Assess Yourself

Write the `Animal findCutest` method for the `Person` class, which accepts one argument:

- `Animal[] animals` - an array of animal objects

This method should return the “cutest” animal, as determined by a given `Person` object?

What are some things we need to know in order to solve this problem?

Assess Yourself

What are some things we need to know in order to solve this problem?

- How do we define/evaluate cuteness?
- What attributes/methods of the `Animal` class will help us with this?

Assumptions:

- A `Person` has the `int getCuteness(Animal animal)` method
- Every `Animal` has an `Image` that is accessed to give a cuteness value

Assess Yourself

```java
public Animal findCutest(Animal[] animals) {

    int cuteness, maxCuteness = 0;
    Animal cutestAnimal = null;

    for (Animal animal : animals) {
        cuteness = this.getCuteness(animal);

        if (cuteness > maxCuteness) {
            cutestAnimal = animal;
            maxCuteness = cuteness;
        }
    }

    return cutestAnimal;

}
```

Assess Yourself

What would change about your solution if we said that little girls determine cuteness differently to old ladies?

You would make `Person` an **abstract** class, and create subclasses for `Girl` and `OldLady`; they would each **override** the abstract `getCuteness` method.

Would the `findCutest` method change?

```
cuteness = this.getCuteness(animal);
```

No. This is a good example of how polymorphism works; we will use this kind of solution again when we learn about *design patterns*.

SWEN20003
Object Oriented Software Development

Software Tools

Semester 1, 2019

The Road So Far

- OOP and Java Foundations
- Classes and Objects
- Abstraction
  - Inheritance
  - Abstract Classes
  - Polymorphism

Lecture Objectives

After this lecture you will be able to:

- Make use of **version control**
- Describe/use various **tools** for software development
- Contribute to **Open Source Software** (OSS) projects

Version Control

What is `git`?

`git` is a type of *version control* system. It is probably the most commonly-used modern system, originally created by Linus Torvalds of Linux fame.

Keyword

Version Control: A systematic approach to storing different versions of source code, data, etc. to ensure changes can be reviewed and reverted.

Using `git` - Step 1: Initialising

The first step is to create a *repository*. We do this by running the command `git init`. This makes the current folder into a `git` repository.

Keyword

Repository: A collection of files to be stored by the version control system, together with some metadata to keep track of them.

Note that any files placed in this folder are not automatically added to the repository. We must do this separately.

Using `git` - Step 2: Staging Changes

Once you have created a repository, we can add files to it using the command `git add`. Some examples:

- `git add .`: adds all files in the current directory to the repository
- `git add file.java`: adds only `file.java` to the repository

"Adding" files also updates any changes made to them, or files that have been deleted; this is called *staging* the changes.

Using `git` - Step 3: Committing Changes

Once we have staged some changes, we can save our changes as a new version of the program. This is called *committing* the changes, and we can do it with the command `git commit`.

- `git commit`: commits the staged changes, and opens a text editor to provide a commit message
- `git commit -m "message"`: commits the staged changes, with the message “message”.

Using `git` - Step 4: Fixing Mistakes

If you accidentally delete some files, or break everything by changing the code, you can easily go back to the previous committed version using `git reset --hard`.

Using `git` - Useful Commands

Here are a couple of commands that are good to know in your `git` adventures.

- `git status`: shows you the current status of your commit, including which files have been modified, and which of those modifications are staged.
- `git log`: shows you the history of the repository, including any previous commits

Storing `git` repositories online

You can also create repositories that can be stored online on websites such as Bitbucket or GitHub.

Once you create the repository online, you can use `git clone <url>` to *clone* the repository to your computer, instead of using `git init` to create a new repository.

After you have cloned your repository and made some commits, you can update the online repostory using the command `git push`.

But wait, there's more...

This has been a very quick introduction to `git`! There are many more advanced features that make *collaborating* on projects much easier, such as branching and merging. I have only focused on what you need to get started with your own repositories.

Git Repositories

What does a `git` repository look like if we *don't* use the command line?

Introducing SourceTree...

Team Projects

Team Project Tools

Managing a single person software project is relatively easy.

Managing a project with multiple people is a recipe for disaster... As I'm sure we all know.

But there are tools to help.

Issue Tracking

How many people use “To Do” lists?

Did you know that “To Do” lists are super important for software projects?

To do lists for developers.

Communication

How many people use Messenger to communicate with teammates?

How many people send files/updates/angry messages to teammates over Messenger?

How many people would love to separate "work" messages from "social" messages?

Messenger for developers.

Global Projects

What is an Open-Source Project

Keyword

Open-Source: Software for which the original source code is made freely available and may be redistributed and modified.

How do they work?

All Open-Source projects are... well, open.

Anyone can contribute... So long as you follow the “rules” of the project.

Each project also comes with their own set of *tools*; some are mandatory (like a build system, or a project management tool), some are optional (like joining a Slack team).

How do you start?

There are plenty of beginner-friendly entry points, such as:

- First Timers Only
- Jump in here, or here, or here

Try your hand at a simple, easily completed task. What's the worst that can happen?

Why do you care?

Contributing to open-source projects is one of the best ways to demonstrate employability.

Why?

Because making a contribution, big or small, requires a lot of things:

- High quality code
- Understanding a complex application
- Learning new, unfamiliar skills
- Working in a distributed team
- Following the *rules*, *expectations*, and *guidelines* of the project

Google

Google

Google is literally a developer's best friend.

In no universe will you get hired for a job where you understand or are familiar with 100% of the tools. You need to be able to *find your own solutions*.

How do you demonstrate you can do that?

- Contribute to an open-source project!
- Build your own projects, outside of the scope of university
- Do things you've never done before, that force you to learn new skills; then talk about it!

Advanced Debugging

What techniques do you use to debug?

- "Debug by `printf`"?
- Randomly change code and hope it works?
- Or the student favourite: ask the teacher!

Fortunately, there's a better way...

Advanced Debugging

Keyword

Breakpoint: A predetermined line where your code will *pause*, allowing you to inspect its state; the contents of variables, any active methods, etc.

Metrics

What should you take from today's lecture?

- `git` is useful, and you should learn it as soon as you can
- There are a million and one tools you can use to make projects easier, particularly in teams; get familiar with a few of them
- Open-source projects are excellent gateways for skill-building; give them a go
- Learn. How. To. Google.
- The debugger is your friend

Project 1 Demonstration