CS2030 Lab
B12A

TABLE OF CONTENTS

Introduction

ABOUT US

Daniel Ong

Year 2 Information Systems

Email: daniel.ong.ee.shaeon@u.nus.edu

Bryan Teo

Year 2 Business Analytics

Email: bryanteo@u.nus.edu

Telegram Group

https://t.me/joinchat/IJeomJVB9lf2RxiA

Attendance Taking

Lab Admin

Admin Information

Course Breakdown for Lab Sessions

- Practical Assessment 1 (15%) → **Week 7**
- Practical Assessment 2 (20%) → **Week 12**
- Individual Project (10%)
- Best 5 out of 7 labs (5%)
    - **Best to complete all labs!!**

Admin Information

Recitations

- Submit your recitation answers by Sunday 2359 weekly
- Submission attempts are counted (i.e. you will get marks for submitting something)
- Full marks for submitting code for at least 8 recitations

Basic Coding Concepts

- Equality in Java
  - String equality
  - Class equality
- toString method
- @Override
- Pass by value vs Pass by reference
- Access Modifiers

Equality in Java

String equality

- Use .equals() instead of == to compare Strings()
- == compares memory address

```
String a = "hello";
String b = "hello";

System.out.println(a == b); // false
System.out.println(a.equals(b)); // true
```

Class equality

- Create your own implementation of .equals() method to compare Java Classes!
- By default, .equals() class is inherited from Object class
  - It compares memory address by default
  - Basically the same as just using ===
- Need to override this default class to make meaningful comparisons
  - E.g. if colour and model of two cars are the same, then two objects are the same → return true

toString

- Override toString method to make any object print out a custom string
- By default, Java will print out ClassName@memoryAddress

@Override

- Not necessary but good to have
- If tag is included, code will not compile if method is not overridden properly
    - Useful for checking if:
        i. You have overridden the correct method
        ii. You have overridden the method properly

```
@Override
public String toString() {
    return "Hello!";
}
```

Pass by value vs Pass by reference

- Mainly affects method calls!
- Pass by value: value is *copied* and passed into method
  - **Does not modify original object!**
  - Primitive types are passed by value
- Pass by reference: *actual address* of object is passed into method
  - **Modifies original object!**
  - Reference types (objects) are passed by reference

Pass by value

```
private static void swap(int a, int b) {
    int temp = a;
    a = b;
    b = temp;
}

int a = 1;
int b = 2;
swap(a, b);
System.out.println("a: " + a + ", b: ");

// output: a: 1, b: 2
```

Pass by reference

```
private static void swap(int[] array, int a, int b) {
    int temp = array[a];
    array[a] = array[b];
    array[b] = temp;
}

int[] array = {0, 1}; // array[0] = 0, array[1] = 1
swap(array, 0, 1);
System.out.println("array[0] = " + array[0] + ", array[1] = " + array[1]);

// array[0] = 1, array[1] = 0
```

Access Modifiers

- **private**: cannot be accessed outside of this class
  - Typically for class level attributes and helper methods.
- **protected**: can be accessed by this class and its child classes
  - Typically for class level attributes and methods that a super/parent class wants the sub/child class to inherit.
- **public**: can be accessed by everyone
  - Typically for constructors, getters, setters & methods that can be used by an instance of a class, for example: “public Point getMidPoint(int x, int y)”.
- **default** (i.e. no modifier): Package private. Will talk more about this after you learn about package

Access Modifiers

- Important: **static** keyword. For methods that we want to be able to use without instantiating an object of the class.
  - Static methods/variables belong to the ***class*** and NOT ***instances of an object***
  - For example, Math.PI, Math.atan(). It doesn't make sense to do Math math = new Math(); math.PI or math.atan() everytime you want to access those values/methods.

Tips

- **HIGHLY RECOMMENDED** to finish the lab before the next lab session to prevent it from piling up
- Use vim to code instead of IDE to simulate the exam environment, so that you do not panic during exams
- To compile your code in cmd using *javac* <FILE NAME>, before running the file using *java*
- Do **NOT** plagiarize as you are only cheating yourself and you will face disciplinary actions as we do check for plagiarism in the codes you submitted
- If you are unsure of Java do seek help from me **ASAP**
- Post in the Telegram group chat for any questions relating to Lab, topics or setting up the environment so that everyone can learn together :)
- Feel free to telegram me if you need extra help

LINUX COMMAND

LINUX COMMANDS

ls

- List directory and files

cd <FOLDER PATH>

- Change current directory to <FOLDER PATH>

mkdir <DIRECTORY>

- Create <DIRECTORY> if it does not exist

cat <FILE>

- Prints out the content in the file

LINUX COMMANDS

mv <OLD FILE PATH> <NEW FILE PATH>

- Rename/Move file

rm <FILE PATH>

- Delete file

<tab>

- Auto Complete

LINUX COMMANDS

cp <SOURCE> <DIRECTORY>

- Copy <SOURCE> to <DIRECTORY>
- E.g cp *.java New_Folder
- * is a wildcard meaning all
- This command copies all .java files in <SOURCE> to <DIRECTORY>

java Main < test.in

- Outputs the results of the test cases

jshell Point.java < test.in

- Outputs the results of the test cases in jshell

VIM COMMANDS

VIM COMMANDS

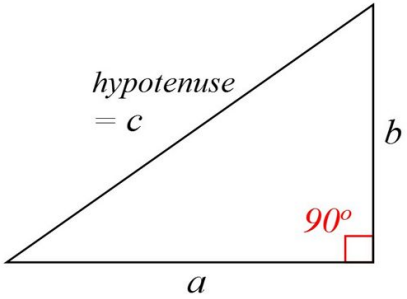
- esc - Change to command mode
- Y - Copy/Yank line
- dd - Delete line
- p - Paste text after cursor
- P - Paste text before cursor
- gg=G - Indent all the code in the file
- :%s/<TO REPLACE>/<REPLACE WITH>/gg - Replace everything that matches **WITHOUT** prompt
- :%s/<TO REPLACE>/<REPLACE WITH>/gc - Replace everything that matches **WITH** prompt
- i - Change to insert mode

VIM COMMANDS

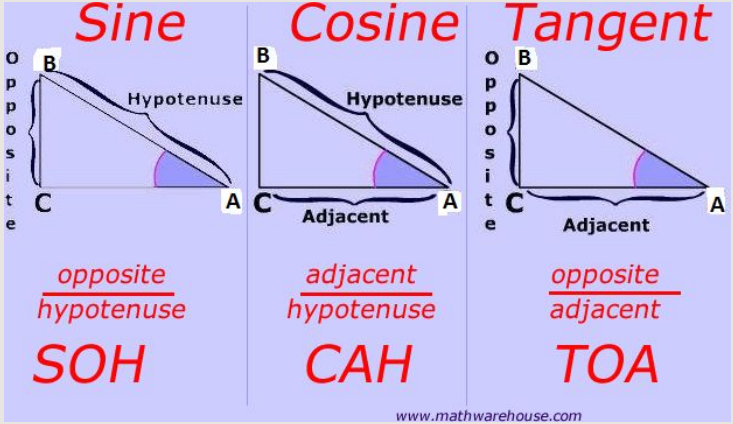
Credits to CS2040

https://bit.ly/3j47Buy

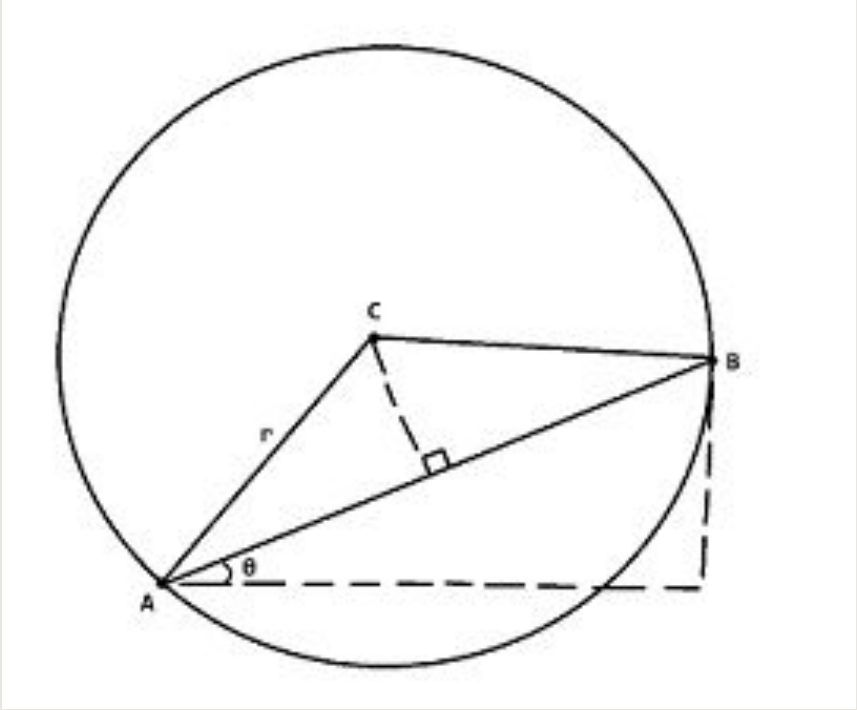
LAB 1


hypotenuse
= c
b
90°
a

$$c^2 = a^2 + b^2$$


Sine
Cosine
Tangent
Opposite
B
C
A
Hypotenuse
Adjacent
opposite / hypotenuse
adjacent / hypotenuse
opposite / adjacent
SOH
CAH
TOA
www.mathwarehouse.com

LAB 1



THANKS!