

COMP2511

23T2 Week 1

Wednesday 1PM - 4PM (W13A)

Thursday 3PM - 6PM (H15B)

Friday 2PM - 5PM (F14B)

Slides by Alvin Cherk (z5311001)

Today

- General Tutorial/Lab Introduction
- Java Syntax
- Object-Oriented Paradigm

Introduction

- My name is Alvin
- 4th Year Computer Science / Mechatronics student
- I like mechanical keyboards, Destiny 2, Gundams.
- ~~Full Stack Developer~~
- Email: a.cherk@unsw.edu.au (Please try using the **forums** before emailing me, **unless its for a personal reason**)
 - Include your **zID** somewhere so I can easily identify you
 - I will usually take **24-48** hours at most to reply. If you don't get a reply, then send a follow up email.
 - Prefix your email's title with 2511-[CLASSNAME] (e.g., 2511-F10A)
- Course email: cs2511@cse.unsw.edu.au
 - Don't like how something is handled? You're welcome to escalate it to course admins. (Do talk to me first if possible).

How will it work?

- 1 Hour tutorial, 2 hour lab
 - Tutorial: Tutorial questions that go over recent lecture topics
 - Lab: Lab exercises & marking, general help, assignment check-ins/marking (later).
- Slides: <https://slides.com/kuroson/decks/2511-23t2>
- Repository: <https://github.com/Kuroson/comp2511-23T2-tutorial>
- Coursework: 15% (from your Course Outline)
 - Lab exercises (Must be marked **within 2 weeks of it being due**)
 - I.e., you cannot get lab01 marked off in week 10

My Suggestions

- Get to know the people in your tutorial. Make friends. You will be working in a pair for assignment 2/3 (optional for assignment 3).
- Make sure you are keeping up with lecture content
- Read your course outline. The course has changed!
- Plan out your term, COMP2511 can take up a lot of time!
- Please start assignments/projects early! (Looking at assignment 1)
- Ask lots of questions
- Prepare to learn how to read documentation & google on your own

Ice Breaker

- Name (and preferred name)
 - Degree
 - Favourite language & why
- Or
- Random interesting fact

Git

Git Revision

- `git add`
 - Stage files
- `git commit`
 - Commit the staged files as a snapshot
- `git push`
 - Push your new commits to an online origin (GitLab, BitBucket, GitHub)
- `git status`
 - State of current repository & branch
- `git log`
 - History of current branch

Differences between Java, C, Python, JS/TS

- Syntax
 - C, Java, JS/TS use { and } to describe code blocks (also scopes)
 - Python uses whitespaces (tabs/indentations)
- Classes:
 - Java, Python, JS/TS support Object Oriented programming (OOP)
 - Supports classes and inheritance
 - C does not support classes. Closest things are pure 'data classes' called structs
 - All code within Java needs to exist within a class
 - JS/TS and Python can have runnable code outside classes

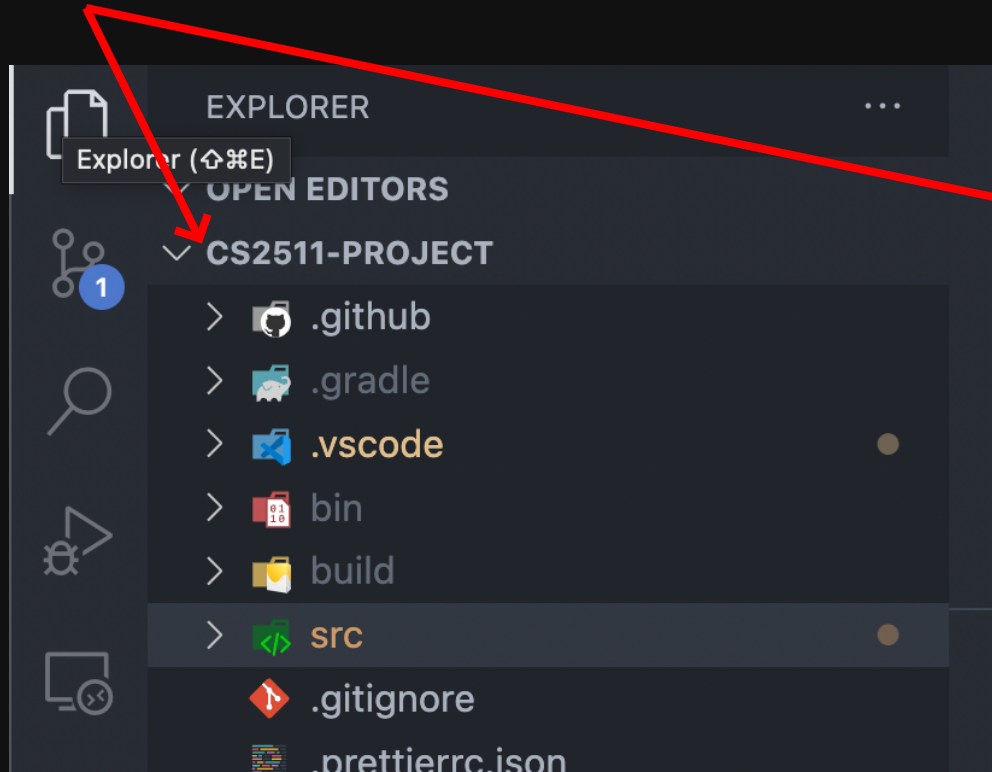
Differences between Java, C, Python

- Types:
 - Java, C and TypeScript are statically typed
 - Python and JavaScript are dynamically typed
- Memory:
 - C allows you to manually allocate memory
 - Java, Python, JS/TS have automatic memory management
- Compilation
 - C compiles into machine code
 - Java and Python compile into byte code, which is interpreted
 - TypeScript transpiles to JavaScript which is then interpreted

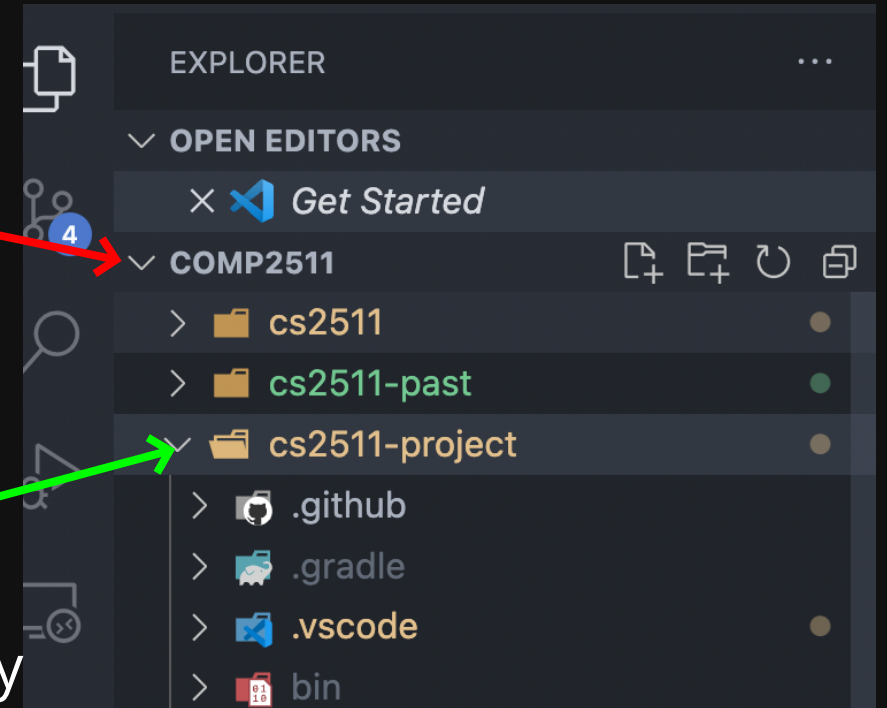
VSCode Config Tips

Ensure that you open the correct folder
example: cs2511-project

Name of folder open



Good



The folder I
want to actually
open

Bad

VSCode Config Tips

Ensure you have the correct Java extension installed

The screenshot shows the VS Code interface with the Extensions view open. The left sidebar lists several Java-related extensions. The main panel displays the details for the 'Extension Pack for Java' by Microsoft, which is currently installed. The details panel shows the extension's icon, name, version (v0.23.0), publisher (Microsoft), download count (13,120,043), and rating (4.5 stars from 49 reviews). It also includes buttons for 'Disable', 'Uninstall', and 'Switch to Pre-Release Version'. Below the main details, there are tabs for 'Details', 'Feature Contributions', 'Changelog', and 'Runtime Status'. The 'Details' tab is active, showing a list of extensions included in the pack: 'Language Support for Java(TM)' by Red Hat, 'Debugger for Java' by Microsoft, 'Spring Initializr' by Microsoft, 'Java Language Support' by George Fraser, and 'Java Debugger' by Microsoft. The bottom of the details panel shows the title 'Extension Pack for Java' and a brief description: 'Extension Pack for Java is a collection of popular extensions that can help write, test and debug Java'.

EXTENSIONS: MARKETPLACE

java

- Extension Pack for J...** 45ms
Popular extensions for Java d...
Microsoft
- Maven for Java**
Manage Maven projects, exec...
Microsoft
- Debugger for Java** 9ms
A lightweight Java debugger f...
Microsoft
- Project Manager for ...** 12ms
Manage Java projects in Visua...
Microsoft
- Test Runner for Java** 42ms
Run and debug JUnit or TestN...
Microsoft
- Language Support fo...** 67ms
Java Linting, Intellisense, form...
Red Hat
- Spring Initializr...** 1.5M ★ 3.5
A lightweight extension based...
Microsoft
- Java Language Su...** 1M ★ 3
Java support using the Java C...
George Fraser
- Java Debugger** 533K ★ 3.5

Extension: Extension Pack for Java — cs2511-22t2-private

Extension Pack for Java v0.23.0 **Preview**

Microsoft | 13,120,043 | ★★★★★ (49)

Popular extensions for Java development that provides Java IntelliSense, de

Disable Uninstall Switch to Pre-Release Version

This extension is enabled globally.

Details Feature Contributions Changelog Runtime Status

Extension Pack (6)

- Language Support for Java(TM) ...** 67ms
Java Linting, Intellisense, formatting, refac...
Red Hat
- Debugger for Java** 9ms
A lightweight Java debugger for Visual Stu...
Microsoft

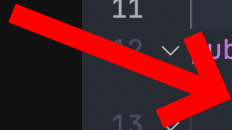
Extension Pack for Java

Extension Pack for Java is a collection of popular extensions that can help write, test and debug Java

VSCode Config Tips

Ensure that you are running code using the "Run" button

Always use this to
run code in this
course



```
You, last week | 1 author (You)
1 package example;
2
3 import java.util.Arrays;
4 import java.util.Scanner;
5
You, last week | 1 author (You)
6 /**
7  * Write a program that uses the `Scanner` class
8  * which reads in a line of numbers separated by spaces,
9  * and sums them.
10 */
11
You, last week • Week 1 ...
12 public class Sum {
13     public static void main(String[] args) {
14         /**
15          * new - Creates a new Scanner object. (Think of it like C Malloc, but Java's
16          * garbage collection frees it)
17          * Scanner is an object that allows us to specify an input
18          * System.in = stdin in C
19          */
20         Scanner scanner = new Scanner(System.in);
21     }
```

Code Demo

HelloWorld.java

Code Demo

Make a simple Java program that prints "Hello World" in the **HelloWorld** class.

Code Demo

```
1 package example;
2
3 /**
4  * Prints "Hello World" to the console.
5  */
6 public class HelloWorld {
7     public static void main(String[] args) {
8         // Does it need a \n?
9         // No, .println appends a \n to your string when it prints
10        System.out.println("Hello World");
11    }
12 }
```


Code Demo

Sum.java

Code Demo

Inside a new file called **Sum.java**, write a program that uses the **Scanner** class which reads in a line of numbers separated by spaces, and sums them.

```

1 package example;
2
3 import java.util.Arrays;
4 import java.util.Scanner;
5
6 /**
7  * Write a program that uses the `Scanner` class
8  * which reads in a line of numbers separated by spaces,
9  * and sums them.
10 */
11
12 public class Sum {
13     public static void main(String[] args) {
14         /**
15          * new - Creates a new Scanner object. (Think of it like C Malloc, but Java's
16          * garbage collection frees it)
17          * Scanner is an object that allows us to specify an input
18          * System.in == stdin in C
19          */
20         Scanner scanner = new Scanner(System.in);
21
22         /**
23          * Keeps reading input until it sees a \n
24          *
25          * Splits each string into an array of strings
26          */
27         String[] numbers = scanner.nextLine().split(" ");
28
29         int sum = 0;
30         for (String number : numbers) {
31             sum += Integer.parseInt(number);
32         }
33         System.out.println("The sum is " + sum);
34
35         // Advanced
36         // Using streams
37         int streamSum = Arrays.asList(numbers).stream().mapToInt(x -> Integer.parseInt(x)).sum();
38         System.out.println(String.format("The sum is %d", streamSum));
39
40         /**
41          * Frees I/O resources
42          * Java's garbage collector only manages memory, not other resources
43          */
44         scanner.close();
45     }
46 }

```

Code Demo

Loop.java

Code Demo

```
1 public class LoopExample {
2     public static void main(String[] args) {
3         String[] myStrings = { "Hello", "World", "No" };
4
5         // Index based looping
6         for (int i = 0; i < myStrings.length; i++) {
7             String current = myStrings[i];
8             System.out.println(current);
9         }
10
11        // For-range / for-in loop
12        for (String current : myStrings) { // Very python like
13            System.out.println(current);
14        }
15    }
16 }
```

- Use the for-range loop unless you need access to index control
- You can use a index based loop if you need to change the value while looping over a collection of items
- **Style marks will be lost in assignments** if index loops are used when for-range loops could have been used

Why Classes?

Why do we use classes?

Ignoring OOP, Inheritance, Polymorphism, classes are useful for:

- Grouping "global variables" together
- Group relevant functions together
- Scoping of variables/functions
- Data hiding

Code Demo

Shouter.java

Code Demo

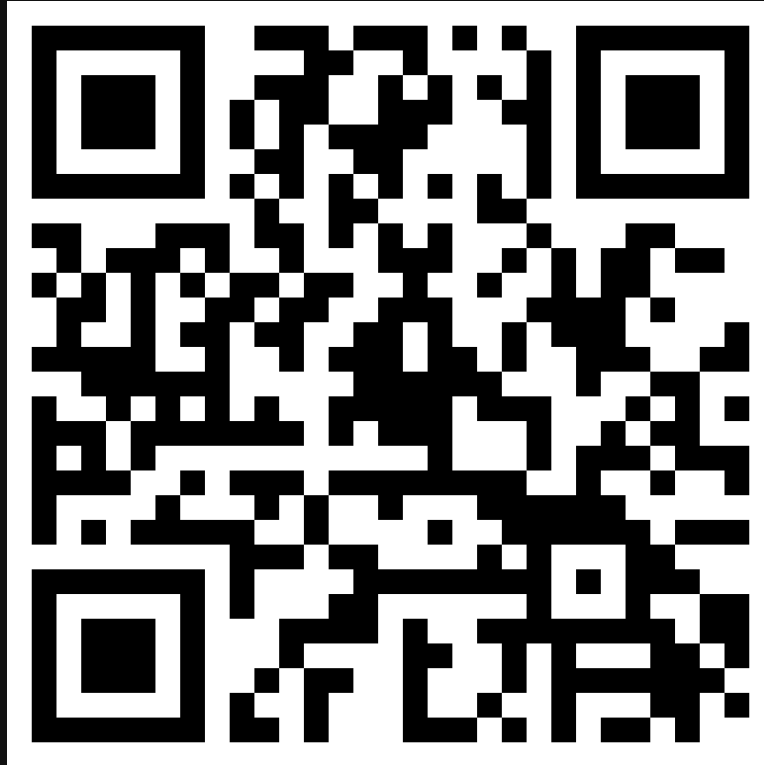
Inside a new file **Shouter.java**, Write a program that stores a message and has methods for getting the message, updating the message and printing it out in all caps. Write a **main()** method for testing this class.


```

1 package example;
2
3 public class Shouter {
4     private String message;
5
6     public Shouter(String message) {
7         this.message = message;
8     }
9
10    public String getMessage() {
11        // NOTE: You don't have to use the keyword `this`
12        // But I use it because of clarity
13        return this.message;
14    }
15
16    public void setMessage(String newMessage) {
17        this.message = newMessage;
18    }
19
20    public String toString() {
21        return String.format("Shouter message = %s", this.message);
22    }
23
24    public void printMe() {
25        System.out.println(this.message);
26    }
27
28    public void shout() {
29        System.out.println(this.message.toUpperCase());
30    }
31
32    public void printAndShout() {
33        // NOTE: You don't have to use the keyword `this`
34        // But I use it because of clarity
35        this.printMe();
36        this.shout();
37    }
38
39    public static void main(String[] args) {
40        Shouter s = new Shouter("This is my message");
41        s.printMe();
42        s.shout();
43        // When printing objects, Java will try and stringify
44        // In this case, it calls the .toString() method
45        System.out.println(s);
46    }
47 }

```

Feedback



<https://forms.gle/R4sMTTQzPC4vqXSN8>