COMP2511

Week 1

TUESDAY 1PM - 4PM (T13B)

Introduction

- My name is Alvin
- 3rd Year Computer Science / Mechatronics student
- I like mechanical keyboards and Lost Ark
- Full stack developer (aka professional googler)

- Email: a.cherk@student.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.
  - Try not to message me on teams
- Course email: cs2511@cse.unsw.edu.au

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 (later).
- Slides: https://slides.com/kuroson/decks/2511-22t3
- Repository: https://github.com/Kuroson/comp2511-T13B-22T3

- Coursework: 15% (from your Course Outline)
  - Tutorial attendance + participation (2 marks)
    - Attend tutorial (& get name marked off) (1 mark)
    - Ask or answer questions (1 mark)
    - Sick? Need to miss tutorial? => **Email me** with a short explanation
  - Lab exercises (6 marks)

My Suggestions

- Get to know the people in your tutorial. Make friends. You will be working in a pair for assignment 2/3.
- 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
- 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

- Syntax
  - C and Java use { and } to describe code blocks (also scopes)
  - Python uses whitespaces (tabs/indentations)

- Classes:
  - Java and Python 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

Differences between Java, C, Python

- Types:
  - Java and C are both statically typed
  - Python is dynamically typed
- Memory:
  - C allows you to manually allocate memory
  - Java and Python have automatic memory management
- Compilation
  - C compiles into machine code
  - Java and Python compiles into byte code, which is interpreted

Code Demo

HelloWorld.java

Code Demo

```java
package example;

/**
 * Prints "Hello World" to the console.
 */
public class HelloWorld {
    public static void main(String[] args) {
        // Does it need a \n?
        // No, .println appends a \n to your string when it prints
        System.out.println("Hello World");
    }
}
```

Code Demo

Loop.java

Code Demo

```java
public class LoopExample {
    public static void main(String[] args) {
        String[] myStrings = { "Hello", "World", "No" };

        // Index based looping
        for (int i = 0; i < myStrings.length; i++) {
            String current = myStrings[i];
            System.out.println(current);
        }

        // For-range / for-in loop
        for (String current : myStrings) { // Very python like
            System.out.println(current);
        }
    }
}
```

- 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

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
package example;

import java.util.Arrays;
import java.util.Scanner;

/**
 * Write a program that uses the `Scanner` class
 * which reads in a line of numbers separated by spaces,
 * and sums them.
 */

public class Sum {
    public static void main(String[] args) {
        /**
         * new - Creates a new Scanner object. (Think of it like C Malloc, but Java's
         * garbage collection frees it)
         * Scanner is an object that allows us to specify an input
         * System.in == stdin in C
         */
        Scanner scanner = new Scanner(System.in);

        /**
         * Keeps reading input until it sees a \n
         *
         * Splits each string into an array of strings
         */
        String[] numbers = scanner.nextLine().split(" ");

        int sum = 0;
        for (String number : numbers) {
            sum += Integer.parseInt(number);
        }
        System.out.println("The sum is " + sum);

        // Advanced
        // Using streams
        int streamSum = Arrays.asList(numbers).stream().mapToInt(x -> Integer.parseInt(x)).sum();
        System.out.println(String.format("The sum is %d", streamSum));

        /**
         * Frees I/O resources
         * Java's garbage collector only manages memory, not other resources
         */
        scanner.close();
    }
}
```

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);
    }
}
```

Attendance

Feedback

https://forms.gle/R4sMTTQzPC4vqXSN8