COMP2511

23T1 Week 1

WEDNESDAY 1PM - 4PM (W13B)

FRIDAY 11AM - 2PM (F11A)

Slides by Alvin Cherk (z5311001)

Today

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

Introduction

- My name is Alvin
- 4rd Year Computer Science / Mechatronics student
- I like mechanical keyboards, Genshin Impact, Gundams, Valorant.
- 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/2511-23t1
- Repository: https://github.com/Kuroson/comp2511-23T1-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 compiles into byte code, which is interpreted
  - TypeScript transpiles to JavaScript which is then interpreted

VSCode Setup

If you are using VSCode on CSE machines (VLAB or in-person), please ensure you are using **2511 code**

```
1  cvx02 % code --version
2  1.67.2
3  c3511e6c69bb39013c4a4b7b9566ec1ca73fc4d5
4  x64
5  vx02 % 2511 code --version
6  1.74.3
7  97dec172d3256f8ca4bfb2143f3f76b503ca0534
8  x64
```

If you don't some Java Extensions (notably the test runners) will not work properly.

See the **Setup Troubleshooting** Confluence page if you have encounter any issues

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.

```java
 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

```java
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

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.

```java
package example;

public class Shouter {
    private String message;

    public Shouter(String message) {
        this.message = message;
    }

    public String getMessage() {
        // NOTE: You don't have to use the keyword `this`
        // But I use it because of clarity
        return this.message;
    }

    public void setMessage(String newMessage) {
        this.message = newMessage;
    }

    public String toString() {
        return String.format("Shouter message = %s", this.message);
    }

    public void printMe() {
        System.out.println(this.message);
    }

    public void shout() {
        System.out.println(this.message.toUpperCase());
    }

    public void printAndShout() {
        // NOTE: You don't have to use the keyword `this`
        // But I use it because of clarity
        this.printMe();
        this.shout();
    }

    public static void main(String[] args) {
        Shouter s = new Shouter("This is my message");
        s.printMe();
        s.shout();
        // When printing objects, Java will try and stringify
        // In this case, it calls the .toString() method
        System.out.println(s);
    }
}
```

Feedback

https://forms.gle/R4sMTTQzPC4vqXSN8