructions on reporting absences and application for an extension are available on the CEMS Student Support Moodle page

Plagiarism

- All submitted work must be your own
- We run checks on all submitted work
- We take disciplinary action against anyone found to have committed plagiarism
- Some frequently asked questions concerning plagiarism and computing assignments can be found at

http://www.cs.kent.ac.uk/teaching/student/assessment/plagiarism.local

Textbooks (1)

- The link to “Reading List” is available on Moodle

Head first Java

Book - by Kathy Sierra; Bert Bates - 2008

- *It covers all the basic OO concepts as well as the Java language in an easy-to-follow approach.*
- *It does not assume a specific development environment.*

Objects first with Java: a practical introduction using BlueJ

Book - by David J. Barnes; Michael Kölling - 2017 - Sixth edition

- *This is used for COMP8810/COMP8820*
- *Full introduction to Java, also some higher aspects including OO design*

Textbooks (2)

- We aim to build programs that are robust, maintainable and clear to understand. A widely adopted approach is to use design patterns.
- A popular book on design patterns in Java:

Head First design patterns: building extensible and maintainable object-oriented software

Book - by Eric Freeman; Elisabeth Robson; Kathy Sierra; Bert Bates - 2020 -

- Another useful online resource is this short guide with Java examples:

  https://www.tutorialspoint.com/design_pattern/design_pattern_quick_guide.htm

Textbooks (3)

Java 8 in action: lambdas, streams, and functional-style programming

Book - by Raoul-Gabriel Urma; Mario Fusco; Alan Mycroft; Mario Fusco - 2015

- *A good book on new features added in Java 8*
  - *E.g. lambdas, streams, functional programming*
- *It assumes solid knowledge of OOP and Java*
- *Also available at: https://www.academia.edu/31676908/Java_8_in_Action_Manning*

- There are a wealth of Java resources available, including tutorials and API documentation
- These can be found at (and downloaded from) Oracle's site

  http://download.oracle.com/javase/tutorial/

Java versions

- Java was conceived in 1991 by James Gosling and colleagues at Sun Microsystems
- The first public announcement of Java 1.0a2 in 1995
- Java version 1.1 (1997), 1.2 (1998), …, 6 (2006), Java 7 (2011)
- Java 8 (2014) – major update of the Java language itself
  - e.g. lambda expressions, functional Interfaces, method references, …
  - A new collection API (streams)
- Six monthly release cycle: Java 9 (Sept 2017), Java 10 (Mar 2018), ..., Java 22 (Mar 2024)
- We use the latest LTS (Long-Term-Support): Java 21 (2023)
  - You can download OpenJDK JDK 21 from https://jdk.java.net/archive/

Recap – Variables

Variables (1)

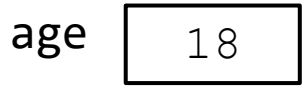
- Variables are the named locations in a computer memory
- In Java, when declaring a variable, we need to specify type and name of the variable
- Variables may be initialised on declaration
- e.g.

```
int index;

String name = "Tom";

Student tom = new Student(name);
```

Variables (2)

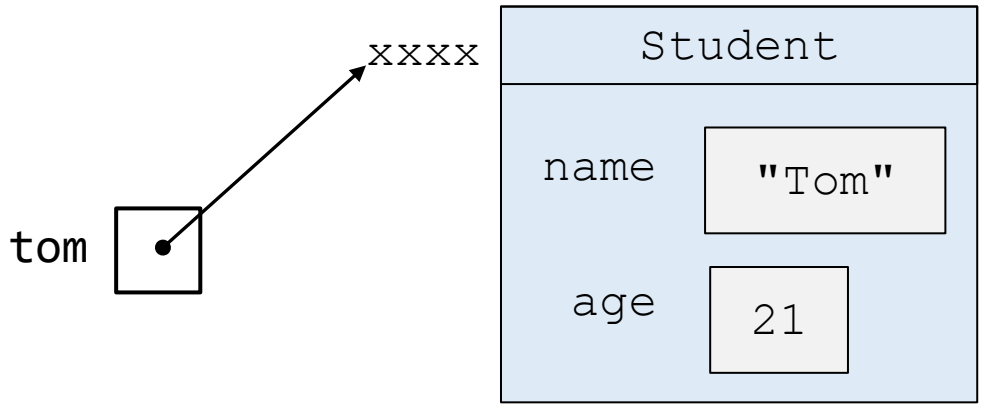
- The way how a variable holds its value depends on the type of the variable
- There are two types of variables
  - Primitive data type
    - e.g.

```
int age = 18;
```



  - Objects
    - e.g.

```
Student tom = new Student("Tom", 21);
```



Variables (3)

- In Java, variables include
  - Class fields, or instance variables

```
private int age;
```

  - Parameters

```
void setAge(int newAge);
```

  - Local variables

```
HashMap<String, String> contacts = new HashMap<>();
```

Variable’s scope and lifetime

- The scope of a variable is the part of the program in which it can be used
- The lifetime of a variable refers to the time when a variable can be used

Local variable type inference (JDK 10)

- We can use var to declare local variables with initial values that are *not* null
- The types of the variables are inferred *automatically* by the compiler
- E.g.

```
var age = 0;
var message = "Hello world!";
var found = false;
var contacts = new HashMap<String, String>();
```

Recap – Selection

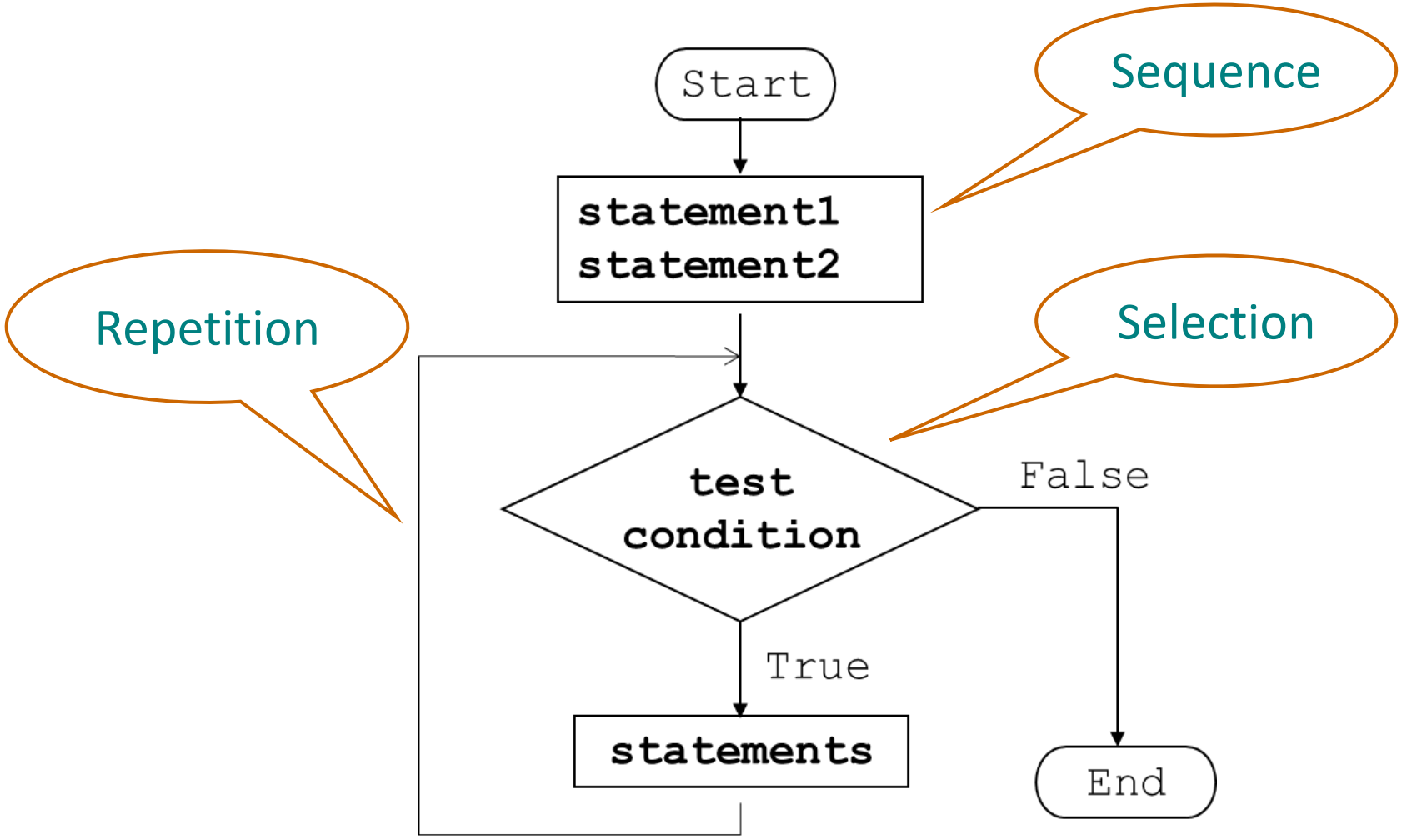
Program control structures

Sequence

Selection

Repetition

Flow chart


Start
statement1
statement2
test
condition
False
True
statements
End
Sequence
Selection
Repetition

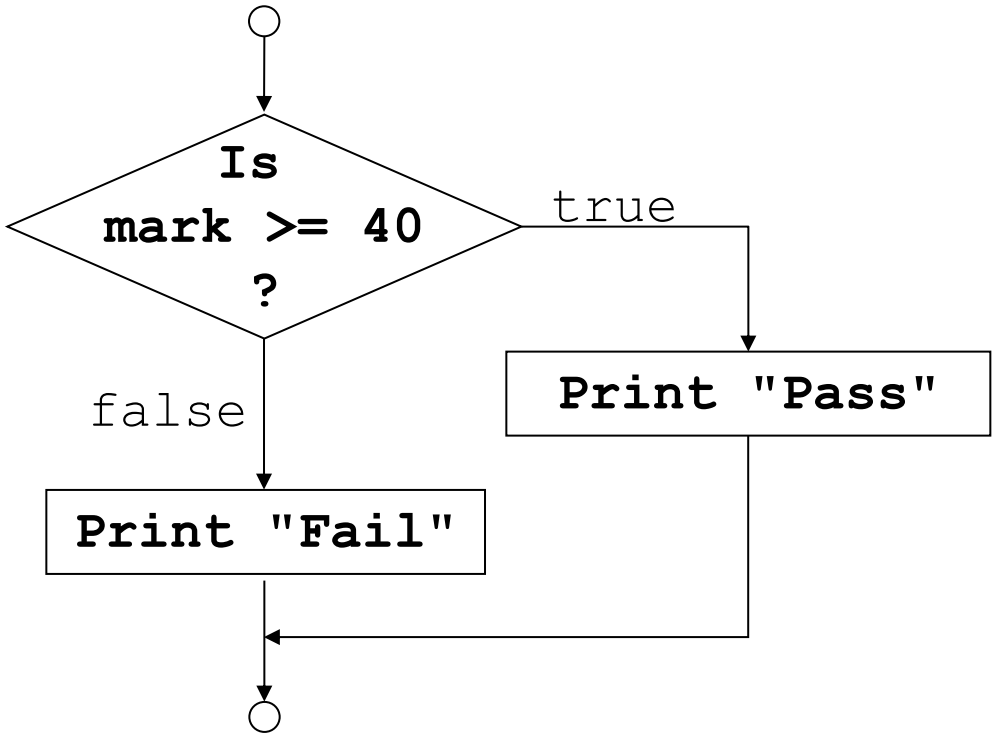
Selection structures

- if statement
  - if … else statement
  - Nested if statement
- Conditional operator
- The switch statement

If…else statement, or conditional operator

- e.g.

```
if (mark >= 40) {
    System.out.println("Pass");
}
else {
    System.out.println("Fail");
}
```



Or simply using conditional operator:

```
System.out.println(mark >=40 ? "Pass" : "Fail");
```

The switch statement

- E.g.

```
public String getWeekday(int day) {

    switch (day) {
        case 1:
            return "Monday";
        case 2:
            return "Tuesday";
        case 3:
            return "Wednesday";
        case 4:
            return "Thursday";
        case 5:
            return "Friday";
        default:
            return "Error: invalid weekday";
    }
}
```

Recap – Loops

Repetition structures

- for each loop
- while loop
- for loop

for each loop

- E.g. A list of Dogs

```
ArrayList<Dog> dogs;
```

- E.g. Print out the names of all dogs owned by Wilson:

```
for (Dog d: dogs) {
   if (d.getOwner().equals("Wilson") ) {
      System.out.println( d.getName() );
   }
}
```

Dog class provides accessor methods for a dog's name, age and owner.

Define a method

- It's better to define a method that takes an owner's name and prints the names of all dogs owned by the given owner.

```
public void printDogsOwnedBy(String owner)
{
   for (Dog d: dogs) {
      if ( d.getOwner().equals(owner) ) {
         System.out.println( d.getName() );
      }
   }
}
```

Homework:
Improve the method to print an appropriate message
if no dog is found for a given owner.

Method with variable arguments (varargs)

- We can define a method that takes an arbitrary number of parameters using varargs (…)
- E.g.

```
void printFullName(String... names) {

    var fullname = "";
    for (String name : names) {
        fullname += name + " ";
    }
    System.out.println(fullname);
}


printFullName("Sam", "Miller");
printFullName("Sam", "Peter", "Miller");
```

while loop (1)

- The syntax:

```
while ( condition )  {
    statements in loop body
}
```

- The loop body is executed repeated as long as the condition is met
- The test of the condition is always performed at the beginning of the repetition
- Consequently, the body of the loop may not be executed at all!

while loop (2)

- E.g. Write Java code to calculate the sum of the squares that are less than 200 of all the odd numbers,

  i.e. $\text{sum} = 1^2 + 3^2 + 5^2 + \ldots$

```
int sum = 0;
int k = 1;
int square = 1;y

while (square < 200) {
   sum = sum + square;
   k = k + 2;
   square = k * k;
}

System.out.println(sum);
```

Java development environment

- We shall use IntelliJ IDEA
  - It is a cross-platform IDE (Integrated Development Environment) that provides consistent experience on Windows, macOS, & Linux
  - It is designed to maximize developer productivity
  - It offers free educational licenses for all university students

- **Homework**: Complete Class 0 (available on Moodle)
  a) Install IntelliJ IDEA on your own computer
  b) Explore IntelliJ IDEA