University of Southampton

University of Southampton

COMP1202 – Building Better Programs

Computational Thinking

Son Hoang
(adapted from Prof David Millard's slides)

COMP1202 (AY2022-23)

Part 1

Organisation

Reminder: Getting Help

- Now is about the time when some people realise they are getting lost
- If this is you, Act!
  - Check you are practicing enough (6 hours per week + lectures and labs)
  - Attend Ground Controllers
  - Ask the demonstrators in the Labs
  - See Helpdesk with specific topics/questions (Discord, Email, B16 Level 2)
    - See their calendar https://secure.ecs.soton.ac.uk/student/wiki/w/Helpdesk
  - Form a study group

Building Better Programs

- To help we will spend this week consolidating some of the ideas we have encountered so far in order to answer a not-so-simple question:
  - How do you write **good** code?
- In the previous lecture, we discussed one of the fundamental principles of Object Oriented Programming: Encapsulation

Coming Up

- Introduction to Algorithms
  - Definition
  - Characteristics

- Problems to Solutions
  - The Difficulties
  - From Steps to Methods
  - Object-Oriented Solutions

Part 2

Algorithms

Definitions - Etymology

Algorism (n)

- the Arabic system of arithmetical notation (with the figures 1, 2, 3, etc.).
- the art of computation with the Arabic figures, performing arithmetic.

- Persian mathematician Abu Abdullah Muhammad ibn Musa *al-Khwarizmi* (the early 9th century)
  - Their name is Latinised as *Algorithmi*
- Europe became aware of his work on Algebra
- Arab numerals became associated with his name
- Has since evolved to mean all processes for solving tasks

Definitions - Dictionary

Algorithm (n)

"… is a finite sequence of rigorous instructions, typically used to solve a class of specific problems or to perform a computation."

Wikipedia (2022)

"A procedure or set of rules used in calculation and problem-solving; … a precisely defined set of mathematical or logical operations for the performance of a particular task."

Oxford English Dictionary (2022)

"a step-by-step procedure for solving a problem or accomplishing some end"

Merriam Webster Dictionary (2022)

Definitions - Dictionary

Algorithm (n)

"… is a finite sequence of rigorous instructions, typically used to solve a class of specific problems or to perform a computation."

Wikipedia (2022)

"A procedure or set of rules used in calculation and problem-solving; … a precisely defined set of mathematical or logical operations for the performance of a particular task."

Oxford English Dictionary (2022)

"a step-by-step procedure for solving a problem or accomplishing some end"

Merriam Webster Dictionary (2022)

Definitions - Dictionary

Algorithm (n)

"… is a finite sequence of rigorous instructions, typically used to solve a class of specific problems or to perform a computation."

Wikipedia (2022)

"A procedure or set of rules used in calculation and problem-solving; … a precisely defined set of mathematical or logical operations for the performance of a particular task."

Oxford English Dictionary (2022)

"a step-by-step procedure for solving a problem or accomplishing some end"

Merriam Webster Dictionary (2022)

Definitions - Dictionary

Algorithm (n)

"… is a finite sequence of rigorous instructions, typically used to solve a class of specific problems or to perform a computation."

Wikipedia (2022)

"A procedure or set of rules used in calculation and problem-solving; … a precisely defined set of mathematical or logical operations for the performance of a particular task."

Oxford English Dictionary (2022)

"a step-by-step procedure for solving a problem or accomplishing some end"

Merriam Webster Dictionary (2022)

Characteristics of an Algorithm

- Performance

What else?

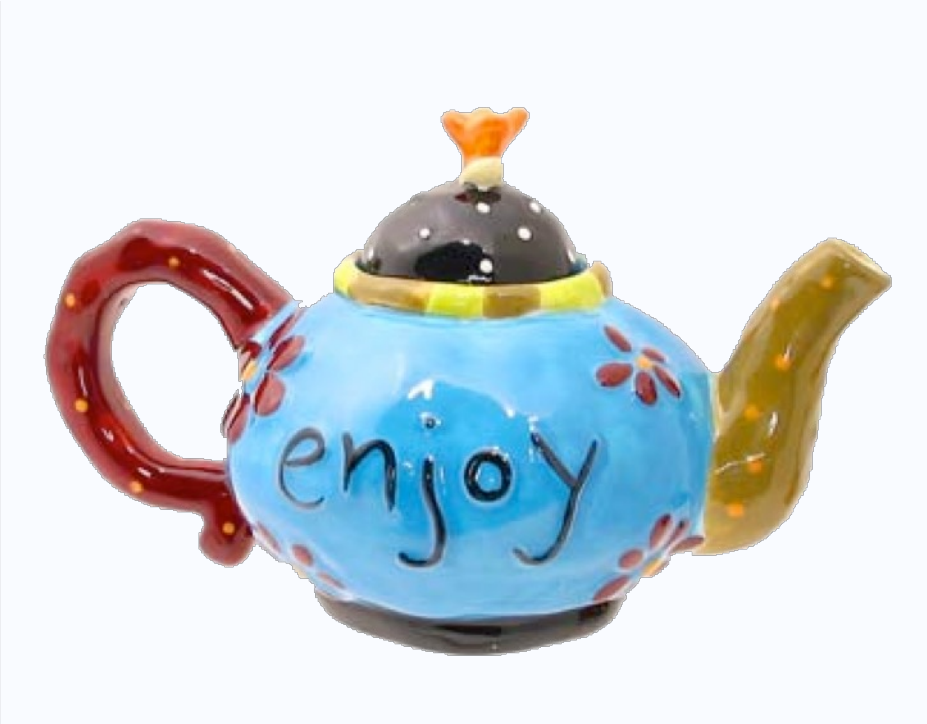
Characteristics of an Algorithm

- Performance
- Efficiency
- Understandability
- Scalability
- Reusability
- Reliability
- Elegance
  - *Elegance (n). Of scientific processes, demonstrations, inventions, etc.: 'Neatness', **ingenious simplicity,** convenience, and effectiveness - OED*

Part 3

From Problem to Solution

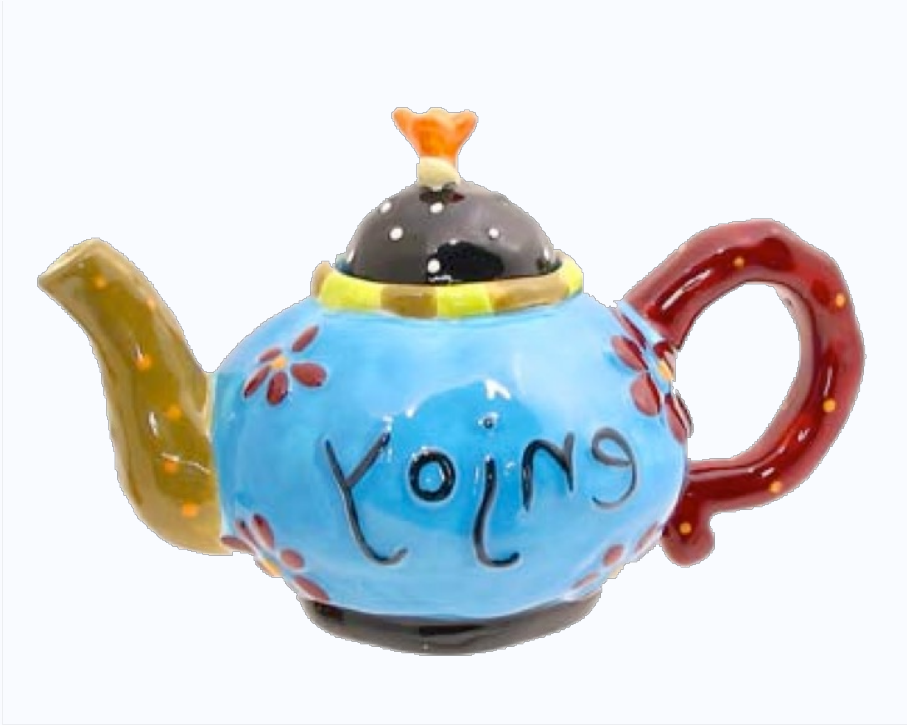
Example: Making a Cup of Tea



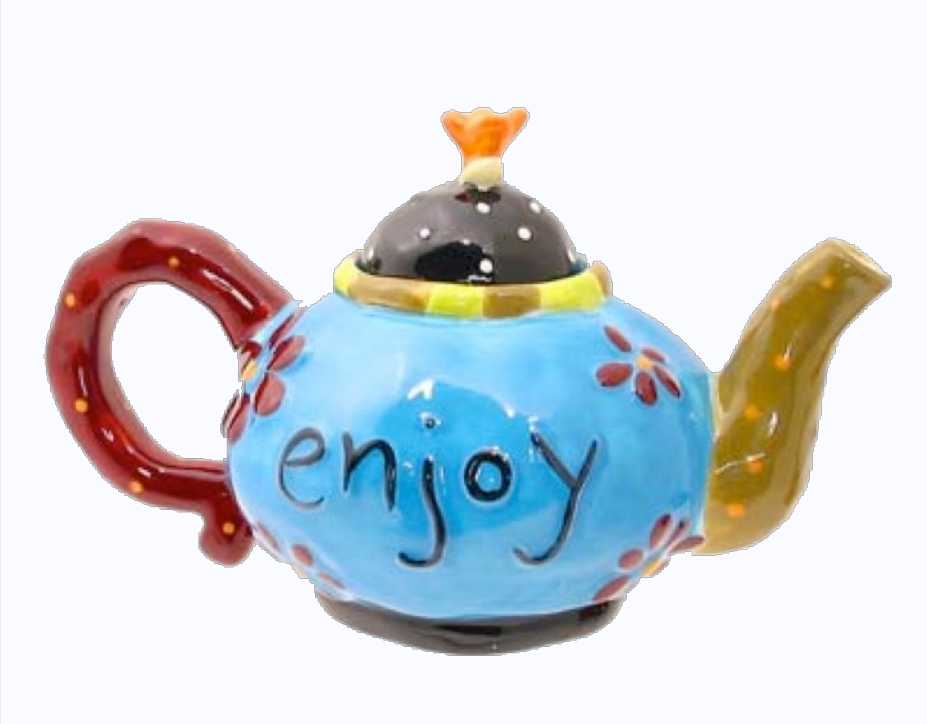
- Get into small groups of 3 or 4 and write down the steps that you need to do in order to make a cup of tea
- Use a sequence of simple statements like 'Boil Water' or 'Put Milk in Cup'
- Try and put down an appropriate level of detail so that a person could follow your instructions unambiguously
- Can you group certain steps together (for example, are the first few about preparation)? Give these groups sensible names.

Swap with Another Group

- Take a look at the tea-making instructions of another group
- Annotate their instructions if:
  - Any instruction is ambiguous
  - They have missed something out
  - They have made an assumption
  - They have created a group that doesn't make sense to you
- Also if they have done anything you really like :-)



Go Back to Your Own Instructions



- Take a few moments to see what they have said
  - Are the comments fair?
  - Any unexpected ones?
  - Would you make any changes now you have seen someone else's instructions?

Writing Sequences is Easy…

… But getting the sequence right is hard

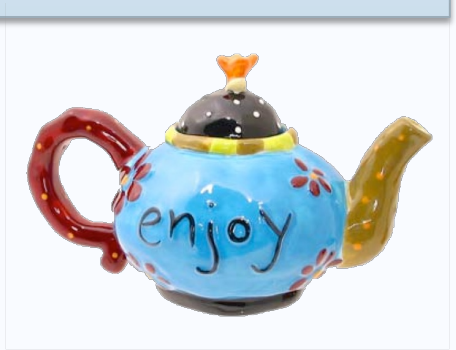
- Often the specification is inadequate
  - It is easy to make assumptions without realising it
- Making it complete is challenging
  - Making sure not to miss smaller, less-obvious steps
  - Creating unambiguous instructions
- Machines are very unforgiving, they do exactly what you ask – nothing more, nothing less

Example. Making a Cup of Tea

```
Make a Cup of Tea
Get Cup
Get Kettle
Get Tea
Get Milk
Get Sugar Lumps
Empty Kettle
Fill Kettle with Water
Switch Kettle on
Wait until Kettle Boils
Put Tea in Pot
Put Boiling Water in Pot
Wait 2 Minutes
Put Milk in Cup
Pour Tea in Cup
Put 1 Sugar Lump in Cup
Stir Tea in Cup
Give Cup of Tea to User
```

Pseudocode

Assumptions?



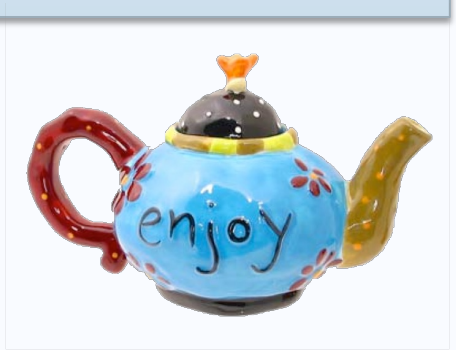
Example. Making a Cup of Tea

```
Make a Cup of Tea
Get Cup
Get Kettle
Get Tea
Get Milk
Get Sugar Lumps
Empty Kettle
Fill Kettle with Water
Switch Kettle on
Wait until Kettle Boils
Put Tea in Pot
Put Boiling Water in Pot
Wait 2 Minutes
Put Milk in Cup
Pour Tea in Cup
Put 1 Sugar Lump in Cup
Stir Tea in Cup
Give Cup of Tea to User
```

Pseudocode

Assumptions?

- Electric Kettle
- Users want Milk and Sugar
- Only making 1 cup of tea
- Nothing goes wrong!

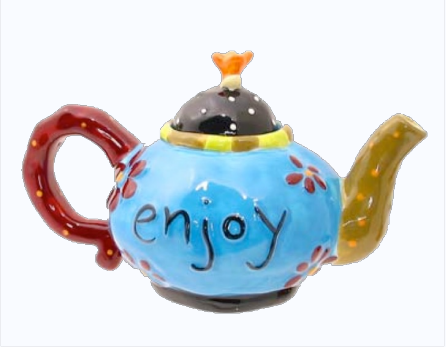


Modules

- Modules break an algorithm into logical parts (like your groups)
  - Helps with Clarity and Understandability
- Modules can be reused
  - Within the same algorithm
  - In a different algorithm
- In Programming Modules can be called:
  - Sub-routines (in older languages)
  - Functions (in procedural languages like C)
  - Methods (in object-oriented languages like Java)

Example. Making a Cup of Tea

```
Make a Cup of Tea
Get Cup
Get Kettle
Get Tea
Get Milk
Get Sugar Lumps
Empty Kettle
Fill Kettle with Water
Switch Kettle on
Wait until Kettle Boils
Put Tea in Pot
Put Boiling Water in Pot
Wait 2 Minutes
Put Milk in Cup
Pour Tea in Cup
Put 1 Sugar Lump in Cup
Stir Tea in Cup
Give Cup of Tea to User
```



Example. Making a Cup of Tea

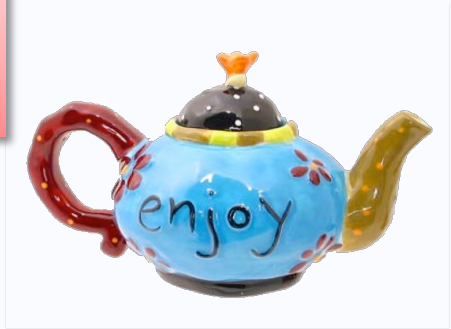
```
Make a Cup of Tea
Get Cup
Get Kettle
Get Tea
Get Milk
Get Sugar Lumps
Empty Kettle
Fill Kettle with Water
Switch Kettle on
Wait until Kettle Boils
Put Tea in Pot
Put Boiling Water in Pot
Wait 2 Minutes
Put Milk in Cup
Pour Tea in Cup
Put 1 Sugar Lump in Cup
Stir Tea in Cup
Give Cup of Tea to User
```

Fetch Utensils and Ingredients

Boil Water in Kettle

Make Tea in Pot

Add Tea, Milk and Sugar to Cup



Example. Making a Cup of Tea

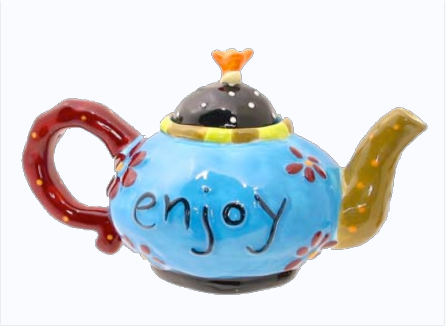
Make a Cup of Tea
Fetch Utensils and Ingredients
Boil Water in Kettle
Make Tea in Pot
Add Tea, Milk and Sugar to Cup
Give Cup of Tea to User

Fetch Utensils and Ingredients
Get Cup
Get Kettle
Get Tea
Get Milk
Get Sugar Lumps

Boil Water in Kettle
Empty Kettle
Fill Kettle with Water
Switch Kettle on
Wait until Kettle Boils

Make Tea in Pot
Put Tea in Pot
Put Boiling Water in Pot
Wait 2 Minutes

Add Tea, Milk and Sugar to Cup
Put Milk in Cup
Pour Tea in Cup
Put 1 Sugar Lump in Cup
Stir Tea in Cup
Give Cup of Tea to User



A Procedural Approach?
What about Object Oriented Solutions?

Make a Cup of Tea
Fetch Utensils and Ingredients
Boil Water in Kettle
Make Tea in Pot
Add Tea, Milk and Sugar to Cup
Give Cup of Tea to User

Fetch Utensils and Ingredients
Get Cup
Get Kettle
Get Tea
Get Milk
Get Sugar Lumps

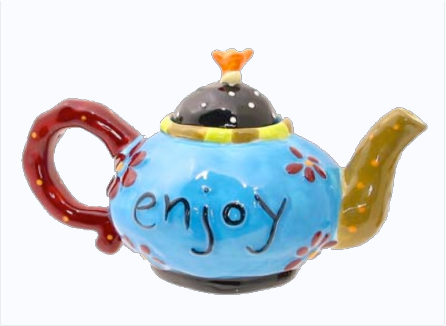
Boil Water in Kettle
Empty Kettle
Fill Kettle with Water
Switch Kettle on
Wait until Kettle Boils

Make Tea in Pot
Put Tea in Pot
Put Boiling Water in Pot
Wait 2 Minutes

Add Tea, Milk and Sugar to Cup
Put Milk in Cup
Pour Tea in Cup
Put 1 Sugar Lump in Cup
Stir Tea in Cup
Give Cup of Tea to User

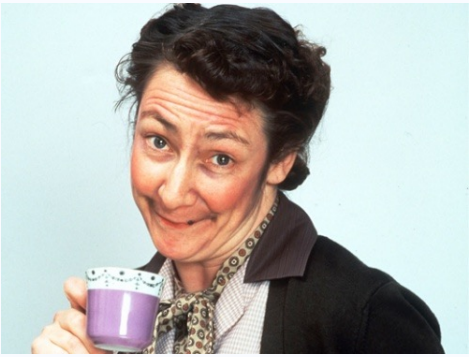
What are the objects?

Remember:
objects should have behaviour and/or bring together a particular set of data



A Procedural Approach?
What about Object Oriented Solutions?

Make a Cup of Tea
Fetch Utensils and Ingredients
Boil Water in Kettle
Make Tea in Pot
Add Tea, Milk and Sugar to Cup
Give Cup of Tea to User



TeaMaker?

Fetch Utensils and Ingredients
Get Cup
Get Kettle
Get Tea
Get Milk
Get Sugar Lumps

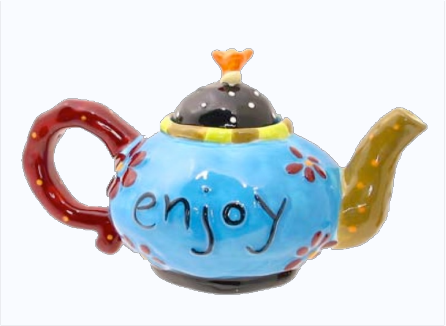
Boil Water in Kettle
Empty Kettle
Fill Kettle with Water
Switch Kettle on
Wait until Kettle Boils

Make Tea in Pot
Put Tea in Pot
Put Boiling Water in Pot
Wait 2 Minutes

Add Tea, Milk and Sugar to Cup
Put Milk in Cup
Pour Tea in Cup
Put 1 Sugar Lump in Cup
Stir Tea in Cup
Give Cup of Tea to User

What are the objects?

Remember:
objects should have behaviour and/or bring together a particular set of data



A Procedural Approach?
What about Object Oriented Solutions?

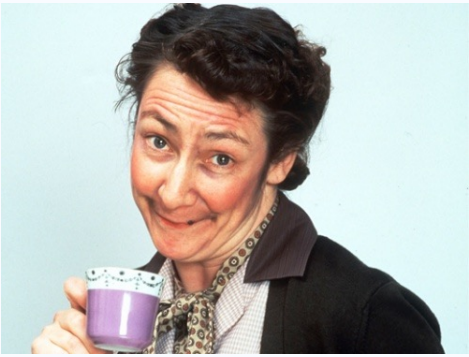
Warning: Kettle, Teapot and Cup classes need to be defined, and some other methods are needed in TeaMaker

Make a Cup of Tea
Fetch Utensils and Ingredients
Boil Water in Kettle
Make Tea in Pot
Add Tea, Milk and Sugar to Cup
Give Cup of Tea to User

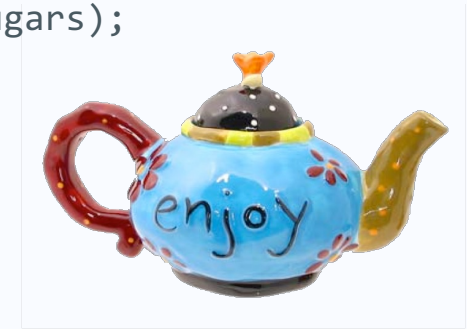
```
public class TeaMaker {
  Kettle kettle = new Kettle();
  Teapot pot = new Teapot();

  public static void main(String[] args){
    TeaMaker maker = new TeaMaker();
    Cup cup = maker.makeTea(true, 1);
  }

  public Cup makeTea(boolean milk, int sugars){
    if (haveIngredients(milk, sugars)) {
      kettle.boilWater();
      pot.addTeaBags();
      kettle.pourWaterInto(pot);
      Cup cup = prepareCup(milk, sugars);
      pot.pourTeaInto(cup);
      return cup;
    }
  }
}
```



TeaMaker?



Part 4

A Note on Coding Style

Naming

- Data and Methods should have **meaningful** names
  - Classes – UpperCamelCase
  - Variables/Methods/Parameters – lowerCamelCase
  - Constants – UPPERCASE
- Java is a verbose language - Embrace it!
- Don't be afraid to be explicit (within reason!)
  - avMk
  - averageMark
  - averageMarkOfAnIndividualStudentInProgramming1Cohort

Layout

- Code should be indented so that the structure is clear
- Use brackets to be explicit about grouping statements

```
public static void main(String[] args) {
  Account myAccountObject = new Account();
  myAccountObject.withdraw(5);
  myAccountObject.withdraw(10);
}
```

vs.

```
public static void main(String[] args)
{Account myAccountObject = new Account();
myAccountObject.withdraw(5);
myAccountObject.withdraw(10);}
```

Comments

- Comments should explain appropriate **blocks of code**
  - Every method / class
  - Every logical section of a method
- There is no need to explain the obvious!

```
// add 100 to the variable i
int i = i + 100;
```

- Java has an advanced comment system called JavaDoc
  - Compiles comments into documentation
  - The Java API is entirely generated by JavaDoc

Style Guide

- Google has a good style guide:
  - https://google.github.io/styleguide/javaguide.html
- Note that everyone has minor differences (quirks) in the way they program
- Most important to be **consistent**, and to be **clear**
- Labs feedback contains "warnings" about code style (no marks deducted).
- **We will ask you to follow Google Java Style in the coursework**

(check out the **Obfuscated Code Contest** – it is worth a quick Google – and is everything that you should NOT do!)

YOUR QUESTIONS

- Introduction to Algorithms
  - Definition
  - Characteristics
- Problems to Solutions
  - The Difficulties
  - From Steps to Methods
- Java Coding Style