Programming in Java

Introduction and overview

School of Computing and Mathematical Sciences
Birkbeck, University of London

Welcome to the MSc Computer Science!

- MSc Computer Science $\approx$ content of undergraduate Computing degree compressed into 1 year full-time or 2 years part-time study

Welcome to the MSc Computer Science!

- MSc Computer Science $\approx$ content of undergraduate Computing degree compressed into 1 year full-time or 2 years part-time study
- 180 credits = 1800 hours of study

Welcome to the MSc Computer Science!

- MSc Computer Science $\approx$ content of undergraduate Computing degree compressed into 1 year full-time or 2 years part-time study
- 180 credits = 1800 hours of study
- You have all shown aptitude for the programme.

Welcome to the MSc Computer Science!

- MSc Computer Science $\approx$ content of undergraduate Computing degree compressed into 1 year full-time or 2 years part-time study
- 180 credits = 1800 hours of study
- You have all shown aptitude for the programme.
- Now you must put in the work!

Programming in Java

- 30 credits = 300 hours

Programming in Java

- 30 credits = 300 hours
- Full-time: 2 sessions a week over 1 term;
  part-time: 1 session a week over 2 terms

Programming in Java

- 30 credits = 300 hours
- Full-time: 2 sessions a week over 1 term;
  part-time: 1 session a week over 2 terms
- Programming: core **skill** for Computer Science graduates

Programming in Java

- 30 credits = 300 hours
- Full-time: 2 sessions a week over 1 term;
  part-time: 1 session a week over 2 terms
- Programming: core **skill** for Computer Science graduates
- Java: well suited for **large** software projects

Programming in Java

- 30 credits = 300 hours
- Full-time: 2 sessions a week over 1 term;
  part-time: 1 session a week over 2 terms
- Programming: core **skill** for Computer Science graduates
- Java: well suited for **large** software projects
- Assessment:
  70% coursework project (submission next term), 30% final exam

Programming in Java

- 30 credits = 300 hours
- Full-time: 2 sessions a week over 1 term;
  part-time: 1 session a week over 2 terms
- Programming: core **skill** for Computer Science graduates
- Java: well suited for **large** software projects
- Assessment:
  70% coursework project (submission next term), 30% final exam
- Module will be taught in "flipped classroom" mode
  $\Rightarrow$ no lectures, you have to come to class **prepared**

Programming in Java

- 30 credits = 300 hours
- Full-time: 2 sessions a week over 1 term;
  part-time: 1 session a week over 2 terms
- Programming: core **skill** for Computer Science graduates
- Java: well suited for **large** software projects
- Assessment:
  70% coursework project (submission next term), 30% final exam
- Module will be taught in "flipped classroom" mode
  $\Rightarrow$ no lectures, you have to come to class **prepared**
- Tentative session plan:

Programming in Java

- 30 credits = 300 hours
- Full-time: 2 sessions a week over 1 term;
  part-time: 1 session a week over 2 terms
- Programming: core **skill** for Computer Science graduates
- Java: well suited for **large** software projects
- Assessment:
  70% coursework project (submission next term), 30% final exam
- Module will be taught in "flipped classroom" mode
  $\Rightarrow$ no lectures, you have to come to class **prepared**
- Tentative session plan:
  - Recap of the material studied at home, address general questions

Programming in Java

- 30 credits = 300 hours
- Full-time: 2 sessions a week over 1 term;
  part-time: 1 session a week over 2 terms
- Programming: core **skill** for Computer Science graduates
- Java: well suited for **large** software projects
- Assessment:
  70% coursework project (submission next term), 30% final exam
- Module will be taught in "flipped classroom" mode
  $\Rightarrow$ no lectures, you have to come to class **prepared**
- Tentative session plan:
  - Recap of the material studied at home, address general questions
  - Work on exercises, get one-on-one help

Programming in Java

- 30 credits = 300 hours
- Full-time: 2 sessions a week over 1 term;
  part-time: 1 session a week over 2 terms
- Programming: core **skill** for Computer Science graduates
- Java: well suited for **large** software projects
- Assessment:
  70% coursework project (submission next term), 30% final exam
- Module will be taught in "flipped classroom" mode
  $\Rightarrow$ no lectures, you have to come to class **prepared**
- Tentative session plan:
  - Recap of the material studied at home, address general questions
  - Work on exercises, get one-on-one help
  - Live programming demo

Programming in Java

- 30 credits = 300 hours
- Full-time: 2 sessions a week over 1 term;
  part-time: 1 session a week over 2 terms
- Programming: core **skill** for Computer Science graduates
- Java: well suited for **large** software projects
- Assessment:
  70% coursework project (submission next term), 30% final exam
- Module will be taught in "flipped classroom" mode
  $\Rightarrow$ no lectures, you have to come to class **prepared**
- Tentative session plan:
  - Recap of the material studied at home, address general questions
  - Work on exercises, get one-on-one help
  - Live programming demo
- Additional online help sessions on Saturdays

Programming in Java

- 30 credits = 300 hours
- Full-time: 2 sessions a week over 1 term;
  part-time: 1 session a week over 2 terms
- Programming: core **skill** for Computer Science graduates
- Java: well suited for **large** software projects
- Assessment:
  70% coursework project (submission next term), 30% final exam
- Module will be taught in "flipped classroom" mode
  $\Rightarrow$ no lectures, you have to come to class **prepared**
- Tentative session plan:
  - Recap of the material studied at home, address general questions
  - Work on exercises, get one-on-one help
  - Live programming demo
- Additional online help sessions on Saturdays

$\Rightarrow$ Plenty of opportunity for **questions**!

Day 1: Here it all begins...

- Day 1: concepts from MSc Computer Science pre-reading **in Java**

Day 1: Here it all begins...

- Day 1: concepts from MSc Computer Science pre-reading **in Java**
- But we have not talked about the module setup before today...

Day 1: Here it all begins...

- Day 1: concepts from MSc Computer Science pre-reading **in Java**
- But we have not talked about the module setup before today...

$\Rightarrow$ Plan for Day 1:

Day 1: Here it all begins...

- Day 1: concepts from MSc Computer Science pre-reading **in Java**
- But we have not talked about the module setup before today...

$\Rightarrow$ Plan for Day 1:

- Overview and hands-on Java programming demo

Day 1: Here it all begins...

- Day 1: concepts from MSc Computer Science pre-reading **in Java**
- But we have not talked about the module setup before today...

$\Rightarrow$ Plan for Day 1:

- Overview and hands-on Java programming demo
- Work on exercises from pre-reading & worksheet on Moodle, one-on-one questions

Day 1: Here it all begins...

- Day 1: concepts from MSc Computer Science pre-reading **in Java**
- But we have not talked about the module setup before today...

$\Rightarrow$ Plan for Day 1:

- Overview and hands-on Java programming demo
- Work on exercises from pre-reading & worksheet on Moodle, one-on-one questions
- Questions, programming demo

Getting into a computer's mindset

Main goal: **solve problems** by writing computer programs.

Getting into a computer's mindset

Main goal: **solve problems** by writing computer programs.

Using computers for problem solving:

- Pro: computers do things very, very **fast**.
- Pro: computers follow their instructions **to the letter**.

Getting into a computer's mindset

Main goal: **solve problems** by writing computer programs.

Using computers for problem solving:

- Pro: computers do things very, very **fast**.
- Pro: computers follow their instructions **to the letter**.
- Con: computers follow their instructions **to the letter**.
- Con: computers need **detailed** instructions.

Getting into a computer's mindset

Main goal: **solve problems** by writing computer programs.

Using computers for problem solving:

- Pro: computers do things very, very **fast**.
- Pro: computers follow their instructions **to the letter**.
- Con: computers follow their instructions **to the letter**.
- Con: computers need **detailed** instructions.

$\Rightarrow$ The **programmer** must break down the solution into unambiguous small pieces + consider all eventualities.

Getting into a computer's mindset

Main goal: **solve problems** by writing computer programs.

Using computers for problem solving:

- Pro: computers do things very, very **fast**.
- Pro: computers follow their instructions **to the letter**.
- Con: computers follow their instructions **to the letter**.
- Con: computers need **detailed** instructions.

$\Rightarrow$ The **programmer** must break down the solution into unambiguous small pieces + consider all eventualities.

$\Rightarrow$ Then the **computer** will solve the problem in (usually) split-seconds.

What are you getting yourself into?

- Programming is intellectually challenging.
  - It can be tremendous fun
    if you like that sort of thing!

What are you getting yourself into?

- Programming is intellectually challenging.
  - It can be tremendous fun
    if you like that sort of thing!
- Lifelong learning is essential (*learn-2-learn*).
  - The technology is constantly changing.
  - We cannot teach you all you will ever need to know.
  - We can point you in the right direction and give you a good, hard push — but the rest is up to you!

Programming can be fun

- Programming is puzzle solving.
  - Very little is mechanical, routine work.
  - You always have to be thinking.

Programming can be fun

- Programming is puzzle solving.
  - Very little is mechanical, routine work.
  - You always have to be thinking.
- If you like solving puzzles, there's a good chance you will like programming.
  - Some puzzles are hard.
  - You need a tolerance for **frustration**.
  - Solving hard puzzles can be very satisfying.

How can I get better at programming?

- Practice, practice, practice ...
  Do as many exercises as you can (and don't look at the answers until you've had a good go at the problem).

How can I get better at programming?

- Practice, practice, practice ...
  Do as many exercises as you can (and don't look at the answers until you've had a good go at the problem).

- In this sense, getting fluent in programming is like getting fluent in a "natural" foreign language (French, Spanish, ...).

Small projects

You can build a dog kennel in a few hours:

Small projects

You can build a dog kennel in a few hours:

- You don't need a blueprint.

Small projects

You can build a dog kennel in a few hours:

- You don't need a blueprint.
- The materials don't cost much.

Small projects

You can build a dog kennel in a few hours:

- You don't need a blueprint.
- The materials don't cost much.
- A little knowledge of tools is enough.

Small projects

You can build a dog kennel in a few hours:

- You don't need a blueprint.
- The materials don't cost much.
- A little knowledge of tools is enough.
- Imperfections are no big deal.

Medium-sized projects

You can build a house in a year or so:

Medium-sized projects

You can build a house in a year or so:

- You really do need blueprints.

Medium-sized projects

You can build a house in a year or so:

- You really do need blueprints.
- Excess materials mean wasted money.

Medium-sized projects

You can build a house in a year or so:

- You really do need blueprints.
- Excess materials mean wasted money.
- House building requires more skills: plumbing, bricklaying, electrical work, carpentry, etc.

Medium-sized projects

You can build a house in a year or so:

- You really do need blueprints.
- Excess materials mean wasted money.

- House building requires more skills: plumbing, bricklaying, electrical work, carpentry, etc.
- Imperfections matter: you don't want a leaky roof!

Medium-sized projects

You can build a house in a year or so:

- You really do need blueprints.
- Excess materials mean wasted money.
- House building requires more skills: plumbing, bricklaying, electrical work, carpentry, etc.
- Imperfections matter: you don't want a leaky roof!
- It's easier if you aren't doing it all by yourself.

Large projects

You cannot build a skyscraper by yourself.

Large projects

You cannot build a skyscraper by yourself.

- It's just too much work for one person.

Large projects

You cannot build a skyscraper by yourself.

- It's just too much work for one person.
- You don't have the money.

Large projects

You cannot build a skyscraper by yourself.

- It's just too much work for one person.
- You don't have the money.
- You don't have all the skills.

Large projects

You cannot build a skyscraper by yourself.

- It's just too much work for one person.
- You don't have the money.
- You don't have all the skills.
- Imperfections could be costly or even fatal.

Large projects

You cannot build a skyscraper by yourself.

- It's just too much work for one person.
- You don't have the money.
- You don't have all the skills.
- Imperfections could be costly or even fatal.

Skyscrapers can only be built by a team.

Large projects

You cannot build a skyscraper by yourself.

- It's just too much work for one person.
- You don't have the money.
- You don't have all the skills.
- Imperfections could be costly or even fatal.

Skyscrapers can only be built by a team.

- Communication is essential.

Large projects

You cannot build a skyscraper by yourself.

- It's just too much work for one person.
- You don't have the money.
- You don't have all the skills.
- Imperfections could be costly or even fatal.

Skyscrapers can only be built by a team.

- Communication is essential.
- A "paper trail" is essential.

What does that mean for you in this course?

- What can we ask you to build in your classes?

What does that mean for you in this course?

- What can we ask you to build in your classes?
- What will be expected of you in industry?

What does that mean for you in this course?

- What can we ask you to build in your classes?
- What will be expected of you in industry?
- We teach skyscraper-level skills, but

What does that mean for you in this course?

- What can we ask you to build in your classes?
- What will be expected of you in industry?
- We teach skyscraper-level skills, but
  - we ask you to apply those skills to kennels.

What does that mean for you in this course?

- What can we ask you to build in your classes?
- What will be expected of you in industry?
- We teach skyscraper-level skills, but
  - we ask you to apply those skills to kennels.
  - It's silly, but what alternative do we have?

What does that mean for you in this course?

- What can we ask you to build in your classes?
- What will be expected of you in industry?
- We teach skyscraper-level skills, but
  - we ask you to apply those skills to kennels.
  - It's silly, but what alternative do we have?
- It's up to you: When you leave here,

What does that mean for you in this course?

- What can we ask you to build in your classes?
- What will be expected of you in industry?
- We teach skyscraper-level skills, but
  - we ask you to apply those skills to kennels.
  - It's silly, but what alternative do we have?
- It's up to you: When you leave here,
  - will you be able to contribute to skyscrapers?

What does that mean for you in this course?

- What can we ask you to build in your classes?
- What will be expected of you in industry?
- We teach skyscraper-level skills, but
  - we ask you to apply those skills to kennels.
  - It's silly, but what alternative do we have?
- It's up to you: When you leave here,
  - will you be able to contribute to skyscrapers?
  - or will you just be very good at building kennels?

What does that mean for you in this course?

- What can we ask you to build in your classes?
- What will be expected of you in industry?
- We teach skyscraper-level skills, but
  - we ask you to apply those skills to kennels.
  - It's silly, but what alternative do we have?
- It's up to you: When you leave here,
  - will you be able to contribute to skyscrapers?
  - or will you just be very good at building kennels?
  - I know what I'd prefer!

What does that mean for you in this course?

- What can we ask you to build in your classes?
- What will be expected of you in industry?
- We teach skyscraper-level skills, but
  - we ask you to apply those skills to kennels.
  - It's silly, but what alternative do we have?
- It's up to you: When you leave here,
  - will you be able to contribute to skyscrapers?
  - or will you just be very good at building kennels?
  - I know what I'd prefer!

$\Rightarrow$ We will mark coursework as if on a skyscraper-sized project!

Whom to ask if you get stuck?

If you have questions about the course (also outside of live sessions):

Whom to ask if you get stuck?

If you have questions about the course (also outside of live sessions):

- Asking your colleagues is perfectly acceptable and encouraged (not about solutions for your coursework or exam questions, of course).

Whom to ask if you get stuck?

If you have questions about the course (also outside of live sessions):

- Asking your colleagues is perfectly acceptable and encouraged (not about solutions for your coursework or exam questions, of course).
- Asking on the forums in Moodle (there is a dedicated student discussion forum on the PiJ Moodle page, which we are monitoring).

Whom to ask if you get stuck?

If you have questions about the course (also outside of live sessions):

- Asking your colleagues is perfectly acceptable and encouraged (not about solutions for your coursework or exam questions, of course).
- Asking on the forums in Moodle (there is a dedicated student discussion forum on the PiJ Moodle page, which we are monitoring).
- Online help sessions on Saturdays.

Whom to ask if you get stuck?

If you have questions about the course (also outside of live sessions):

- Asking your colleagues is perfectly acceptable and encouraged (not about solutions for your coursework or exam questions, of course).
- Asking on the forums in Moodle (there is a dedicated student discussion forum on the PiJ Moodle page, which we are monitoring).
- Online help sessions on Saturdays.
- Pasting error messages into a search engine (e.g., google) can take you a *long* way.

Whom to ask if you get stuck?

If you have questions about the course (also outside of live sessions):

- Asking your colleagues is perfectly acceptable and encouraged (not about solutions for your coursework or exam questions, of course).
- Asking on the forums in Moodle (there is a dedicated student discussion forum on the PiJ Moodle page, which we are monitoring).
- Online help sessions on Saturdays.
- Pasting error messages into a search engine (e.g., google) can take you a *long* way.
- Ask or e-mail your module tutor. :)

Rest of today's session

- Programming demo: first steps.

Rest of today's session

- Programming demo: first steps.
- Work on exercises with reading material.

Rest of today's session

- Programming demo: first steps.
- Work on exercises with reading material.
  - Opportunity for individual questions.

Rest of today's session

- Programming demo: first steps.
- Work on exercises with reading material.
  - Opportunity for individual questions.
  - If you **have** your student ID card: please swipe it on electronic register by the door.

Rest of today's session

- Programming demo: first steps.
- Work on exercises with reading material.
  - Opportunity for individual questions.
  - If you **have** your student ID card: please swipe it on electronic register by the door.
  - If you **don't have** your student ID card: please tell me your name, we will enter your attendance into the system.

Rest of today's session

- Programming demo: first steps.
- Work on exercises with reading material.
  - Opportunity for individual questions.
  - If you **have** your student ID card: please swipe it on electronic register by the door.
  - If you **don't have** your student ID card: please tell me your name, we will enter your attendance into the system.
- Programming demo.