COMPGC01 Introductory Programming

Induction Week - Lecture 1

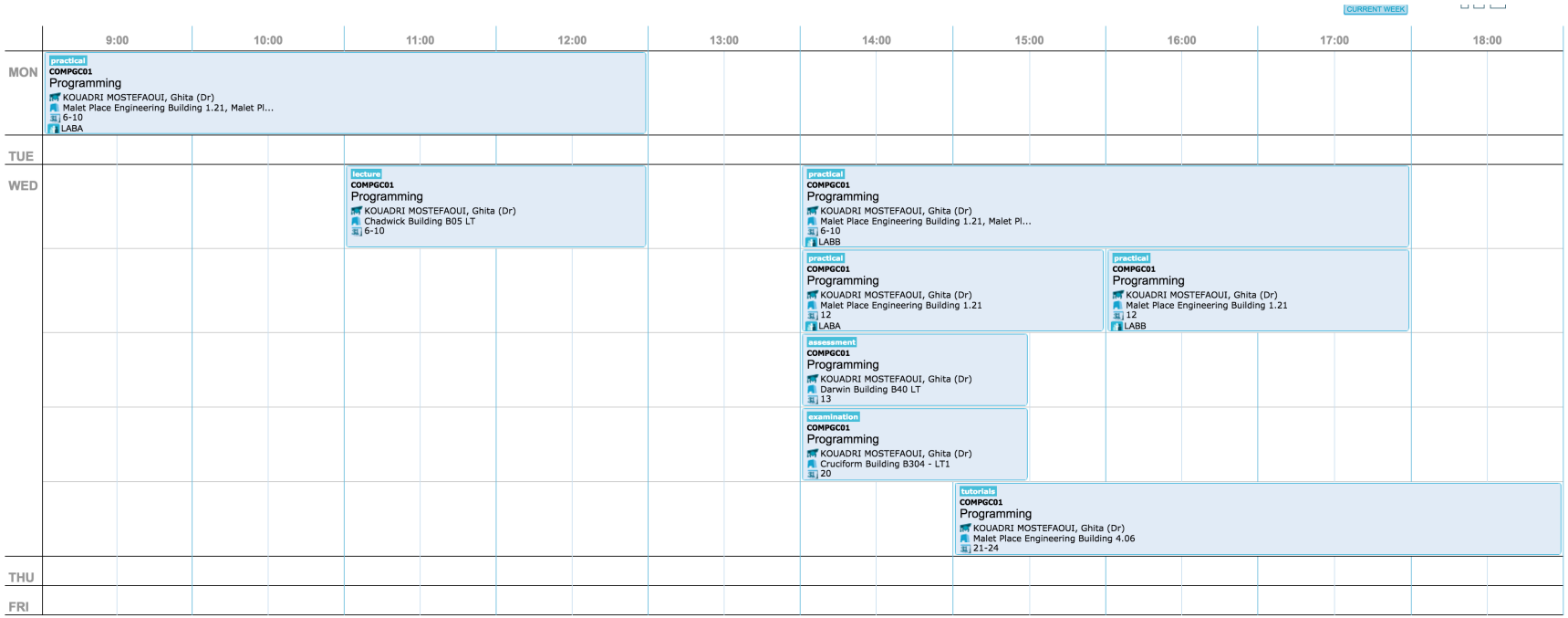
Dr Ghita Kouadri Mostefaoui

Department of Computer Science

Welcome to UCL

- Today's Content
  - Module timetable
  - Goals of GC01
  - Syllabus
  - Module format
  - Module assessment
  - Hello World
  - Tools

Understanding the timetable



Academic calendar
http://www.ucl.ac.uk/estates/roombooking/calendar/1617.pdf

moodle.ucl.ac.uk

- Used for course announcements
- Lecture slides, tutorials and exercises will be uploaded weekly to Moodle.
- All information regarding assessment
- All information regarding coursework in particular
- Module discussion forum

GC01 Goals

- ✓ Primary Aim = learn and **be able to** program.
- ✓ Second Aim = learn and **be able to** program in Java.
- ✓ Tertiary Aim = apply concepts to GC02 App Design course and GC05 Algorithmics

- This course introduces **imperative** and **object-oriented** programming using **Java**
- It aims to develop key programming and problem solving skills
- **Experience and confidence building**
  - provides a large amount of programming practice
  - Labs with help
- Connection with GC02 – Design with Apps
  - You need to **learn to program in the languages** and **developer environment (IDE)** for your apps design (GC02) class
  - You can re-apply the principles you learnt in GC01.
- Connection with GC05 – Algorithmics
  - Content of GC01 adapted in order to provide a basis for GC02 and to remove redundancy.

Syllabus

- Introduction to Java and imperative programming
- Variables.
- Javadoc and Commenting.
- Operators and expressions.
- Conditions and loops.
- Strings, ArrayList
- Files and file handling
- Methods and using method parameters.
- Scope and lifetime.
- Java class libraries
- Creating and using objects
- Declaring classes.
  Instance variables and methods.
- Inheritance.
- Drawing simple pictures.
- GUIs (Graphical User Interfaces).
- Data Structures
- What's new in Java 8?

Course format (2016)

- INTENSIVE COURSE. Runs for 1st half of Term 1 ONLY (1 induction week + 5 weeks).
- You must be comfortable at programming by week 3.
- 2 hours of lecture each week on Wednesday 11am-1pm
- Labs:
  - Tutorials + exercises.
  - Tutorials… means that you will be instructed to learn new things not 'explicitly' covered in lectures.
  - 4 hours of available lab time (3 hours mandatory, 1 hour for extra help and recovery).
  - *Must complete all exercises.*

The program …

- Induction week labs –**work individually.**
  - Check timetable: http://www.cs.ucl.ac.uk/students/student_information/timetables/induction_week/msccsfc/

- Weeks 1 – 5 – **Paired exercises** in labs allocated. Binary marked by Teaching Assistants for progression. Auto-enrolled onto 1hr recovery sessions if not keeping up. **MUST** share exercise workload in pairs.

- Two in-class tests, one just after reading week, one just after Christmas break.

- Coursework – Two options: **individual work** or **group work** - due after reading week.

Team Exercises during Labs
(Weeks 1-5)

- Work as a PAIR for exercises.
- Do NOT let one person do all the work – Teaching Assistants have been instructed to issue fails if there is not enough joint effort.
- Share exercises as joint tasks. Plan solutions and manage how you will collaborate together.
- Use dropbox, google drive, GitHub, or any similar tool to backup your work. *If you lose your code, no excuse will be accepted!*
- "Commenting" in code should explain who did what. Commenting java code will be discussed in a coming lecture.

The role of the teaching assistants

- All questions regarding exercises and lab work should be directed to the *Teaching Assistants*.
- Lab sessions: A and B. Check your timetable.
- Please respect the timetable set!

Your progression

- Get ticked off the register whenever you complete a set of exercises, it goes onto your moodle records.
- If the lab is filled, feel free to use the lift lobby working areas as well if you have your own laptops. The teaching assistants will be in your Lab when you want to ask them questions.
- When you have finished you may work on the coursework (released in Week 1) or boost your experience by finding more exercises on the net on the same topic. You must practice **more** than what is given.
- Teaching Assistants will be present to give you help and advice while you program.
- Feel free to ask for more exercises on certain topics.

The UCL lab ticketing system

- This year we are introducing a new system for students/TAs management during lab sessions
- The system is available here: http://46.101.23.91/
- You should sign-up before being able to use it
- It‘s an Alpha version so please feel to email me with any bug, or any feature you would like to see added to it.

The UCL lab ticketing system

- The use of the ticketing system to ask for support is mandatory! Unless you can't use it for technical reasons for example (system down, etc)
- You can access it from your laptop, mobile phone, etc
- Demo: : http://46.101.23.91/

How much time should I spend on coding?

- You need to schedule a lot more time, >15 hours a week, once the coursework starts.
- Keep a diary of your plan and progress.
- Learn from outside resources – try other examples.
- You need to cover enough of each topic **several times**, to be comfortable with it.
- Your endurance and fluency for coding will grow with time.

Getting started

- The best way to learn how to program is to write programs!
- So, a large part of the course is based on lab classes and their exercises.
- Course will support and expand but also explore a wider range of subjects.
- There will be puzzles in class to solve. Be prepared with a *notepad*.

Assumptions

- *No previous programming experience is assumed -we start from the beginning.*
- Be proactive in your research, find literature, examples, tutorials. Share good resources on moodle.
- Be prepared to defend and talk about your coding to others.
- You'll need your own notes for exam revision!

Assessment

- The course is assessed by both in class assessment and coursework.
  - All tutorials on Moodle must be completed (binary marked)
  - Tests (Multiple choice) x 2 (60%)
    - First MCQ (30%)
    - Second MCQ (30%)
  - Programming exercises (40%)
    - Lab exercise (10%) – *induction week not included in final mark*
    - Coursework (30%)

50% required at MSc level to pass this course.

What to do if …

- You can't attend an exam or
- Can't submit your coursework by the set deadline
- https://www.ucl.ac.uk/srs/academic-manual/c4/extenuating-circumstances/principles
- Tutors don't deal with extenuating circumstances!
- Check your timetable for the exam dates and moodle for the coursework deadline.

Plagiarism

- Plagiarism = Copying someone else's work.
- Copying = Cheating.
- DON'T DO IT. IT WILL **NOT BE TOLERATED**.
- It is dishonest, shameful and risky.
- http://www.ucl.ac.uk/current-students/guidelines/plagiarism

Plagiarism Case

- Students asked to “share their answers” to make friends.
- Do NOT do this!!!
- Attempting to copy parts of others works and rename, rephrase, alter… you WILL BOTH get caught.

- Use of well published libraries and functions are WELCOMED and ENCOURAGED. You must learn to cite and reference academically. However, citing your friends works to expect the marks – NOT ACCEPTED.

How to pass

- Keep working methodically and keep up pace.
- WRITE CODE.
- WRITE MORE CODE.
- WRITING A LOT MORE CODE.
- Find alternative solutions to your programming.
- Read books on Java.
- Use web resources.
- Use multiple sources (lectures, labs, book, web tutorials).
- Learn to pose your own questions to yourself to solve.
- Remember: this particular course is central to your **entire MSc!**

Don’t Panic!

- We *want you to pass*.
- There are plenty of people to ask for help if the going gets tough - your teaching assistants, the lecturer, your personal tutor, the departmental tutor and others.

You CAN do it.

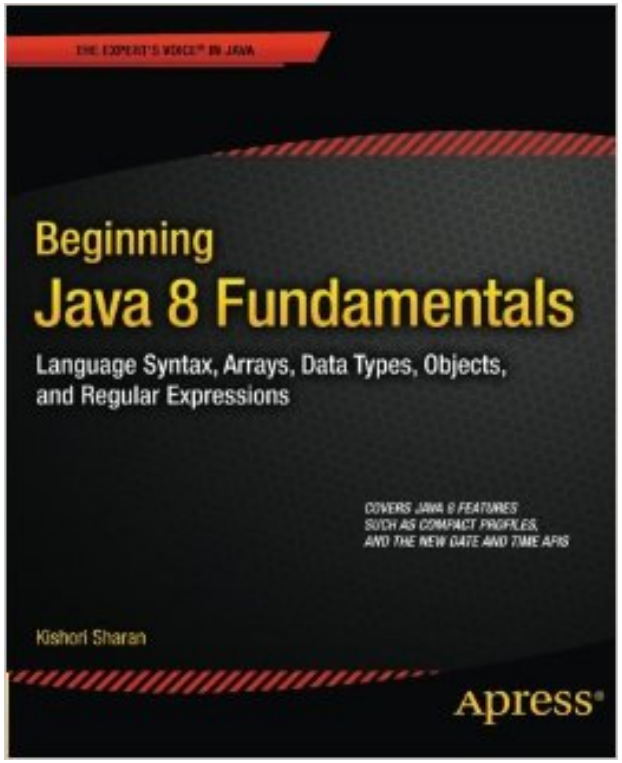
Course Textbook

Beginning Java 8 Fundamentals

by Kishori Sharan

Apress; 2014

Available online (UCL library)



Web Resources

- https://www.ucl.ac.uk/lynda
  - Excellent video lectures (e.g. Java Essential Training with David Gassner)
- http://www.tutorialspoint.com/java/
  - Excellent and free
- http://docs.oracle.com/javase/tutorial/java/
  - Excellent online tutorials, highly recommended
- Search java + “a task” to look for resources for a specific task, library or package
  - e.g. JAVA file reading
  - Forums are very useful

On a piece of paper…

- Write out the following please.

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

Congratulations!

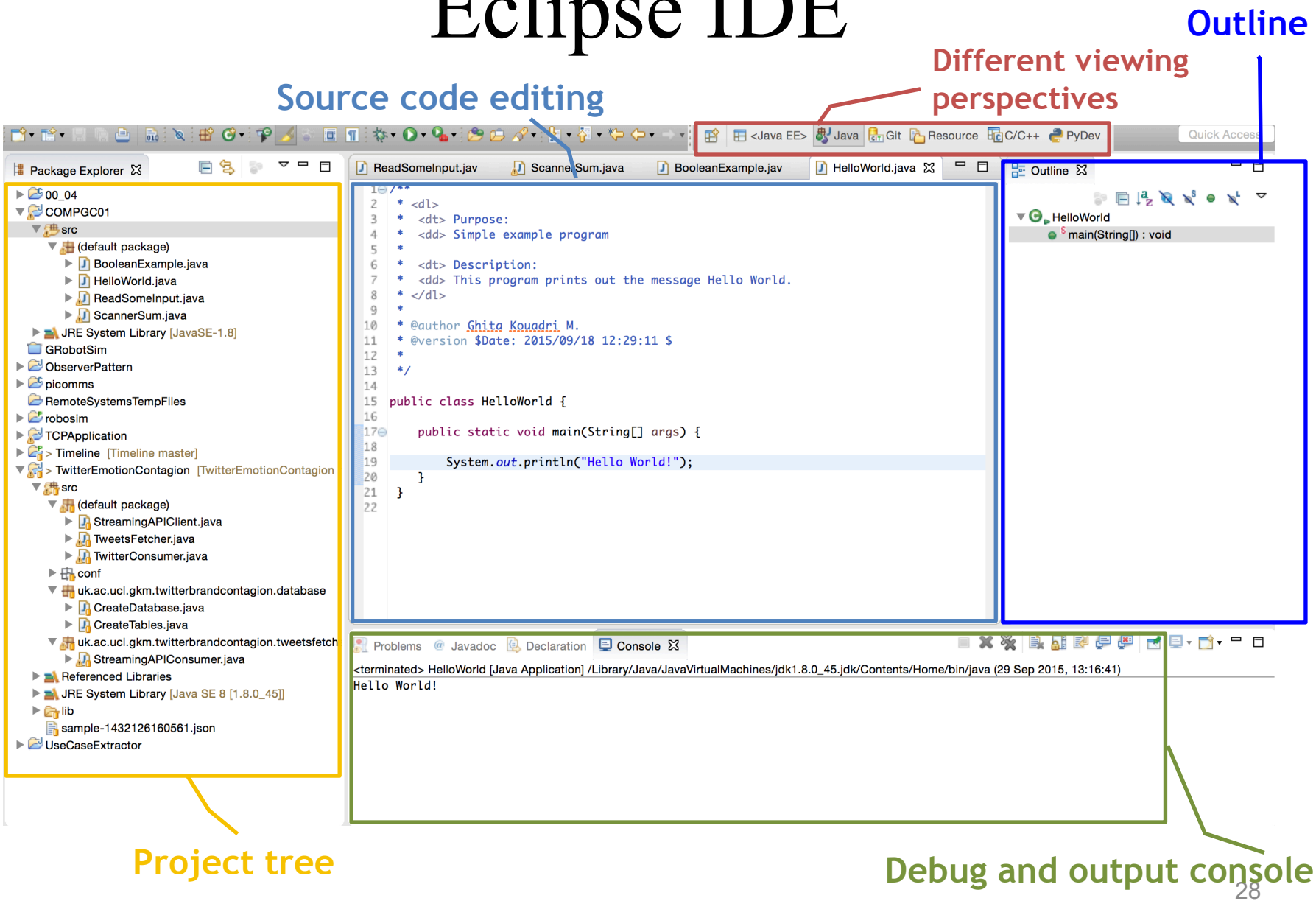
- You've started your journey with your first program ☺
- This course is Java focused, but in time and with the right rationale, you will learn to apply it to a multitude of programming languages.
- Please make it your first "hello world" program in the lab tomorrow.

Tools for GC01

- Java – is the programming language. By itself, you can program in plain text files and compile and run it. Java runs on a lot of machines.

- Eclipse – is an integrated development environment. Instead of writing in notepad, you can program inside a tool that does a lot more useful things; like check each line as you write, help you to find and fix errors and helps with key words. Eclipse is free.

Eclipse IDE


Outline
Different viewing perspectives
Source code editing
<Java EE>
Java
Git
Resource
C/C++
PyDev
Quick Access
Package Explorer
00_04
COMPGC01
src
(default package)
BooleanExample.java
HelloWorld.java
ReadSomeInput.java
ScannerSum.java
JRE System Library [JavaSE-1.8]
GRobotSim
ObserverPattern
picomms
RemoteSystemsTempFiles
robosim
TCPApplication
> Timeline [Timeline master]
> TwitterEmotionContagion [TwitterEmotionContagion
src
(default package)
StreamingAPIClient.java
TweetsFetcher.java
TwitterConsumer.java
conf
uk.ac.ucl.gkm.twitterbrandcontagion.database
CreateDatabase.java
CreateTables.java
uk.ac.ucl.gkm.twitterbrandcontagion.tweetsfetch
StreamingAPIConsumer.java
Referenced Libraries
JRE System Library [Java SE 8 [1.8.0_45]]
lib
sample-1432126160561.json
UseCaseExtractor
ReadSomeInput.jav
ScannerSum.java
BooleanExample.jav
HelloWorld.java
/**
* <dl>
* <dt> Purpose:
* <dd> Simple example program
*
* <dt> Description:
* <dd> This program prints out the message Hello World.
* </dl>
*
* @author Ghita Kouadri M.
* @version $Date: 2015/09/18 12:29:11 $
*
*/
public class HelloWorld {
public static void main(String[] args) {
System.out.println("Hello World!");
}
}
Outline
HelloWorld
main(String[]) : void
Problems
Javadoc
Declaration
Console
<terminated> HelloWorld [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_45.jdk/Contents/Home/bin/java (29 Sep 2015, 13:16:41)
Hello World!
Project tree
Debug and output console

Double check

- All your UCL accounts are working (CS and UCL etc).
- You can log in to moodle, you have registered for all the right courses and are enrolled.
- You signed-up and can log in to the Lab ticketing system before the first lab session.
- For the labs you need to use your own laptops.

A good start

- Familiarise yourself with the messaging system on Moodle in case you need to ask the Teaching Assistants anything.

- Keep your courseworks and exercises separately in your homespace and a good folder naming strategy to keep your courses neat and tidy, e.g.
  - GC01username/labs/week1/exercise1
  - GC01username/notes/
  - GC01username/coursework/
  - GC01username/testingfiles/

Laptops?

- The best option for your own workflow – use your own laptops, work anywhere. *Keep a Backup copy elsewhere, don't trust your laptop!*

- Require a minimum of 2GB minimum ram for the courseworks to run (4GB or higher is preferred).

- Linux, Windows or MacOs are all ok!

Tomorrow

- First lecture on Java Programming.
- Lab tasks by Teaching Assistants.
- *Please bring your laptops!*

Thank you and good luck!