Kingston University London
Faculty of Science, Engineering and Computing

Full Module Guide 2015/16

CI4100 Programming 1

Staff	Name	Room	Extension	Contact email and consultation hours
Module leader	Paul Neve (PN)	SB3034	67041	paul@kingston.ac.uk Office hours may differ from week to week due to variations in timetabled classes. Consult Studyspace for the most up to date details.

Teaching schedule information	You will be able to access your timetable for the 2015/16 academic year via the University mobile app or via OSIS (the Online Student Information System). More information on all aspects of timetabling can be found on the MyTimetable pages on MyKingston.

Each student is expected to attend at least

1 x 2 hour lecture weekly
1 x 2 hour workshop weekly

- Times and locations of lectures workshops vary throughout the year by cohort and group.
- Please consult OSIS for times and locations for your regular lectures and workshop sessions.
- Note that it may take some time for the published online timetable on OSIS or the MyTimetable pages to catch up with any changes. Thus you should ALWAYS check Studyspace the start of each week. Changes will also be emailed to your university email account. If Studyspace and/or an email sent to you contradicts the timetabling app, Studyspace and/or email should be considered the authoritative source.

In-course assessment	Type	%	Due dates	Feedback
These dates are indicative. Consult Studyspace for up-to-date information on assessment.	Weekly workshop activities	60%	Unit 1: Fri 6th November Unit 2: 11th January Unit 3: 22nd February Unit 4: 23rd April	Immediate via NoobLab
	In-class “spot” tests plus one-shot questions during lectures	40%	Spot Test 1: Week of 26th October Spot Test 2: Week of 7th December Spot Test 3: Week of 8th February Spot Test 4: Week of 4th April	Immediate, following the test

MODULE SUMMARY

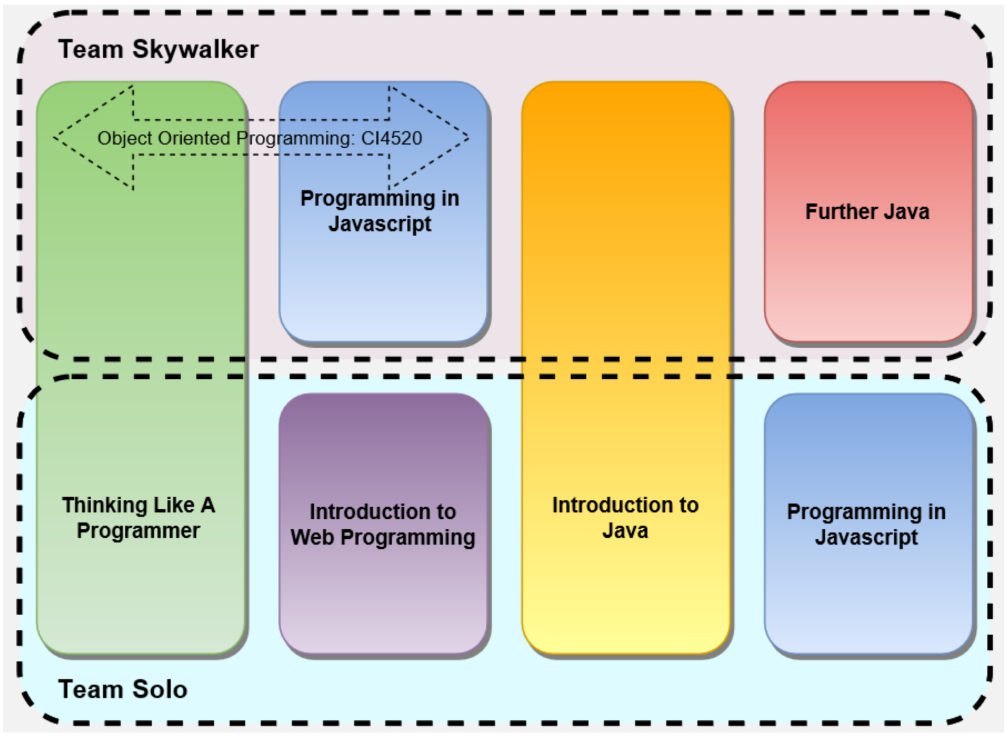
Welcome to the module!

The aim of the module is to provide a foundation for all programming activities that follow in subsequent years of your course. We do not assume that you have previous experience of programming; we start from the very beginning. We try to develop your ability to break problems down and "think like a programmer" before we break your brain with complex programming languages and more advanced concepts.

There is a misconception that programming is "hard" or "boring". My job is to persuade you that it's not. We try to make the material as engaging and fun as possible and we make use of our own homegrown NoobLab environment to do so.

Students doing Object Oriented Programming will do the units Thinking Like A Programmer and Programming in JavaScript with Paul before Christmas. They will then go off to do C++ with Ahmed.

Students doing Programming 1 will do four out of five possible units. At the end of Thinking Like A Programmer we will take a look at your progress. You will then be allocated into either Team Skywalker or Team Solo. This will then determine which units you will do as follows:


Team Skywalker
Object Oriented Programming: CI4520
Programming in Javascript
Further Java
Thinking Like A Programmer
Introduction to Web Programming
Introduction to Java
Programming in Javascript
Team Solo

During Enrichment Activity Week (week 6 of term) I will look at your progress. Those of you who have demonstrated an aptitude for the material in Thinking Like A Programmer will join Team Skywalker and will focus more on Java with on Object Oriented principles in the latter part of the module. Those of you who need a little more support will not do as much Java, and you'll look at Web application programming instead as a member of Team Solo. Both routes through the module carry the same assessment weight and will set you up for Programming 2 in the second year. Neither route is "better" and it is possible to give 100% for the module regardless of which route you end up following. What we want to do is to give everyone the best possible chance of success, and the best possible chance to maximise their potential.

LECTURE PROGRAMME

Each unit consists of five lectures with associated workshops. This is an ***indicative*** schedule and may be subject to change.

Thinking Like a Programmer (Both groups)

Week of	**Subject**
Sep 28	**Introduction to the Module**
Oct 5	**Fundamental programming constructs**
Oct 12	**Booleans and more on functions**
Oct 19	**Moving to "real" code**
Oct 26	**In Class Test**

Programming in Javascript (both groups – Team Skywalker do this as their second unit, Team Solo do this as their last unit)

Week of (Skywalker)	Week of (Solo)	**Subject**
Nov 9	Feb 22	**Introduction to Javascript**
Nov 16	Mar 29	**Functions and variable**
Nov 23	Mar 7	**HTML and the Document Object Model (or "everything you've learned is a lie")**
Nov 30	Mar 14	**Events on the DOM and creating interactivity**
Dec 7	Apr 4	**In Class Test**

The Basics of Web Programming (Team Solo)

Week of	**Subject**
Nov 9	**The basics of HTML**
Nov 16	**Links, images, forms and multi-page sites**
Nov 23	**Introduction to PHP**
Nov 30	**Reading form data with PHP**
Dec 7	**In Class Test**

Introduction to Java (both groups)

Week No	**Subject**
Jan 11	**The basics of the Java language**
Jan 18	**Conditional and loop constructs / arrays**
Jan 25	**Introduction to object orientation (or "everything you've learned is a lie")**
Feb 1	**"Madness in the Methods": parameters, return values and constructors**
Feb 8	**In Class Test**

Object Oriented Programming in Java (Team Skywalker only)

Week No	**Subject**
Feb 22	**Encapsulation and packaging**
Mar 29	**Arrays of Objects / Inheritance**
Mar 7	**Collections**
Mar 14	**Java programming in the real world: IDEs and bringing it all together**
Apr 4	**In Class Test**

WORKSHOP/SEMINAR/TUTORIAL PROGRAMME

The 2 hour lecture each week will be supplemented by a 2 hour hands-on workshop session. Please see Studyspace and OSIS for details of the location and times.

During enrichment weeks, we will endeavour to provide additional support available in the form of optional “codebash” sessions. Look for announcements on Studyspace.

6 Assessment

There are several ways you will be assessed on this module. Your practical work makes up 60% of the module. Each and every NoobLab medal you win contributes towards your final module mark. We break this down by unit as follows:

Team Skywalker	**Team Solo**
• Practical work for *Thinking Like A Programmer*: **15%** • Practical work for *Programming in Javascript*: **15%** • Practical work for *Introduction to Java*: **15%** • Practical work for *Further Java*: **15%**	• Practical work for *Thinking Like A Programmer*: **15%** • Practical work for *Introduction to Web Programming*: **15%** • Practical work for *Introduction to Java*: **15%** • Practical work for *Programming in Javascript*: **15%**
Total for practical work: 60%	

The remaining 40% of your marks will come from how well you do when you are asked questions in class. You will be asked questions in two settings. At the end of each unit we will run an in-class test. This will be under exam conditions and is designed to test your knowledge of everything that’s been covered during the unit. Most of the questions that count towards this 40% will be given to you during these tests. However, we will also be asking questions during normal lectures that you will answer with your clickers! **These will count towards your final mark!** So, you need to be paying attention during lectures – and you need to make sure you’re at every lecture, too! When you miss lectures you will miss marks.

Total for in-class tests and questions answered during lectures: 40%

Important note
It is also crucial that you bring your clicker to each and every lecture and workshop. If you do not do so then not only do you risk missing out on marks, but it will also affect your attendance record. If you don't have your clicker and don't answer the questions during a lecture, we have no way of knowing whether you were there!

You should follow all instructions given on Studyspace and on NoobLab. It is important that if you have any queries that you ask one of the module team for clarification.

You are reminded of the faculty policy for the late submission of coursework. Any work submitted up to a week late will be capped at 40%, anything submitted later than this will receive a zero mark.

If you are ill or have problems affecting your studies, the **University Mitigating Circumstances policy** may apply. You will need to complete a form and attach suitable independent documentation. Remember if you submit a piece of work or attend an examination, you have judged yourself fit to undertake the assessment and cannot claim mitigating circumstances retrospectively.

Further guidance on mitigating circumstancesis available on the My Kingston site:

My Kingston > My Faculty > Science, Engineering and Computing > SEC Student Support

https://mykingston.kingston.ac.uk/myfaculty/sec/secstudentsupport/Pages/default.aspx

Vivas

The module team reserves the right to ask you to attend a viva about submitted work. At such a viva you will be asked to explain how the code in the submitted work operates and the processes you used to create it. If you are not able to satisfactorily explain your own work then we reserve the right to deduct some or all of the marks awarded for it.

Feedback on Assessment

Feedback and advice on your workshop activities and your medal-winning activities will be given in class either upon request or as the teaching team observe you. You will also receive feedback on activities from the NoobLab environment. You may also receive feedback and/or feedforward by email from the team commenting on your progress as a whole, and general feedback/feedforward will be available for the cohort as a whole on Studyspace. It is important to pay close attention to feedback when it is received.

READING LIST

Please note the books given in the Module Descriptor are indicative and represent an old version of the module. The core texts for the module this year are:

Jon Duckett: HTML & CSS: Design and Build Web Sites

- http://www.amazon.co.uk/HTML-CSS-Design-Build-Sites/dp/1118008189/ref=la_B001IR3Q7I_1_1?s=books&ie=UTF8&qid=143920

Jon Duckett: JavaScript and JQuery: Interactive Front-end Web Development
- http://www.amazon.co.uk/JavaScript-JQuery-Interactive-Front-end-Development/dp/1118531647/ref=la_B001IR3Q7I_1_3?s=books&ie=UTF8&qid=1439202657&sr=1-3

Lynn Beighley and Michael Morrison: Head First PHP and MySQL
- http://www.amazon.co.uk/Head-First-MySQL-Lynn-Beighley/dp/0596006306

Cay Horstmann: Big Java (late objects)
- http://www.amazon.co.uk/Big-Java-Late-Objects-Horstmann/dp/1118087887/ref=sr_1_1?s=books&ie=UTF8&qid=1442237249&sr=1-1&keywords=big+java+late+objects

Do note that programming is a *practical*, hands-on activity that is fast-moving in terms of best practice. Do not make the mistake of thinking that there is a “holy grail” textbook out there that turns people into amazing programmers just by the act of possessing it! Textbooks can quickly become out of date. The best way to become a proficient programmer is by programming – not by reading a textbook! **I do NOT recommend spending any serious amounts of money on textbooks!**

Appendix: Module Descriptor

MODULE CODE: CI4100 **LEVEL: 4** **CREDITS: 30**

TITLE: **Programming I**

PRE-REQUISITES:

CO-REQUISITES:

MODULE SUMMARY *(INDICATIVE)*

This module will be taken by first year (level 4) students enrolled on Computer Science, Software Engineering, Information Systems and Joint Honours degrees. It is not assumed that students have prior programming experience. The teaching and learning is split between several units that will be directed at specific subsets of the above cohorts.
This provides each cohort with a schedule of activity that is appropriate for their background and future needs, while allowing a general visibility and structure of material for the entire year.

AIMS (DEFINITIVE)

- To introduce the essential concepts for a computer program
- To develop students' enthusiasm for, confidence in, and experience with programming by using practical examples
- To compare the similarities and differences of common programming languages

LEARNING OUTCOMES *(DEFINITIVE)*

On successful completion of the module, students will be able to:

1. Decompose a programming task into a set of smaller sub-tasks, expressed using standard control flow structures independent of any specific computing environment.
2. Demonstrate an understanding of the relevant structure and syntax of at least two programming environments, such as language-specific source code and/or mark-up languages permitting procedural instruction.
3. Demonstrate the appropriate use of variables, arrays of variables, expressions, subroutines, and conditional and iterative control flow structures.
4. Demonstrate an understanding of elementary object oriented concepts, such as classes and objects.
5. Use the available library methods to incorporate into their work some elementary user input, and visual output; where necessary testing the validity of such input, and appropriately structuring the output.
6. Write and use simple tests to validate the structured documents and software components they have written; use appropriate tools (such as debugging environments) to locate and fix errors that they find.

CURRICULUM CONTENT (INDICATIVE)

- Introduction to programming concepts in a language independent environment, such as variables, conditions, iterations and subroutines.
- Analysis of and practice at the expression of programming tasks as algorithms using these programming concepts.

- Introduction to variables, data types, logical & arithmetic operators, expressions, statements, conditions, and loops, using programming languages such as Java, Javascript, and C/C++.
- Drawing shapes on grids to make games. Using mouse input. Model-view controller, arrays.
- Introduction to the HTML syntax, structure, common elements; use of style sheets
- Javascript for form processing and other interactive components in an HTML page

TEACHING AND LEARNING STRATEGY (INDICATIVE)

The module material will be divided into approximately 7 units of taught material, unit topics to include problem solving, HTML, Javascript, Java and C++ Students will be directed to engage in four units, according to the required emphasis of their course.
Each unit will consist of approximately 11 hours of lecture, 11 hours of practical and a further 44 hours of self-directed study.

In addition to these units of activity, a further 6 hours of lecture and 6 hours of Practical will be scheduled for specific activity weeks and revision weeks during the semester. A further 32 hours of self-directed study is advised during these periods.

BREAKDOWN OF TEACHING AND LEARNING HOURS

DEFINITIVE KIS CATEGORY	INDICATIVE DESCRIPTION	HOURS
Scheduled learning and teaching	Lectures, tutorials, workshops, and exercises.	**100**
Guided independent study	Online learning materials including guided exercises and formative tests (with integrated support and automatic feed forward to tests).	**200**
	Total	**300**

ASSESSMENT STRATEGY *(INDICATIVE)*

In order to help students on this module achieve their full potential, formative assessment opportunities will be provided as appropriate throughout the module. Examples of formative assessments include worked exercises which emulate aspects of the major assessment and lab work. Feedback on coursework represents an additional opportunity for formative learning and will be given in writing and/or verbally. Formative feedback will be will be provided in various forms such as during short (10 - 15 minutes) feedback sessions. The formative feedback is designed to inform student preparation for the summative assessment which may be within the same module or feed forward across the degree programme.

MAPPING OF LEARNING OUTCOMES TO ASSESSMENT STRATEGY *(INDICATIVE)*

LEARNING OUTCOME	*ASSESSMENT STRATEGY*
On completion of the module, students will be able to:	
1) Decompose a programming task into a set of smaller sub-tasks, expressed using standard control flow structures independent of any specific computing	In-class multiple choice tests to establish basic ontology and concepts. Group exercises to solve problems, assessed through presentations.

environment.	
2) Demonstrate an understanding of the relevant structure and syntax of at least two programming environments, such as language-specific source code and/or mark-up languages permitting procedural instruction.	Weekly assignments (coursework) comprising a set of graded exercises. In-class multiple choice tests to assess understanding and recall of syntax & structure.
3) Demonstrate the appropriate use of variables, arrays of variables, expressions, subroutines, and conditional and iterative control flow structures.	In-class tests in which students modify source code to change the specification of a program.
4) Demonstrate an understanding of elementary object oriented concepts, such as classes and objects.	In-class tests in which students modify source code to change the specification of a program.
5) Use the available library methods to incorporate into their work some elementary user input, and visual output; where necessary testing the validity of such input, and appropriately structuring the output.	Weekly assignments (coursework) comprising a set of graded exercises. In-class tests in which students work on similar problems, in an open-book environment
6) Write and use simple tests to validate the structured documents and software components they have written; use appropriate tools (such as debugging environments) to locate and fix errors that they find.	Individual or Group assignment in which they inspect, debug, test, fix and modify an example computer program.

BREAKDOWN OF MAJOR CATEGORIES OF ASSESSMENT

DEFINITIVE KIS CATEGORY	INDICATIVE DESCRIPTION	PERCENTAGE
Coursework	Portfolio of in-class tests; multiple choice tests; weekly graded exercises; implementations	**100%**
	Total	**100%**

ACHIEVING A PASS (DEFINITIVE)

It *IS NOT* a requirement that any major assessment category is passed separately in order to achieve an overall pass for the module

BIBLIOGRAPHY *(INDICATIVE)*:

Core Text(s):

Robertson, L.A (2004). “Simple Program Design: A step by step approach”. Thomson.

Charatan, Q. and Kans, A. (2009) Java in two semesters, 3rd edn. London: McGraw Hill Higher Education

J. Zeldman and E. Marcotte (2010), “Designing with Web Standards", New Riders

Recommended Reading:

Sprankle M (2006). “Problem Solving and Programming Concepts”. Pearson.

Lewis J and Loftus W (2007). “Java Software Solutions: Foundations of Program Design”. Addison-Wesley.

Charatan Q and Kans A (2001). “Java the first semester”. McGraw-Hill.

Currie, Edward (2006) Fundamentals of programming using Java. London: Thomson Learning.

Rasmussen, R., Mughal, K. and Hamre T. (2007) Java actually: a first course in programming. London: Thomson Learning.

Vickers, Paul (2008) How to think like a programmer: problem solving for the bewildered. London: Cengage Learning.

Savitch, W. (2010) Absolute Java, 4th International edn. London: Pearson Education

Horstmann, Cay S. (2010) Big Java, 4th edn., International student version. Hoboken, N.J.: Wiley.

P. Carey (2006), “Creating Web Pages with HTML, XHTML, and XML”, Thomson Course

Technology

D. Oliver and M. Morrison (2003), "Teach Yourself HTML and XHTML in 24 Hours", SAMS

D. Gosselin, (2003) "Introductory XHTML", Thomson Course Technology

P. K. Yuen and V. Lau, (2003), "Practical Web Technologies", Addison-Wesley

A. Walter (2010), “A Holistic Approach to Web Design", New Riders