COMP2511

Week 1

THURSDAY 9AM - 12PM (H09A)
FRIDAY 10AM - 1PM (F10A)

Introduction

- My name is Alvin
- 3rd Year Computer Science / Mechatronics student
- I am interested in mechanical keyboards and Lost Ark / New World

- Email: a.cherk@student.unsw.edu.au (Try to use the forums before emailing me, unless its for a personal reason)
  - Include **2511-H09A** or **2511-F10A** in the title, and your **zID** somewhere in the message
  - Usually take **24-48** hours at most to reply. If you don't get a reply, send a follow up email

How does it work?

- 1 Hour tutorial, 2 hour lab
  - Tutorial will be mostly going over tutorial questions that cover recent lecture content
  - Lab time will be used for lab marking, lab exercises help, group assignment meetings (later)

- Repository: See Teams for link

- Class Mark: 15% (from your Course Outline)
  - Tutorial attendance + participation (2 marks)
    - Attend full tutorial (Make sure you get your name marked off by me)
    - Ask or answer questions
    - If you do miss a in-person tutorial, please email me with a short explanation
  - Lab exercises (6 marks)

My Suggestions

- Get to know the people around you. Make friends. They will be the people you will be working with soon. (Group Project)
  - You will need to form groups of 4-5 by the end of week 2
- Make sure you are keeping up with lecture content.
- Plan out your term, COMP2511 can take up a lot of time!
- Start assignments/projects early!
- Ask lots of questions & learn to effectively Google & read documentation

Ice Breaker

- Name (and preferred name if necessary)
- Degree
- Interesting fact

Git Revision

- git add
- git commit
- git push
- git status
- git log

Git Revision

- git add – stage files
- git commit
- git push
- git status
- git log

Git Revision

- git add – stage files
- git commit – save staged files as a snapshot
- git push
- git status
- git log

Git Revision

- git add – stage files
- git commit – save staged files as a snapshot
- git push – push your new commits to an online git repository
- git status
- git log

Git Revision

- git add – stage files
- git commit – save staged files as a snapshot
- git push – push your new commits to an online git repository
- git status – state of current branch
- git log

Git Revision

- git add – stage files
- git commit – save staged files as a snapshot
- git push – push your new commits to an online git repository
- git status – state of current branch
- git log – history of current branch

Differences between Java, C, Python

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

- Classes:
  - Java and Python are Object Oriented (OO)
    - Supports classes and inheritance
  - C does not
  - All code within Java needs to be in 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

Hello World

Hello World

```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");
    }
}
```

for in / for each

```
1  String[] myStrings = { "Hello", "World", "No" };
2
3  for (int i = 0; i < myStrings.length; i++) {
4          String current = myStrings[i];
5          System.out.println(current);
6  }
7
8  for (String current : myStrings) {
9          system.out.println(current);
10 }
```

Code Demo

Sum.java

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

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;

/**
 * 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.
 *
 * Create a, Getter, Setter, Constructor, printMe, shout, toString
 */
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/fZDe2zhbo52UNnwh7