COMP2511

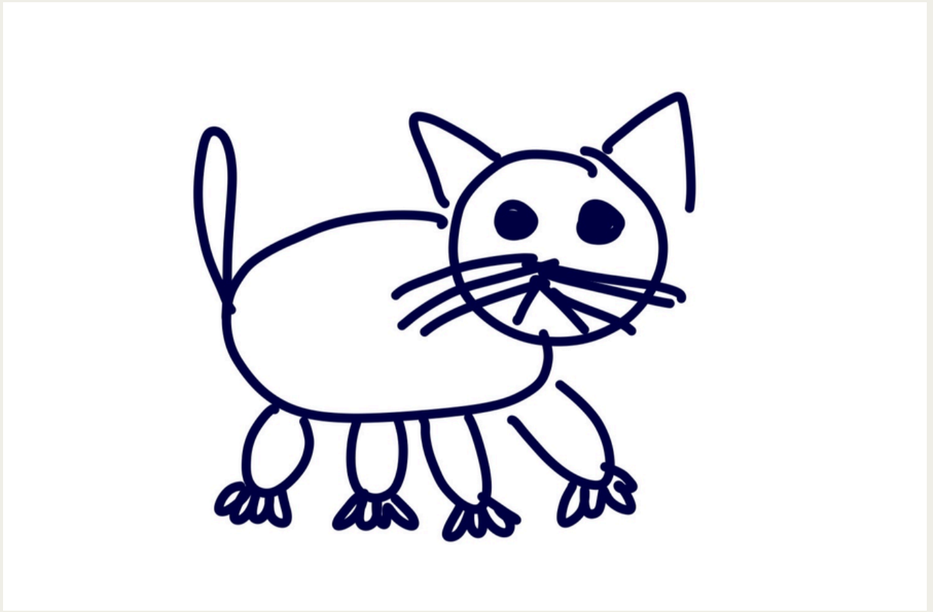
WEEK 1

Software Design!!!!

UNSW

INTRODUCTION

- Jeremy Le (jeremy.le1@unsw.edu.au)
- 4th year Computer Science / Science (Mathematics)
- Third time teaching COMP2511 🙏
- Did COMP2511 in 23T3!
- Taking COMP4920, MATH3171, MATH3511
- Enjoys: Walks, Cooking and Competitive Programming



AGENDA

- Introduction
- Solving Design Problems
- Hello Java and Git Revision
- Java Programming
- Abstraction

ADMIN STUFF

Getting Help

1. Forum
2. Help Sessions
3. Ask your friends
4. Email me (jeremy.le1@unsw.edu.au)

Email the course if you would like anything escalated to course admins (cs2511@cse.unsw.edu.au)

Platforms

1. Course Website
2. GitLab (Code)
3. Discourse Forum
4. Moodle / Echo360 (Lectures)

Assessment Structure

Assignment I : 15% (Week 5)
Assignment II: 20% (Week 10)
Coursework (Labs): 15%
Final Exam: 50%

Tutorials / Labs

1 hour tutorial, 2 hour lab
Tutorial: Content + Demonstration
Labs: Exercises, General Help
At least ~50% of the exam

Slides & Code:
https://github.com/jeremyle56/tutoring/tree/25T3/25T3/cs2511

DESIGN PROBLEM

UNSW has decided that they want to create their own light rail, which takes students from upper campus to lower campus. Design a solution for this - how will it work? What will need to be changed about the campus layout for it to work?

UNSW

WHAT IS GOOD DESIGN?

- Design is inherently a subjective topic, so what do we necessarily mean by ‘well-designed code’ or ‘good design’?
- What are some things that we can all agree are desirable in code?
  - At the text level, code that adheres to widely used **conventions** and stylistic **patterns** for readability.
  - Logic that is **correct, yet simple** enough to read and understand.
  - Being able to focus on how things operate at a high level, rather than having to worry about concrete implementation details (**abstraction**).
  - Having responsibilities and logic be separated into different parts that work together as a whole to form a **cohesive** unit (e.g. hopefully, your COMP1531 project!).
  - Having the ability to easily **adapt** to account for changes in requirements.

OBJECT ORIENTED PROGRAMMING

- Object-oriented programming (OOP) is a programming paradigm which dictates that logic should be organised around user-defined types called **classes** and the interactions between them.
- A class is a structure which holds its own data (like structs in C/C++) and methods (i.e. functions) that act on that data and potentially interact with other classes.
- A class serves as a template/blueprint to create an object. The object itself it an instantiation of a specific class.
- We will be exploring how we can apply OOP to design larger-scale extensible, flexible, maintainable and reusable systems.
- Object-Oriented Programming is most commonly associated with the principles of **encapsulation**, **abstraction**, **inheritance** and **polymorphism**.

JAVA VS OTHER LANGUAGES

Scopes and Code Blocks

- Similar to C and JavaScript/TypeScript, **Java** uses the curly brace syntax { and }.

Classes and Object Oriented Program

- **Java**, Python, JavaScript/TypeScript support Object Oriented Programming (OOP)
  - This means they have classes and inheritance.
- All code in **Java** has to exist within a class.

Types:

- C, TypeScript and Java are statically typed.
- Python, JavaScript are dynamically typed.

JAVA VS OTHER LANGUAGES

Memory:

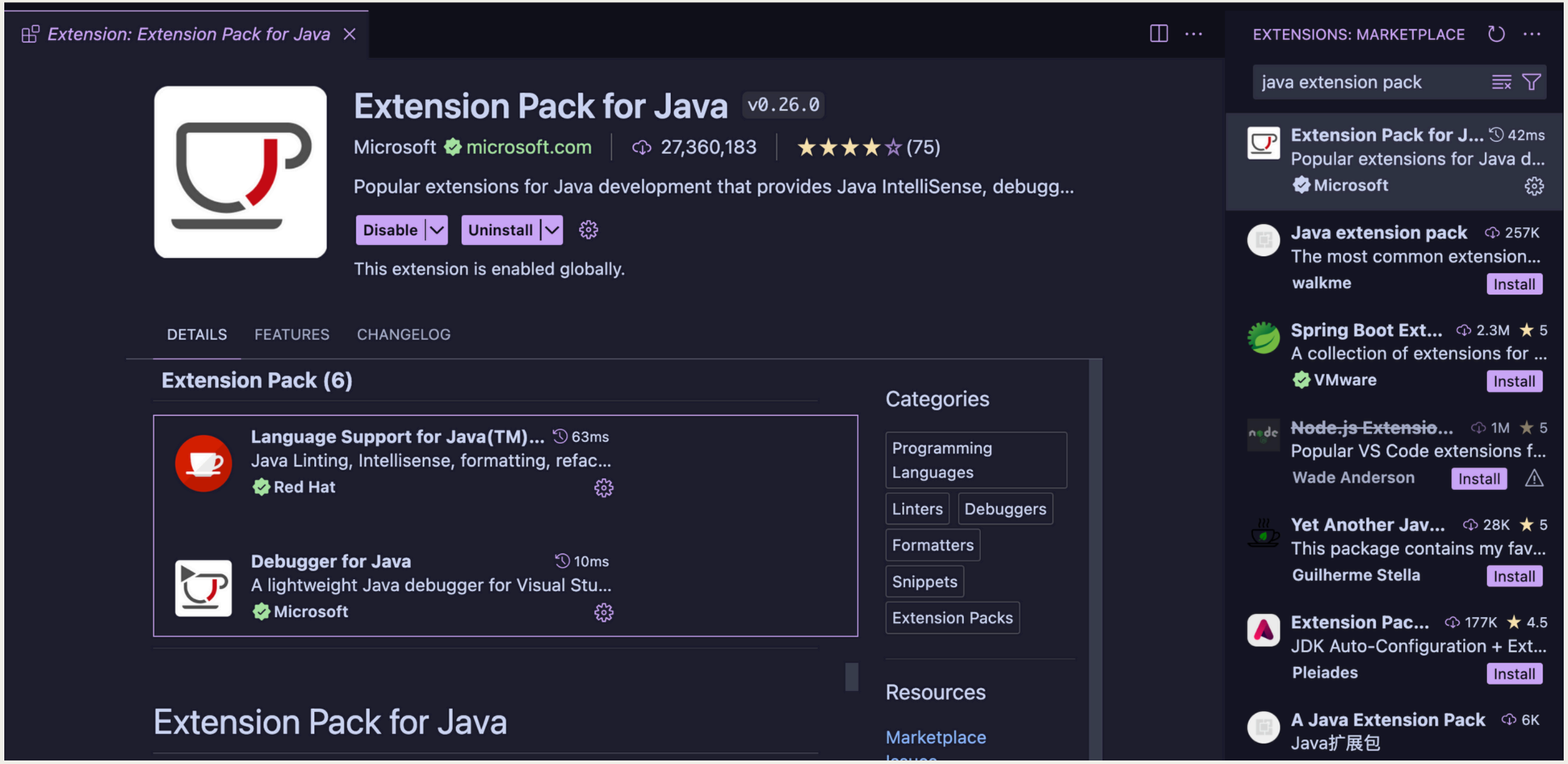
- C allows for manual memory via the standard library.
- **Java**, Python, JavaScript, TypeScript have automatic memory managment with a garbage collector.

Compilation:

- C compiles into machine code and seperate executable.
- **Java** and Python compiles into byte code which is interpreted.
- TypeScript transpiles into JavaScript code which is interpreted.

VSCODE EXTENSIONS

Ensure that you have the correct extensions installed



HELLO JAVA

Make a simple Java program that prints "Hello World" in the HelloWorld class.

```java
public class HelloWorld {

    Run | Debug
    public static void main(String[] args) {
        System.out.println(x:"Hello World");
    }

}
```

GIT REVISION

- Version Control software
- Allows you to save snapshots of code
- Combined with cloud providers / online storage, i.e. Gitlab / Github / Bitbucket, allows users to work on code together

git add: Stage files
git commit: Commit the staged files as a snapshot
git push: Push your new commits to an online origin
git status: State of current repository & branch
git log: History of current branch

SUM

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.

SHOUTER

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.

ABSTRACTION

What is abstraction?

Abstraction is the process of generalising concrete details, such as attributes, away from the study of objects and systems to focus attention on details of greater importance.

What examples of abstraction have we seen in previous courses?

Abstract Data Types (ADTs)

- Linked Lists
- Trees
- Graphs
- Etc...

In C:

- List.h file which provides an “interface” for List.c which is the implementation using the interface.

How does abstraction allow us to write better software?

Reducing complexity of code, reducing the reliance of code and “connectedness” of the whole project. (Coupling)

This means changing one class (section of code) will only have little effect on other aspects of the codebase.

Discuss how OOP takes abstraction to another level from what we have seen previously.

Abstraction is used to hide unnecessary information and display only necessary information to the users interacting.

Everything in Java is an object!

All objects provide an abstraction from each other.

Lab Time!!!

YAYYYY

UNSW