

Programming for Computer Scientists

CS118
Dr Stephen Jarvis



What is the course all about?

- ❑ Is it just computer programming?
- ❑ Solving complex problems using computers
 - ❑ Learning a programming language
 - ❑ Learning how to recognise the complexities of a problem and describe the automation of a solution
 - ❑ Learning how to engineer a solution: including the processes of *design, build and test*
- ❑ These are the basic skills that underpin many computer related professions: *software engineering, software consultancy, programming, software analyst*

Course documentation

- ❑ Lecture notes
 - ❑ You will be guaranteed notes for each lecture
- ❑ *The Guide*
 - ❑ Contains all necessary *administrative information*
 - ❑ Contains all the *problem sheets* for the seminars
 - ❑ Contains all the documentation on *coursework*
 - ❑ Available from the Computer Science reception
- ❑ Course web page
www.dcs.warwick.ac.uk/people/academic/Stephen.Jarvis/cs118/

What do I need to do?

- Lectures (18)
 - ❑ Monday 1-2pm (L3); Thursday 3-4pm (L3)
- Seminars (1 per week)
 - ❑ **Some seminars begin at 5pm today**
 - ❑ Find out which seminar you are to attend
 - ❑ **First seminar - meet in DCS reception**
- Two pieces of coursework
 - ❑ First deadline: Wednesday 26 October (Week 5)
 - ❑ Second deadline: Friday 2 December (Week 10)
- One class test
 - ❑ Monday 31 October (Week 6)

Assessment

- 2 hour Examination (60%)
 - ❑ Week 1 (or 2) Term 3
- Continuous assessment (40%)
 - ❑ 1 hour class test
 - ❑ Coursework 1 and coursework 2
- Seminars
 - ❑ You will not be formally graded on your seminar work
 - ❑ You should attempt the problem sheets and hand them in to your seminar tutor 24 hours before the seminar begins (seminar 1 is an exception)

Computer access and getting started

Computer Accounts

- ❑ University IT Service
 - Online Enrolment System (Computer Use Registration)
- ❑ Department of Computer Science
 - Different from your University account
 - Will be set up automatically
 - Account details mailed to your University account

Seminar 1

- ❑ Meet in DCS reception
- ❑ Make sure you bring your DCS account details and your copy of *The Guide*
- ❑ One-stop introduction to DCS computers, UNIX and Java

Getting started cont...

Thursday (today) 5-6pm seminar groups

0522528 Alzal Sarfraz 0520590 Aiyer Anand 0506858 Anderson Steven 0516970 Bai Haoming 0524072 Baker Michael 0529115 Belskis Vladimirs 0502021 Berragan James 0416024 Blohm Daniel 0505922 Boulter Michael 0520058 Broadbent Philip 0513576 Brookes Joanne 0502157 Byard David 0520088 Calogriou Andrew 0512947 Carpenter Andrew 0513803 Cebula Richard	0534089 Chan Wing 0511210 Chi Yixing 0508457 Chia Xin 0526614 Cipriani Alex 0520079 Clavey Jonathan 0515575 Crane Alexander 0518756 Dagma Djallal 0410725 Datskova Olga 0525309 Demetriou Philippos 0524172 du Shufen 0520994 Duke David 0521958 Duy Linh To 0401498 Ebenezer Timothy 0509858 Emmott Frederick 0414653 Evans John	0502730 Falconer Richard 0503985 Franks Henry 0518676 Fu Tian 0512447 Fuller Charles 0507445 Fullick Andrew 0526508 Gascoigne Joel 9923287 Greenwood Richard 0519038 Griffin James 0524122 Gynn Sam 0504569 Hallett Benjamin 0523756 Hazelden Alan 0512165 Herbert Gareth 0518864 Holder Fiona 0514922 Huang Xiao 0533392 Jarman Joseph
--	---	---

THE UNIVERSITY OF WARWICK
CS118: Programming for Computer Scientists 8

Practicalities

Contacting me

- Office: 2.07 in Computer Science Department
- Email: Stephen.Jarvis@dcs.warwick.ac.uk
- Office hours: Monday, Thursday

Contacting your seminar tutor

- Details on the course web page

Additional help

- Dedicated Teaching Fellow (Zabin Visram): zabin@dcs.warwick.ac.uk

Forum

<http://forums.warwick.ac.uk/departments/computer-science/ugyear1/cs118>

THE UNIVERSITY OF WARWICK
CS118: Programming for Computer Scientists 9

Books

- It would be good to have access to one book
 - Option 1: Library
 - Option 2: Get one second-hand (see Forum and *The Guide*)
 - Option 3: Buy a new one
- Book recommendations
 - Introduction to Java Programming Comprehensive*, Liang, Prentice Hall. ISBN 0131489526 (£60.99 on amazon)
 - Understanding Java*, Cornelius, Addison Wesley. ISBN 0-201-71107-9 (£40.99 from amazon)
 - Core Java 2: Volume 1-Fundamentals*, Horstman and Cornell, PH PTR and Sun Microsystems. ISBN 0-13-089468-0 (£27.99 from amazon)

THE UNIVERSITY OF WARWICK
CS118: Programming for Computer Scientists 10

A long time ago...

1940s

- Computers the size of this room
- Costing millions of 1940s dollars
- As powerful as a modern calculator

Programming was done using a *Machine Language*

- Instructions that directly controlled the processor
- Typically expressed as 8-part sequences of numbers expressed in base (2, 8 or) 16
- Fetch data, calculate, store data, fetch data ... etc.

THE UNIVERSITY OF WARWICK
CS118: Programming for Computer Scientists 13

Greatest Common Divisor (GCD) program

Program which calculates the GCD using Euclid's algorithm

To compute the GCD of a and b , check to see if a and b are equal. If so, print one of them and then stop. Otherwise, replace the larger one by their difference and repeat.

The program is written in machine language, hexadecimal (base 16) for the MIPS R4000 processor

```
27bdf0d0 afbf0014 0c1002a8 00000000 0c1002a8 afa2001c 8fa4001c
00401825 10820008 0064082a 10200003 00000000 10000002 00832023
00641823 1483ffffa 0064082a 0c1002b2 00000000 8fbf0014 27bd0020
03e00008 00001025
```

THE UNIVERSITY OF WARWICK
CS118: Programming for Computer Scientists 14

Greatest Common Divisor (GCD) program: Assembly code

Problems with this programming technique

- Enormously tedious; error prone

After a while machine languages were replaced by *Assembly Languages*

- Each hexadecimal number replaced by a mnemonic abbreviation, e.g.


```
addiu      replaced      27bdf0d0
```
- Translating from mnemonic to machine language was done by an *assembler*

```

graph LR
    A[Assembly code] -- assemble --> B[Machine code]
  
```

THE UNIVERSITY OF WARWICK
CS118: Programming for Computer Scientists 17

Greatest Common Divisor (GCD) program: Assembly code

Problems with this programming technique

- Enormously tedious; error prone

After a while machine languages were replaced by *Assembly Languages*

- Each hexadecimal number replaced by a mnemonic abbreviation, e.g.

```
addiu    replaced    27bdfdd0
```
- Translating from mnemonic to machine language was done by an *assembler*

```
addiu    sp,sp,-32      move    v1,v0      B: subu  a0,a0,a0
sw       ra,20(sp)      beq     a0,v0,D    C: hne   a0,v1,A
jal      getint         slt     at,v1,a0 slt  at,v1,a0
nop                                     A: beq   at,zero,B  D: jal   putint
jal      getint         nop                                     nop
sw       v0,28(sp)      b       C          lw      ra,20(sp)
lw       a0,28(sp)      subu   a0,a0,v1 addiu sp,sp,32
jz       jr            ra
move    v0,zero
```

High-level programming languages

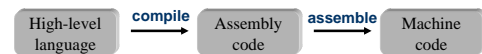
1950s

- New computers evolved, competing designs
- Programs had to be re-written for every new machine
- Became impossible to maintain code

Need for machine independent languages

- In which the program more closely resembled the task (which in those days tended to be mathematical)
- First high-level language – Fortran
- Other languages soon followed – Lisp, Algol etc.

Translating high-level languages required a *compiler*



Programming language spectrum

Today there are loads of different HLPLs

- Evolution*: we find better ways of doing things
- Special purposes*: Prolog is good for representing logical relationships between data, for example
- Personal preference*: Some people like C; some hate it
- Ease of use*: Basic was popular because it was easy
- Compilers*: contributed to Fortran's longevity
- Economics*: IBM supported Cobol, DoD supported Ada

Programming

- Historical view: 'telling a computer what to do'
- Programmers view: 'a means of expressing algorithms'
- Today's view: 'the art of telling another human being what one wants the computer to do'

Programming language spectrum

Declarative

functional	<i>Lisp, ML, Haskell</i>
dataflow	<i>Id, Val</i>
logic, constraint	<i>Prolog, VisiCalc</i>

Imperative

von Neumann	<i>Fortran, Pascal, C</i>
object-oriented	<i>Smalltalk, Java, C#</i>

Why Java?

Closely related to the von Neumann languages

- Which have been the most successful / widely used

We will learn programming in a von Neumann style...

...then add objects half way through term

- Encourages structured programming, re-use and abstraction

Modern, state-of-the-art

- Web development, networking, databases etc.

Advanced features

- Automatic memory management
- Error checking at compile and run-time

Good supply of standard library functions

- GUI builders, networking, input/output, security, data manipulation...

Java

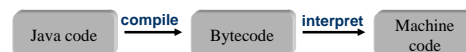
Java (originally called 'Oak')

- Designed by Sun Microsystems in 1991
- 1995 re-designed for Internet applications

Java is more than just a language, it is a platform

- A high-level language specification
- A low-level bytecode specification
- An API specification (libraries)

Java programs compiled to portable bytecode; the bytecode is then *interpreted* to machine instructions



Java

1. Programmer creates high-level instructions
2. Converts these to bytecode using a compiler
3. Puts the bytecode anywhere (on the web say)

```

    graph LR
      A[Java code] -- compile --> B[Bytecode]
      B -- interpret --> C[Machine code]
      subgraph RedBox [ ]
        A
        B
      end
  
```

CS118: Programming for Computer Scientists 32

Java

1. Someone who wants to run the code can pick it up from anywhere (the web say)
2. And using an interpreter run the code on their machine

```

    graph LR
      A[Java code] -- compile --> B[Bytecode]
      B -- interpret --> C[Machine code]
      subgraph RedBox [ ]
        B
        C
      end
  
```

CS118: Programming for Computer Scientists 33

Java

This has a number of advantages:

1. Original programmer keeps their code safe
2. They also don't have to worry about the platform that the code will eventually run on
3. Easy to distribute code

```

    graph LR
      A[Java code] -- compile --> B[Bytecode]
      B -- interpret --> C[Machine code]
  
```

CS118: Programming for Computer Scientists 34

Java

This has a number of advantages:

1. Original programmer keeps their code safe
2. They also don't have to worry about the platform that the code will eventually run on
3. Easy to distribute code

This is why the Java model has become so popular (and the fact that it was in the right place at the right time)

```

    graph LR
      A[Java code] -- compile --> B[Bytecode]
      B -- interpret --> C[Machine code]
  
```

CS118: Programming for Computer Scientists 35

GCD in Java

```

/* A java program that uses Euclid's algorithm to calculate the GCD */
import uk.ac.warwick.dcs.util.io.IO;
public class GCD
{
    public static void main (String [] args)
    {
        int a = IO.readInt("Enter first number: "); // input a
        int b = IO.readInt("Enter second number: "); // input b

        while (a != b) // while a and b not equal
        { if (a > b) a = a - b else b = b - a } // replace larger by the // difference...

        System.out.print("The GCD is: "); // then print the result
        System.out.println(a);
    }
}
  
```

CS118: Programming for Computer Scientists 37

GCD in Java: Comments

Notes to the programmer

```

/* A java program that uses Euclid's algorithm to calculate the GCD */
import uk.ac.warwick.dcs.util.io.IO;
public class GCD
{
    public static void main (String [] args)
    {
        int a = IO.readInt("Enter first number: "); // input a
        int b = IO.readInt("Enter second number: "); // input b

        while (a != b) // while a and b not equal
        { if (a > b) a = a - b else b = b - a } // replace larger by the // difference...

        System.out.print("The GCD is: "); // then print the result
        System.out.println(a);
    }
}
  
```

CS118: Programming for Computer Scientists 38

GCD in Java: Reserved words

Part of the Java high-level language

```
/* A Java program that uses Euclid's algorithm to calculate the GCD */  
  
import uk.ac.warwick.dcs.util.io.IO;  
public class GCD  
{  
    public static void main (String [] args)  
    {  
        int a = IO.readInt("Enter first number: "); // input a  
        int b = IO.readInt("Enter second number: "); // input b  
  
        while (a != b) // while a and b not equal  
        { if (a > b) a = a - b else b = b - a } // replace larger by the  
        // difference...  
  
        System.out.print("The GCD is: "); // then print the result  
        System.out.println(a);  
    }  
}
```

GCD in Java: Statements

Perform some action, terminated with a semi-colon

```
/* A Java program that uses Euclid's algorithm to calculate the GCD */  
  
import uk.ac.warwick.dcs.util.io.IO;  
public class GCD  
{  
    public static void main (String [] args)  
    {  
        int a = IO.readInt("Enter first number: "); // input a  
        int b = IO.readInt("Enter second number: "); // input b  
  
        while (a != b) // while a and b not equal  
        { if (a > b) a = a - b else b = b - a } // replace larger by the  
        // difference...  
  
        System.out.print("The GCD is: "); // then print the result  
        System.out.println(a);  
    }  
}
```

GCD in Java: Program blocks

Groups of statements delimited by braces

```
/* A Java program that uses Euclid's algorithm to calculate the GCD */  
  
import uk.ac.warwick.dcs.util.io.IO;  
public class GCD  
{  
    public static void main (String [] args)  
    {  
        int a = IO.readInt("Enter first number: "); // input a  
        int b = IO.readInt("Enter second number: "); // input b  
  
        while (a != b) // while a and b not equal  
        { if (a > b) a = a - b else b = b - a } // replace larger by the  
        // difference...  
  
        System.out.print("The GCD is: "); // then print the result  
        System.out.println(a);  
    }  
}
```

{
}
}

GCD in Java: Methods

Named collection of statements that performs some activity

```
/* A Java program that uses Euclid's algorithm to calculate the GCD */  
  
import uk.ac.warwick.dcs.util.io.IO;  
public class GCD  
{  
    public static void main (String [] args)  
    {  
        int a = IO.readInt("Enter first number: "); // input a  
        int b = IO.readInt("Enter second number: "); // input b  
  
        while (a != b) // while a and b not equal  
        { if (a > b) a = a - b else b = b - a } // replace larger by the  
        // difference...  
  
        System.out.print("The GCD is: "); // then print the result  
        System.out.println(a);  
    }  
}
```

main,
where in
fact the
execution
begins

{
}
}

GCD in Java: Classes

Programs representing some particular activity

```
/* A Java program that uses Euclid's algorithm to calculate the GCD */  
  
import uk.ac.warwick.dcs.util.io.IO;  
public class GCD  
{  
    public static void main (String [] args)  
    {  
        int a = IO.readInt("Enter first number: "); // input a  
        int b = IO.readInt("Enter second number: "); // input b  
  
        while (a != b) // while a and b not equal  
        { if (a > b) a = a - b else b = b - a } // replace larger by the  
        // difference...  
  
        System.out.print("The GCD is: "); // then print the result  
        System.out.println(a);  
    }  
}
```

A program
that
calculates
the GCD

{
}
}

GCD in Java: Libraries

Collections of previously written programs

```
/* A Java program that uses Euclid's algorithm to calculate the GCD */  
  
import uk.ac.warwick.dcs.util.io.IO;  
public class GCD  
{  
    public static void main (String [] args)  
    {  
        int a = IO.readInt("Enter first number: "); // input a  
        int b = IO.readInt("Enter second number: "); // input b  
  
        while (a != b) // while a and b not equal  
        { if (a > b) a = a - b else b = b - a } // replace larger by the  
        // difference...  
  
        System.out.print("The GCD is: "); // then print the result  
        System.out.println(a);  
    }  
}
```

GCD in Java: Programming style

Important if you want people to be able to read your code

```
/* A java program that uses Euclid's algorithm to calculate the GCD */
import uk.ac.warwick.dcs.util.io.IO;
public class GCD
{
    public static void main (String [] args)
    {
        int a = IO.readInt("Enter first number: "); // input a
        int b = IO.readInt("Enter second number: "); // input b

        while (a != b) // while a and b not equal
        { if (a > b) a = a - b else b = b - a } // replace larger by the // difference...

        System.out.print("The GCD is: "); // then print the result
        System.out.println(a);
    }
}
```

Compiling and running the program

Type the program in and save it as `GCD.java`

Compile the program by typing

```
javac GCD.java
```

this converts the high-level description to bytecode (assembly code) stored in `GCD.class`

Run the program by typing

```
java GCD
```

this converts (at run-time) the bytecode to machine code

Error Detection

You may have typed the program in wrong

```
/* A java program that uses Euclid's algorithm to calculate the GCD */
import uk.ac.warwick.dcs.util.io.IO;
public class GCD
{
    public static void main (String [] args)
    { ...

```

The compiler will signal an error

```
GCD.java:3: Class or interface declaration expected class GCD
^
1 error
```

Testing the program

Check the output of the program

```
input: 33 and 110
```

```
output: The GCD is: 11
```

Always check the program against a good selection of input.

Testing is often forgotten!

What to do next...

- Write down the details of your CS account (from the email you will find in your University inbox)
- Check the CS notice board / course web-page to see which seminar group you are in
- Pick up a copy of *The Guide*; locate a book
- **Meet in the Computer Science reception for your first seminar:** bring your *Guide* and your CS account details