CMPUT 301 Fa19 - INTRO TO SOFTWARE ENGINEERING Combined LAB LEC Fa19

Lab 1 Instructions

REBOOT COMPUTERS INTO 301 MODE

- Restart and change to the CMPUT 301 image
- Make sure no one is on the broken computers

SELECT USE DEFAULT SETUP

How to connect to the internet

```
ssh 10.0.0.1
```

Use Firefox instead of Chrome

Welcome to CMPUT 301 labs (Can use laptops, but it would be best to try on the lab machines for this lab (TAs can't give support on your laptops!))

Download script (Installing Android Studio from WebFile) off of eclass and run it

- Have to add permissions

```
cd Downloads
chmod u+x install-android-studio-web.sh
```

Run the install script

```
./install-android-studio-web.sh
```

```
Go to Android Studio folder and Run this command in terminal:
studio.sh
```

1. Download OOP Notes and follow the presentation of the TA to know about Java OOP.

2. After the presentation, create LonelyTwitter Project and Follow the instruction under the ##**LAB Demo**## section. Follow the TA to complete the LAB Demo.

3. After completing the **LAB Demo**, goto *Lab Participation Exercise 1* and complete it.

############ **LAB Demo** ####################

1. Create a new LonelyTwitter project. Make sure that the project language is **Java**, **not** **Kotlin**!

- `Create Tweet` Class (Click > New > Java Class)
- Make attributes (`Date date` & `String message`) (use alt+enter to include)
  - Note: Access modifiers
    - `private`= class only
    - No modifier = within package
    - `protected` = through inheritance
    - `public` = everyone!
- Create two Constructors (one with the only Message and the other with Date+Message as arguments) and use Date = `new Date() (current date and time)` for the first constructor (the Default value for date).
- Note: Java `Object` Class (everything extends it, calls its constructor and it has built-in methods like `toString()`)
- Note: the `this` keyword (`message = message` doesn't do anything!)
- Make a regular tweet in `LonelyTwitterActivity` (pass in an empty string)

2. Getters and setters

- Try setting or getting a private variable --> Need setters!
- `setMessagemessage.length()` shouldn't be longer than 140 chars - implement it.

3. Inheritance

- Make `ImportantTweet` child class (extends `Tweet`)
  - call `super` in both of `ImportantTweet's` constructors
- Now have access to the parent's methods and attributes. except constructors! (try and make an important Tweet)

```
ImportantTweet(String message){
    super(message);
}
```

- Super calls the parent's constructor (there is a hidden call to Object's constructor)
- Change the `Tweet` to an `ImportantTweet` in `LonelyTwitterActivity`

4. Abstract Stuff

- Make Tweet Class Abstract
  - `public abstract class Tweet { ... }`
  - `public abstract Boolean isImportant();`
- What if they need to behave differently? `@Override` isImportant() to create a compile-time check
- Make a NormalTweet class, could have many types of tweets
  - call `super` in both of `NormalTweet's` constructors
  - `isImportant` method should return `Boolean.FALSE`
- What if we want to use both in our list? (Implicit upcasting)

```
ArrayList<Tweet> tweetList = new ArrayList<Tweet>();
tweetList.add(normalTweet);
```

- Abstract method and base class so all the classes have the `isImportant()` method
- An interface can also be used to force the use of some methods

```
public interface Tweetable {
  public String getMessage();
  public Date getDate();
}
```

- Make `Tweet` implement `Tweetable`

Last modified: Friday, 13 September 2019, 11:28 AM

◄ Book: Head First Object-Oriented Analysis and D Jump to... Lab Participation Exercise 1 ►