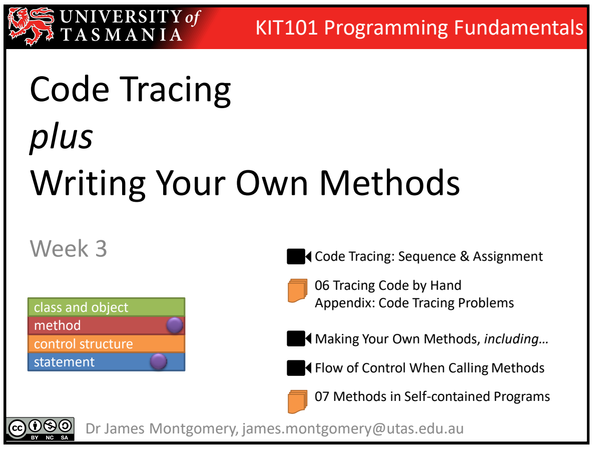


UNIVERSITY *of* TASMANIA

KIT101 Programming Fundamentals

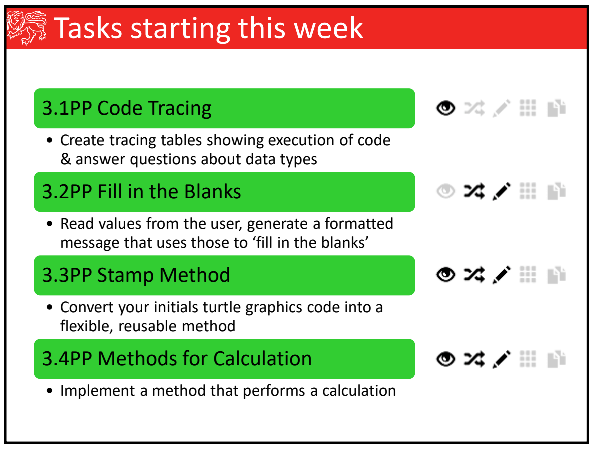
Code Tracing *plus* Writing Your Own Methods

Week 3

class and object
method ●
control structure
statement ●

- Code Tracing: Sequence & Assignment
- 06 Tracing Code by Hand
  Appendix: Code Tracing Problems
- Making Your Own Methods, *including…*
- Flow of Control When Calling Methods
- 07 Methods in Self-contained Programs

Dr James Montgomery, james.montgomery@utas.edu.au



Tasks starting this week

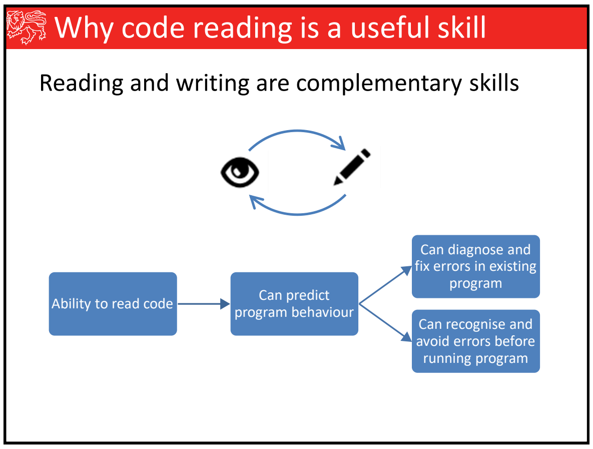
3.1PP Code Tracing
- Create tracing tables showing execution of code & answer questions about data types

3.2PP Fill in the Blanks
- Read values from the user, generate a formatted message that uses those to 'fill in the blanks'

3.3PP Stamp Method
- Convert your initials turtle graphics code into a flexible, reusable method

3.4PP Methods for Calculation
- Implement a method that performs a calculation

Largest # of Tasks this week



Why code reading is a useful skill

Reading and writing are complementary skills

$$\begin{tikzcd} \text{Ability to read code} \arrow[r] & \text{Can predict program behaviour} \arrow[r] \arrow[dr] & \text{Can diagnose and fix errors in existing program} \\ & & \text{Can recognise and avoid errors before running program} \end{tikzcd}$$

Tracing: essential tools

Paper & pen (or text editor)

Knowledge of the programming language's semantics

- ✓ A mental model of computer memory (we will use a table holding values)
- ✓ The effect of assignment statements
- ✓ The order in which parts of a statement are executed by the computer
- ✓ The behaviour of methods that are called

If unsure about the effect of a statement: ask for help, consult documentation or write a small program to find out

Demonstrations & Activities

Let's trace some code
- including declaration, assignment & simple expressions

You trace some code
- and we'll check it together

Implementing an action-oriented method
- making a sequence of actions easily reusable

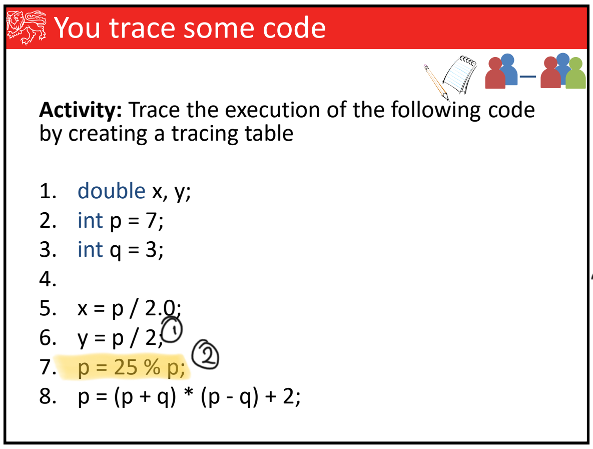
Implementing a function
- making a calculation easily reusable

Download accompanying sample code from MyLO

Demonstration outcome

```
1.  int a, b, c;
2.
3.  a = 5;
4.  b = 12;
5.  System.out.println("b was " + b);
6.  c = b / a;
7.  b = c + a;
8.  System.out.println("b is now " + b);
```

Line	a	b	c	Output
3	5			
4		12		
5				b was 12
6			2	
7		7		
8				b is now 7



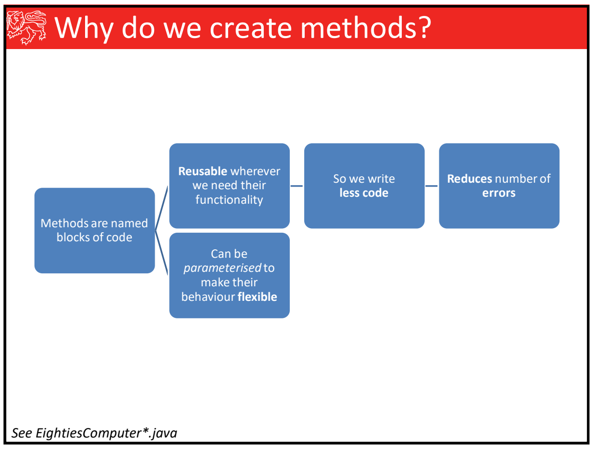
You trace some code

Activity: Trace the execution of the following code by creating a tracing table

```
1.  double x, y;
2.  int p = 7;
3.  int q = 3;
4.
5.  x = p / 2.0;
6.  y = p / 2; ①
7.  p = 25 % p; ②
8.  p = (p + q) * (p - q) + 2;
```

① NOTE: INTERGER divided by an INTERGER IS AN INTERGER

② % = REMAINDER or Modulo
(Look up IN RECORDING)



Why do we create methods?

Methods are named blocks of code

Reusable wherever we need their functionality

So we write **less code**

Reduces number of **errors**

Can be *parameterised* to make their behaviour **flexible**

See EightiesComputer.java*

* ALSO CALLED FUNCTIONS or PROCEDURES IN other languages