Comp1202: Introduction I

Welcome!
Welcome to Programming I

Son Hoang  
(t.s.hoang@ecs.soton.ac.uk)

Heather Packer  
(hp3@esc.soton.ac.uk)

Jian Shi  
(jian.shi@soton.ac.uk)
Purpose of this Module

“This module aims to introduce students to the principles of programming using an object oriented approach, and to provides them with the programming skills necessary to continue the study of computer science. Java is used as the introductory language.”

- Writing and running programs
- Compilation, Interpretation and the Java Virtual Machine
- Variables, Objects, Primitives and Scope
- Methods
- Computational Thinking
- Constructors
- Loops and Arrays
- Collections and Iterators
- The Java Library
- Integrated Development Environments
- Testing and Debugging
- Software Design (What makes a good program)
- Super and Sub Classes (Inheritance)
- Polymorphism and Dynamic Binding
- Abstract Classes and Interfaces
- Designing Applications (Moving from problem to solution)
Main Course Structure

- Two recorded sessions each week
- An MS Teams FAQ session Monday 11:00
- An interactive session B32/1015
  - Tuesday 09:00/12:00/15:00
  - Thursday 12:00/15:00
  - Weeks 1, 2, 4, 6, 8, 10
- Also a Lab each week (worth 20%)
  - Thursday 09:00
  - MS Teams
- Other Assessments
  - Coursework
    (set in week 4, due in week 10, worth 40%)
  - Exam (after X-mas, worth 40%)

More details on next few slides
About the Recorded Sessions

• Watch before the Monday FAQ session.
• Slides and a summary video on EdShare
  – Login using your University credential
• Recording sessions on Panopto
  – Login via Blackboard using your University credential
• Reading sections in the BlueJ book
About the FAQ Sessions

• Monday 11:00 on Microsoft Teams.

• General questions about lectures, labs, etc.
  – Can you please explain this on slide X in Lecture Y?
  – How do we suppose to submit the lab work?
  – …

• No specific technical questions please
  – You can save them for the Labs sessions, …
  – … or the Ground Controllers sessions.
  – I’m stuck, please help!!!
  – Can you please help me to see what this error is?
About the Interactive Sessions

• B32/1015 (i.e., this lecture theatre)
• Week 1, 2, 4, 6, 8, 10
• Exercise-driven
• You can ask question via Microsoft Teams
  – You do not have to shout
  – You can type in the chat or speak
  – Please join with microphone and camera off
  – Unmute and turn on the camera when speaking.
About Labs

• 10 labs every Thursday

• Starting October 8\textsuperscript{th} (i.e., this week)

• You will be emailed your lab group name before Thursday

• On Thursday join the Lab meeting and then go to your group

• Demonstrators will be available to help you, fill in the form to be put into a queue
About Labs

• Group Rules
  – Practice common courtesy
  – There are no dumb questions
  – Let someone know when you are stuck, either your peers in your breakout room or fill in the demonstrator form
  – DO NOT cut and paste code
  – DO NOT share your code
Code Functionality Test Harness

- Code Functionality is assessed using a **test harness**
- You will **receive an email** detailing any
  - Errors and whether your code passed the tests
  - Styling problems (incorrect indentation, naming problems,..)
- **Make sure you follow the lab instructions carefully**, otherwise your code might not meet the spec
  - That includes spelling in outputs!
- You must **structure** your code **exactly** the same as specified in the lab otherwise the test harness won’t work
- **Submit your code well before the deadline**, otherwise you might encounter an issue making you miss the deadline
Submission

• You must submit labs
  – 1, 2 and 3 at the end of the 3\textsuperscript{rd} week
  – 4, 5 and 6 at the end of the 6\textsuperscript{th} week
  – 7, 8 and 9 at the end of the 9\textsuperscript{th} week

• If you’d like to discuss your code’s style and comments please talk to a demonstrator
About Coursework and Exam

• Coursework
  – Set in Week 4
  – Due in Week 10 (just before the X-mas break)
  – Feedback in the new year (often before the exam)
  – Tentative topic: ECS Battle Arena
    • Might have some competitions (to be decide)

• Exam
  – In the past: Multiple choice computer-based exam
  – To be decided according to University regulation
Additional Streams

• **Space Cadets**
  – For people who are more experienced programmers
  – Run by Son and/or some demonstrators
  – Optional weekly challenge and discussion
    • Friday 16:00 (MS Teams)
    • Weeks 1-7 (i.e., starts this week!)

• **Ground Controllers**
  – For people who are new to programming
  – Run by Son and demonstrators
  – Optional weekly workshop
    • Week 2, 3, 4 B59/Level 3 Lab (i.e., starts next week!)
      – Wed 9:00 or 12:00 (registration on Ground Controllers page)
    • Weeks 5-9 - MS Teams - Friday 16:00
BlueJ

• The main course text is *Objects First with BlueJ* (available from the library)

• Although we will start with using Java from the command line, we will soon introduce you to the BlueJ environment

• BlueJ is an environment designed for learning, but later in the course we will change over to use a proper professional environment
A Dirty Secret

• No matter how we teach you will mainly learn through practice!

• Programming is the single most important skill for a computer scientist or IT specialist
  – Systematic thinking and problem solving
  – Abstraction and data modeling

• Did we mention that you need to PRACTICE?
PRACTICE!

“I've often thought that sucking less every year is how humble programmers improve. You should be unhappy with code you wrote a year ago.”

- Jeff Atwood, http://www.codinghorror.com/
  (Founder of stackoverflow.com)

“I have no talent. What I do have is a lot of practice. And I am not talking about occasionally dabbling in Ruby on the weekends. I am talking about the kind of practice where I beat code that isn’t working into submission (though often times the code wins).”

Your Lecturer: A Warning from History
Online Notes Wiki

https://secure.ecs.soton.ac.uk/module/comp1202/

leads to

https://secure.ecs.soton.ac.uk/student/wiki/w/COMP1202
Timetables

• Your timetable shows all of the sessions / labs allocated to your modules

• Modules do not always use all of those sessions!
  – For example, we have optional seminars

• So check the notes pages of each course for more info and adjust your timetable accordingly!
<table>
<thead>
<tr>
<th>Week</th>
<th>Start Date</th>
<th>Topic</th>
<th>Section of Objects First Textbook</th>
<th>Recording Sessions</th>
<th>Online FAQ (Mon 11am) - Microsoft Teams</th>
<th>Interactive Sessions (Tue 9-10, 12-13, 15-16, Thu 12-13, 15-16) - B32/1015</th>
<th>F2F Labs (Wed 9-11, 12-14) - B59/Level 3</th>
<th>Interactive Sessions (Fri 16-18) - Microsoft Teams</th>
<th>Labs (Thu 9-11) - Microsoft Teams</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>05/10/20</td>
<td>Introduction</td>
<td>1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13</td>
<td>(1) Java (sh)</td>
<td>n/a</td>
<td>Starting Out (sh)</td>
<td>n/a</td>
<td>Space Cadets introduction (sh, jhs)</td>
<td>Hello World and Conditionals (hp)</td>
</tr>
<tr>
<td>2</td>
<td>12/10/20</td>
<td>Programming in Java</td>
<td>1.1, 1.2, 1.5, 2.15, 2.16, 2.17, 2.18, 2.19</td>
<td>(2) Variables, Objects, Primitives and Scope (js)</td>
<td>(3) Methods (js)</td>
<td>FAQ on (1), (2), and (3) (sh, js, hp)</td>
<td>Exercises on (1) and (2) (js)</td>
<td>Ground Controllers (sh)</td>
<td>Space Cadets (sh)</td>
</tr>
<tr>
<td>3</td>
<td>19/10/20</td>
<td>Building Better Programs/Objects</td>
<td>2.4, 2.5, 2.6, 2.7, 2.8, 2.9, 2.10</td>
<td>(4) Computational Thinking (dem)</td>
<td>(5) Constructors (js)</td>
<td>FAQ on (4) and (5) (sh, js, hp)</td>
<td>n/a</td>
<td>Ground Controllers (sh)</td>
<td>Space Cadets (sh)</td>
</tr>
<tr>
<td>4</td>
<td>26/10/20</td>
<td>Loops and Arrays</td>
<td>4.9, 4.10</td>
<td>(6) Loops and Arrays (sh)</td>
<td>(7) Collections and Iterators (sh)</td>
<td>FAQ on (6) and (7) (sh, js, hp)</td>
<td>Exercises on (4), (5), (6), (7) (sh)</td>
<td>Ground Controllers (sh)</td>
<td>Space Cadets (sh)</td>
</tr>
<tr>
<td>5</td>
<td>02/11/20</td>
<td>Environments and the Java Lib</td>
<td>4.12, 4.14, 4.16, 4.18, 4.20, 4.22, 4.24</td>
<td>(8) The Java Library (sh)</td>
<td>(9) Changing IDE (sh)</td>
<td>FAQ on (8) and (9) (sh, js, hp)</td>
<td>n/a</td>
<td>Ground Controllers (sh) and Space Cadets (sh)</td>
<td>Arrays, HashMaps and APIs (hp)</td>
</tr>
<tr>
<td>6</td>
<td>09/11/20</td>
<td>Inheritance</td>
<td>8.2, 8.3, 8.4</td>
<td>(10) Super and Sub-classes (js)</td>
<td>(11) Polymorphism (js)</td>
<td>Handout for inheritance 1+2.pdf</td>
<td>FAQ on (10) and (11) (sh, js, hp)</td>
<td>Exercises on (8), (9), (10), and (11) (js)</td>
<td>n/a</td>
</tr>
<tr>
<td>7</td>
<td>16/11/20</td>
<td>Building Better Classes</td>
<td>3.13</td>
<td>(12) Software Design (dem)</td>
<td>(13) Testing and Debugging (sh)</td>
<td>FAQ on (12) and (13)</td>
<td>n/a</td>
<td>n/a</td>
<td>Ground Controllers (sh) and Space Cadets (sh)</td>
</tr>
<tr>
<td>8</td>
<td>23/11/20</td>
<td>Object-Oriented Design</td>
<td>10.3, 10.4, 10.6, 10.7</td>
<td>(14) Designing Applications (dem)</td>
<td>(15) Abstract Classes and Interfaces (js)</td>
<td>Panopto Handout.pdf</td>
<td>FAQ on (14) and (15) (sh, js, hp)</td>
<td>Exercises on (12), (13), (14), and (15) (js)</td>
<td>n/a</td>
</tr>
<tr>
<td>9</td>
<td>30/11/20</td>
<td>Coding in Practice</td>
<td>10.10</td>
<td>(16) Coding Dojo - part 1 (sh, js)</td>
<td>Slides</td>
<td>FAQ on (16)</td>
<td>n/a</td>
<td>n/a</td>
<td>Ground Controllers (sh)</td>
</tr>
<tr>
<td>10</td>
<td>07/12/20</td>
<td>Revision and Other Languages</td>
<td>10.10</td>
<td>Revision Lecture (sh)</td>
<td>Panopto</td>
<td>Languages Balloon Debate (sh)</td>
<td>Revision Exercises (sh)</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>11</td>
<td>04/01/21</td>
<td>Back-up</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>12</td>
<td>11/01/21</td>
<td>Back-up</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
</tr>
</tbody>
</table>
How to Get Help

• **Don’t** email (get’s lost in the avalanche)
• **Don’t** drop by the office (not there atm, often busy)
• **Don’t** come down to ask questions at the end of the session
  – Not CoVID secure
  – We will be outside (between B32 and B85)
• **Do** ask questions in the sessions
• **Do** ask questions in the labs
• **Do** ask each other!
  – Be careful of academic integrity.
• **Do** book 15-minute consultation slots
  – Monday 12:00—13:00 and Tuesday 17:00-18:00
    • with Son (details on the Resources page)
Comp1202:
Introduction II

Starting Out
In this part

• What is Programming?
• Programming Paradigms

• A Taste of Things to Come
  – Classes and Objects
  – Variables
  – Logic
  – Data Structures
  – Skills
What is Programming?

- Wikipedia (2019)
  - “is the process of designing and building an executable computer program for accomplishing a specific computing task ... The purpose of programming is to find a sequence of instructions that will automate the performance of a task for solving a given problem.”

- thefreedictionary.com (2019)
  - “creating a sequence of instructions to enable the computer to do something”

- dictionary.com (2019)
  - “the act or process of planning or writing a program”

  - “the writing or preparation of computer programs.”
What is Programming?

• Wikipedia (2019)
  – “is the process of designing and building an executable computer program for accomplishing a specific computing task ... The purpose of programming is to find a sequence of instructions that will automate the performance of a task for solving a given problem.”

• Thefreedictionary.com (2019)
  – “creating a sequence of instructions to enable the computer to do something”

• Dictionary.com (2019)
  – “the act or process of planning or writing a program”

• Oxford English Dictionary (2019)
  – “the writing or preparation of computer programs.”
What is Programming?

• Wikipedia (2018)
  – “is the process of designing and building an executable computer program for accomplishing a specific computing task ... The purpose of programming is to find a sequence of instructions that will automate the performance of a task for solving a given problem.”

• Thefreedictionary.com (2018)
  – “creating a sequence of instructions to enable the computer to do something”

• Dictionary.com (2019)
  – “the act or process of planning or writing a program”

• Oxford English Dictionary (2019)
  – “the writing or preparation of computer programs.”
What is Programming?

- Wikipedia (2018)
  - “is the process of designing and building an executable computer program for accomplishing a specific computing task ... The purpose of programming is to find a sequence of instructions that will automate the performance of a task for solving a given problem.”

- Thefreedictionary.com (2018)
  - “creating a sequence of instructions to enable the computer to do something”

- Dictionary.com (2019)
  - “the act or process of planning or writing a program”

  - “the writing or preparation of computer programs.”
Programming Flavours

• Procedural (e.g. C or Pascal)
  – Fixed list of instructions
  – Do this, Do that, if this then do that

• Declarative
  – More like declaring rules (or a grammar)
  – Behaviour emerges from the rules being applied
  – Examples
    • Functional Programming (e.g. Scheme or Haskell)
    • Logic Programming (e.g. Prolog)
Object Orientated

The main idea

– Everything is a Thing
– A Program is made up of Things interacting
– Things have both properties and behaviours
– E.g. Dogs

• What are the properties of a dog?
• What can a dog do?
Classes and Objects

All the properties of Dog can be wrapped up or contained in a class

- A class is like a Blue Print
- We can build many unique dogs from the same Blue Print
- These are called objects
- They all have similar properties as defined by the class

1 Class

http://www.animalblueprintcompany.com/
Building Blocks

• Objects and Classes are specific to Object Orientated Programming

• But there are more common, more fundamental Programming Principles that we will be covering in the course
Variables

• Like algebra
  \[ x = 4 \]

• Not so much like algebra
  \[ \text{name} = \text{“Rover”} \]

• Object properties are stored in variables
Methods

• Blocks of code that define a sequence of actions
• Object behavior is defined in methods
• Often this will use the object’s properties

```javascript
printDogsDetails()
{
    print name;
    print age;
}
```
## Logic

https://blackboard.soton.ac.uk/webapps/blackboard/content/launchAssessment.jsp?course_id=191255_1&content_id=4851249_1&mode=cpview

**True or False – the basis of all decisions**

<table>
<thead>
<tr>
<th>?</th>
<th>( x = 3 )</th>
<th><strong>Meaning</strong></th>
<th>( x = 11 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x == 4 )</td>
<td>( x ) is equal to 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( x &gt; 4 )</td>
<td>( x ) is greater than 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( x &lt;= 4 )</td>
<td>( x ) is smaller than or equal to 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( x != 4 )</td>
<td>( x ) is not equal to 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( x &gt; 2 ) &amp;&amp; ( x &lt; 8 )</td>
<td>( x ) is greater than 2 and ( x ) is smaller than 8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( x &gt; 8 ) || ( x &lt; 2 )</td>
<td>( x ) is greater than 8 or ( x ) is smaller than 2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Logic

### True or False – the basis of all decisions

| ? | x == 4 | x > 4 | x <= 4 | x != 4 | x > 2 && x < 8 | x > 8 || x < 2 |
|---|--------|------|-------|-------|---------------|---------------|
| ? | false  | false| true  | true  | true          | false         |
| x == 4 | false  | true | true  | false  | false         | true          |
| x > 4 | false  | true | true  | true  | true          | false         |
| x <= 4 | true  | true | true  | true  | true          | false         |
| x != 4 | true  | true | true  | true  | false         | true          |
| x > 2 && x < 8 | true  | true | true  | false  | false         | false         |
| x > 8 || x < 2 | false  | false | false  | false  | true          |
Control Flow

• If statements

```python
if size<10
    dog says “Yip Yip”
else
    dog says “Ruff Ruff”
```

• Loops

```python
while number<10
    dog says “Bark!”
    number = number + 1
```

Side Note:

Human readable versions of programs are called **Pseudocode**.

They look like real programs but are not as precisely defined.

They are good for communicating ideas and showing structure.
Data Structures

• More complex data...

• Arrays:
  – Like Lists, Tables, Matrices

• HashMaps
  – Associate a key with a value
  – Like a Dictionary
Skills

• Computational Thinking
• Program Design
• Choosing Tools (IDEs)
• Testing/Debugging
Summary

• What is Programming?
• Programming Paradigms
• A Taste of Things to Come
  – Classes and Objects
  – Variables
  – Logic
  – Data Structures
  – Skills