# COMP2511

Week 2

TUESDAY 1PM - 4PM (T13B)

## Today

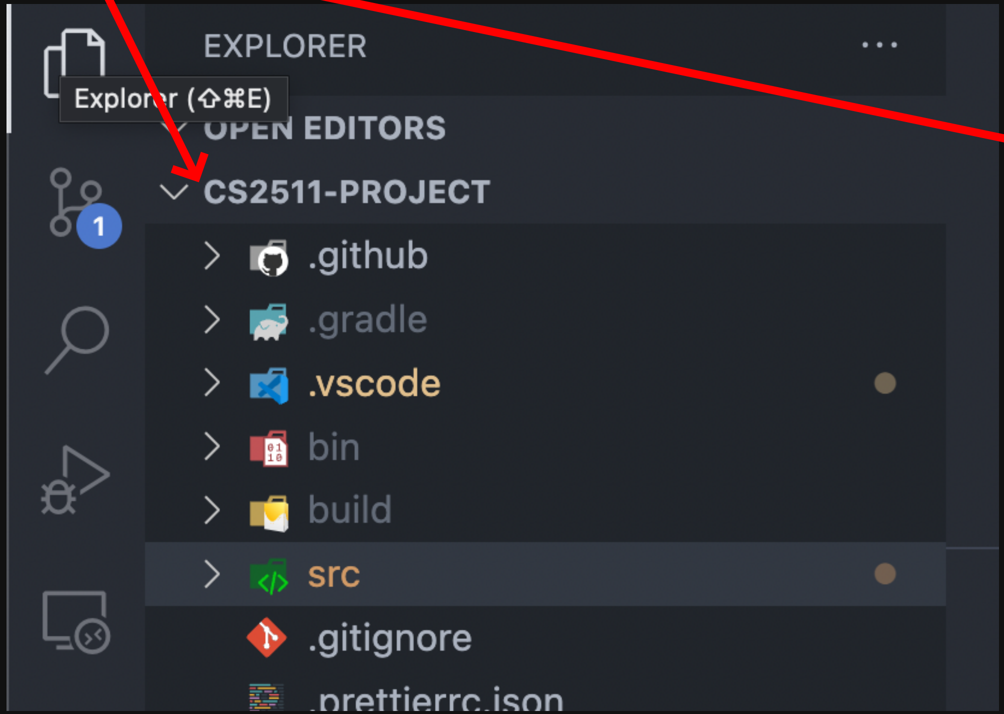
- Some VSCode, GitLab, Git tips
- Classes
- Commenting & Documentation
- Basic Inheritance
- Access Modifiers

# VSCode Config Tips

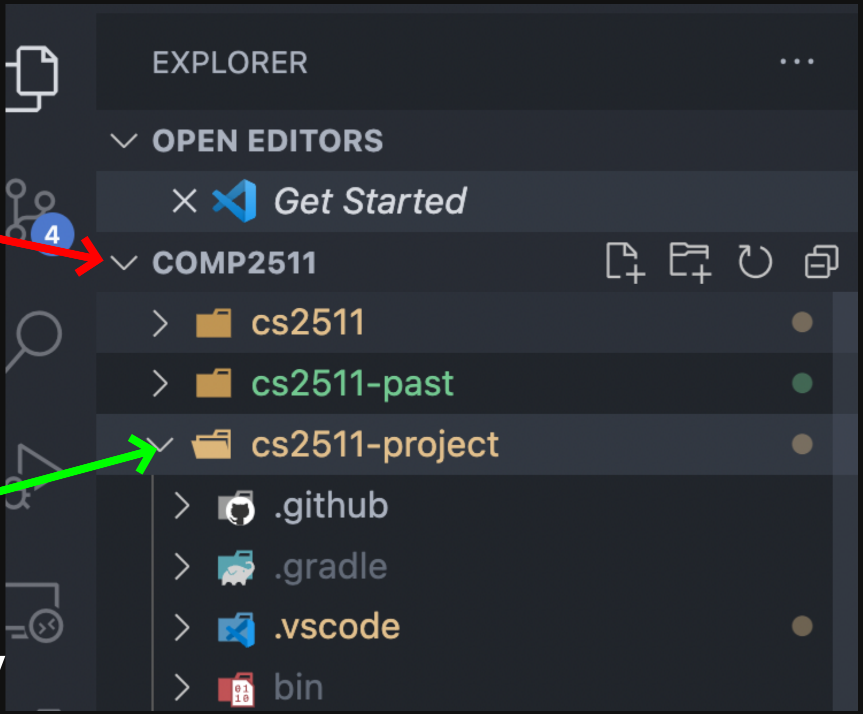
Ensure that you open the correct folder

example: cs2511-project

Name of folder open



Good

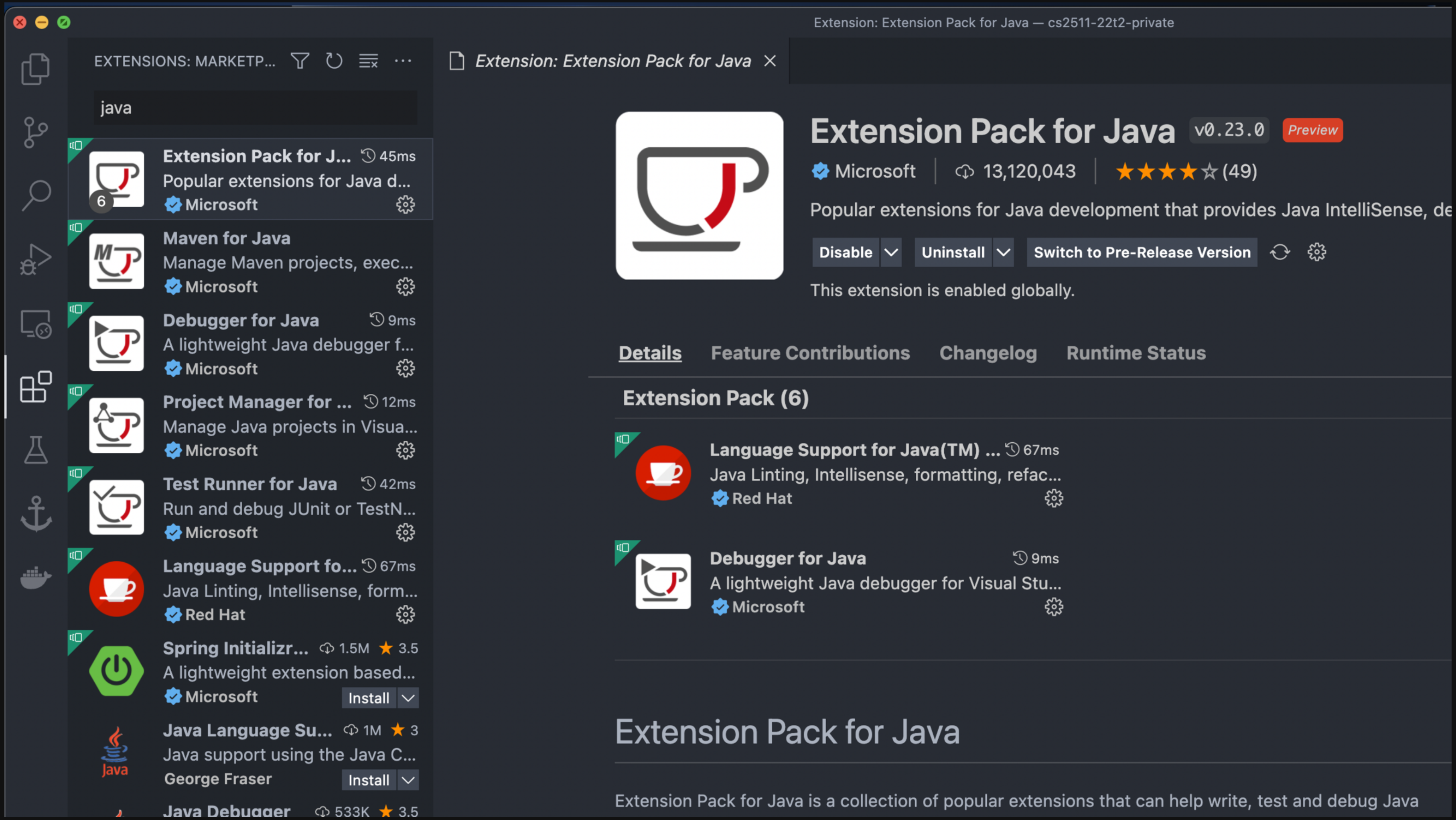


The folder I want to actually open

Bad

# VSCode Config Tips

Ensure you have the correct Java extension installed



Extension: Extension Pack for Java — cs2511-22t2-private

EXTENSIONS: MARKETP...

java

**Extension Pack for J...** 45ms
Popular extensions for Java d...
Microsoft

**Maven for Java**
Manage Maven projects, exec...
Microsoft

**Debugger for Java** 9ms
A lightweight Java debugger f...
Microsoft

**Project Manager for ...** 12ms
Manage Java projects in Visua...
Microsoft

**Test Runner for Java** 42ms
Run and debug JUnit or TestN...
Microsoft

**Language Support fo...** 67ms
Java Linting, Intellisense, form...
Red Hat

**Spring Initializr...** 1.5M ★ 3.5
A lightweight extension based...
Microsoft — Install

**Java Language Su...** 1M ★ 3
Java support using the Java C...
George Fraser — Install

**Java Debugger** 533K ★ 3.5

*Extension: Extension Pack for Java*

**Extension Pack for Java** v0.23.0 *Preview*

Microsoft | 13,120,043 | ★★★★☆ (49)

Popular extensions for Java development that provides Java IntelliSense, de

Disable | Uninstall | Switch to Pre-Release Version

This extension is enabled globally.

Details | Feature Contributions | Changelog | Runtime Status

**Extension Pack (6)**

**Language Support for Java(TM) ...** 67ms
Java Linting, Intellisense, formatting, refac...
Red Hat

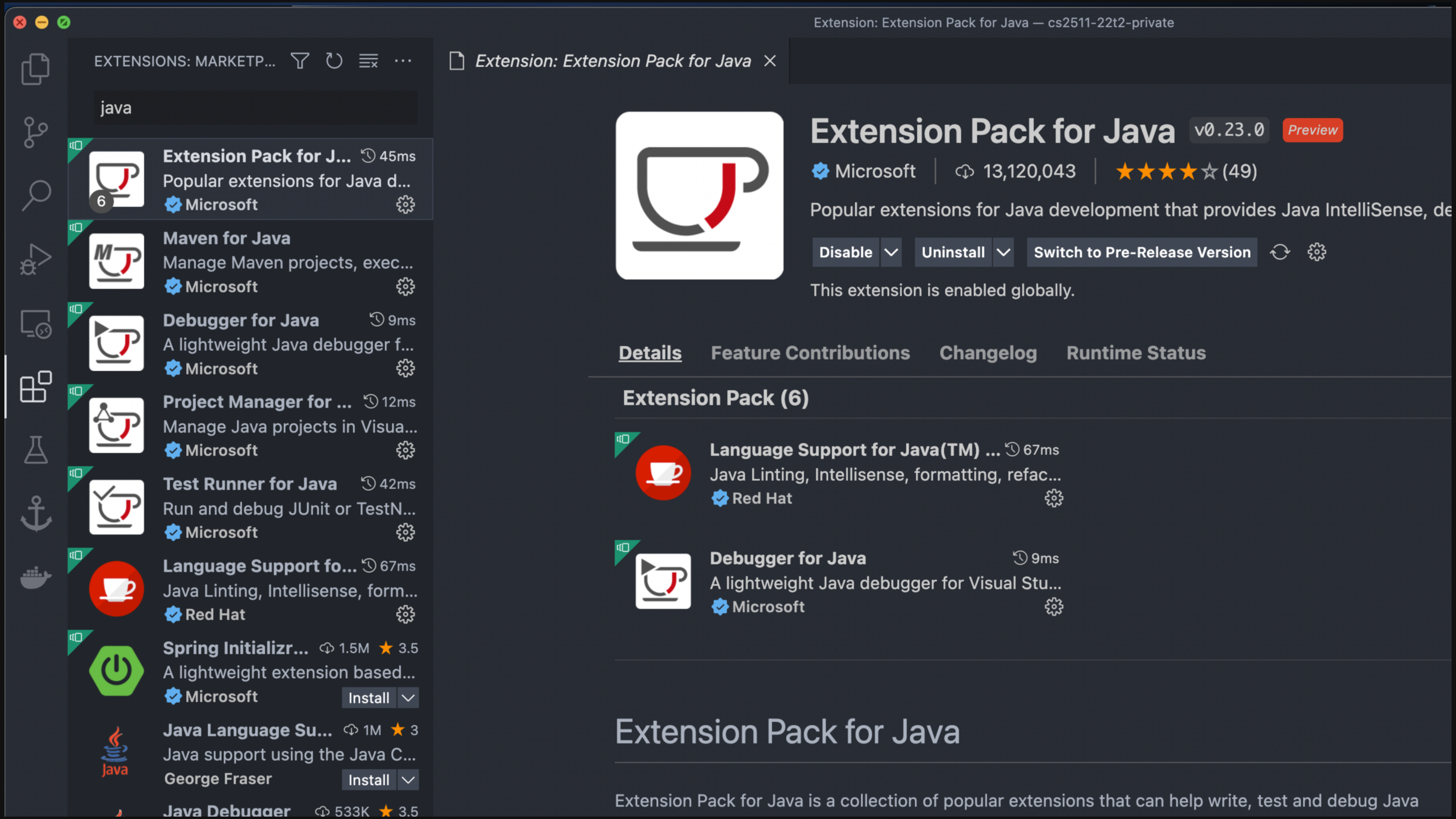
**Debugger for Java** 9ms
A lightweight Java debugger for Visual Stu...
Microsoft

## Extension Pack for Java

Extension Pack for Java is a collection of popular extensions that can help write, test and debug Java

# VSCode Config Tips

Ensure you have the correct Java extension installed



Extension: Extension Pack for Java — cs2511-22t2-private

EXTENSIONS: MARKETP...

java

Extension Pack for J... 45ms
Popular extensions for Java d...
Microsoft

Maven for Java
Manage Maven projects, exec...
Microsoft

Debugger for Java 9ms
A lightweight Java debugger f...
Microsoft

Project Manager for ... 12ms
Manage Java projects in Visua...
Microsoft

Test Runner for Java 42ms
Run and debug JUnit or TestN...
Microsoft

Language Support fo... 67ms
Java Linting, Intellisense, form...
Red Hat

Spring Initializr... 1.5M ★ 3.5
A lightweight extension based...
Microsoft Install

Java Language Su... 1M ★ 3
Java support using the Java C...
George Fraser Install

Java Debugger 533K ★ 3.5

*Extension: Extension Pack for Java*

Extension Pack for Java v0.23.0 *Preview*
Microsoft | 13,120,043 | ★★★★☆ (49)
Popular extensions for Java development that provides Java IntelliSense, de

Disable | Uninstall | Switch to Pre-Release Version

This extension is enabled globally.

Details | Feature Contributions | Changelog | Runtime Status

Extension Pack (6)

Language Support for Java(TM) ... 67ms
Java Linting, Intellisense, formatting, refac...
Red Hat

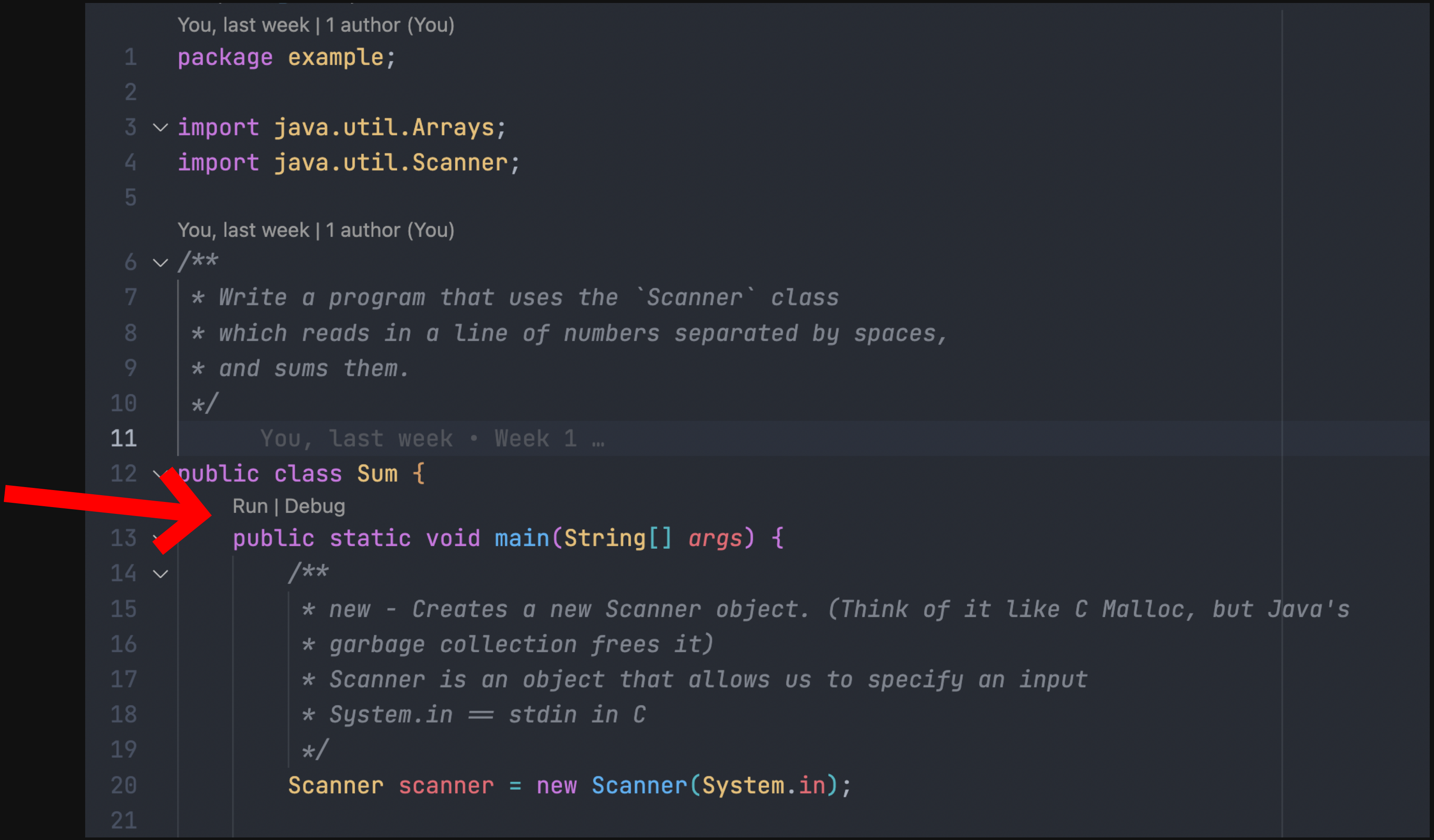
Debugger for Java 9ms
A lightweight Java debugger for Visual Stu...
Microsoft

Extension Pack for Java

Extension Pack for Java is a collection of popular extensions that can help write, test and debug Java

# VSCode Config Tips

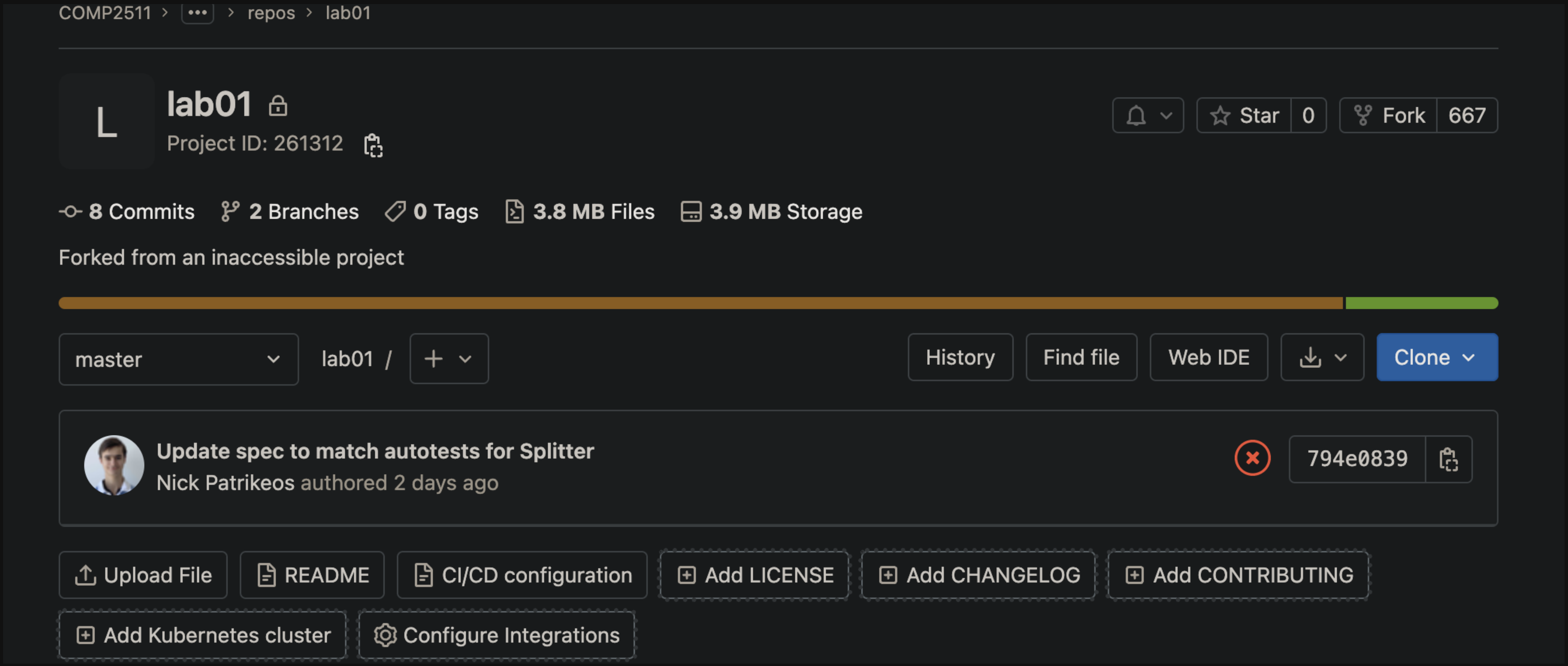
Ensure that you are running code using the "Run" button



```
You, last week | 1 author (You)
package example;

import java.util.Arrays;
import java.util.Scanner;

You, last week | 1 author (You)
/**
 * Write a program that uses the `Scanner` class
 * which reads in a line of numbers separated by spaces,
 * and sums them.
 */
        You, last week • Week 1 …
public class Sum {
    Run | Debug
    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);

```

# GitLab Tips

How to view pipeline output


COMP2511 › ••• › repos › lab01

lab01

Project ID: 261312

Star 0 | Fork 667

8 Commits | 2 Branches | 0 Tags | 3.8 MB Files | 3.9 MB Storage

Forked from an inaccessible project

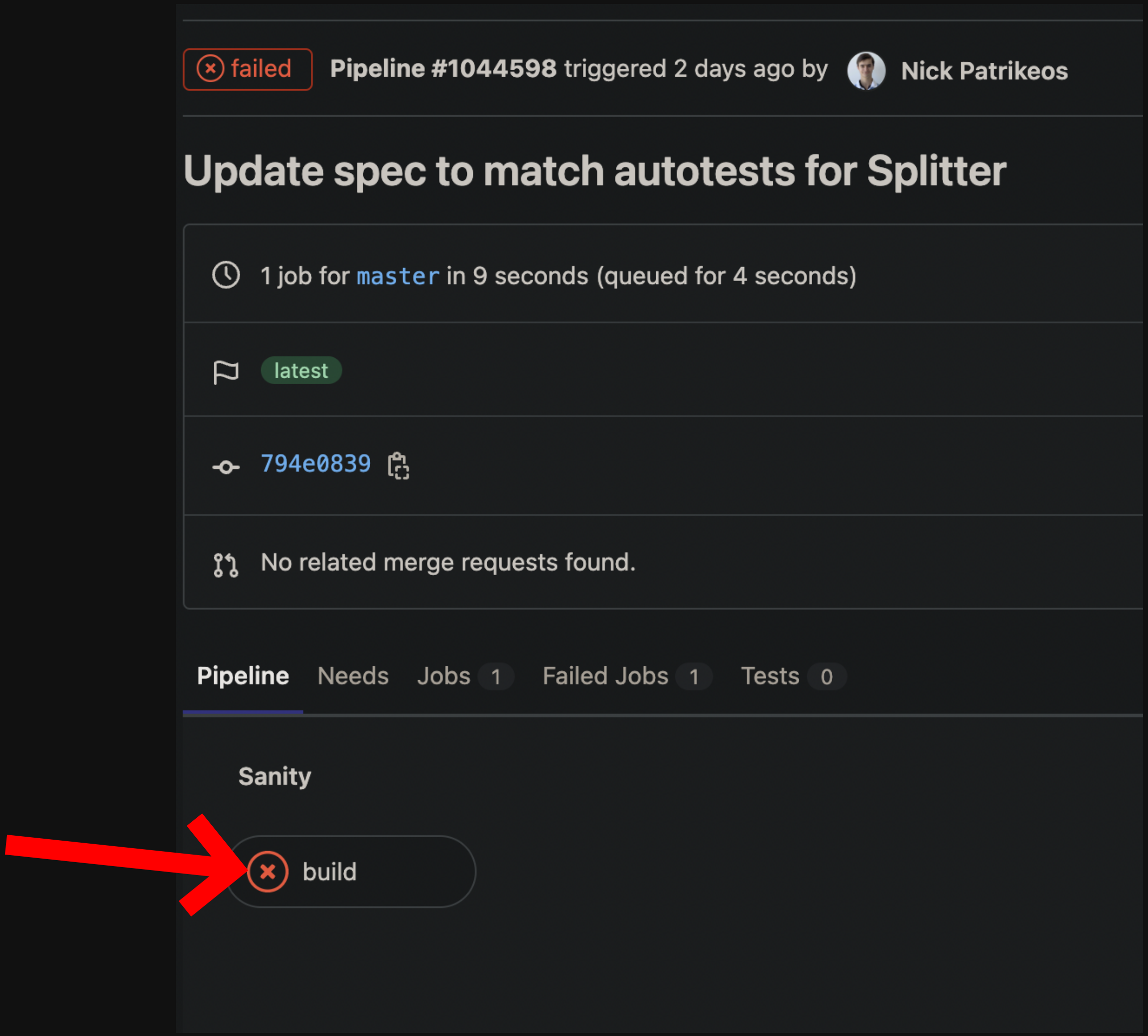
master | lab01 / | History | Find file | Web IDE | Clone

**Update spec to match autotests for Splitter**
Nick Patrikeos authored 2 days ago

794e0839

# GitLab Tips

How to view pipeline output



failed **Pipeline #1044598** triggered 2 days ago by Nick Patrikeos

**Update spec to match autotests for Splitter**

1 job for `master` in 9 seconds (queued for 4 seconds)

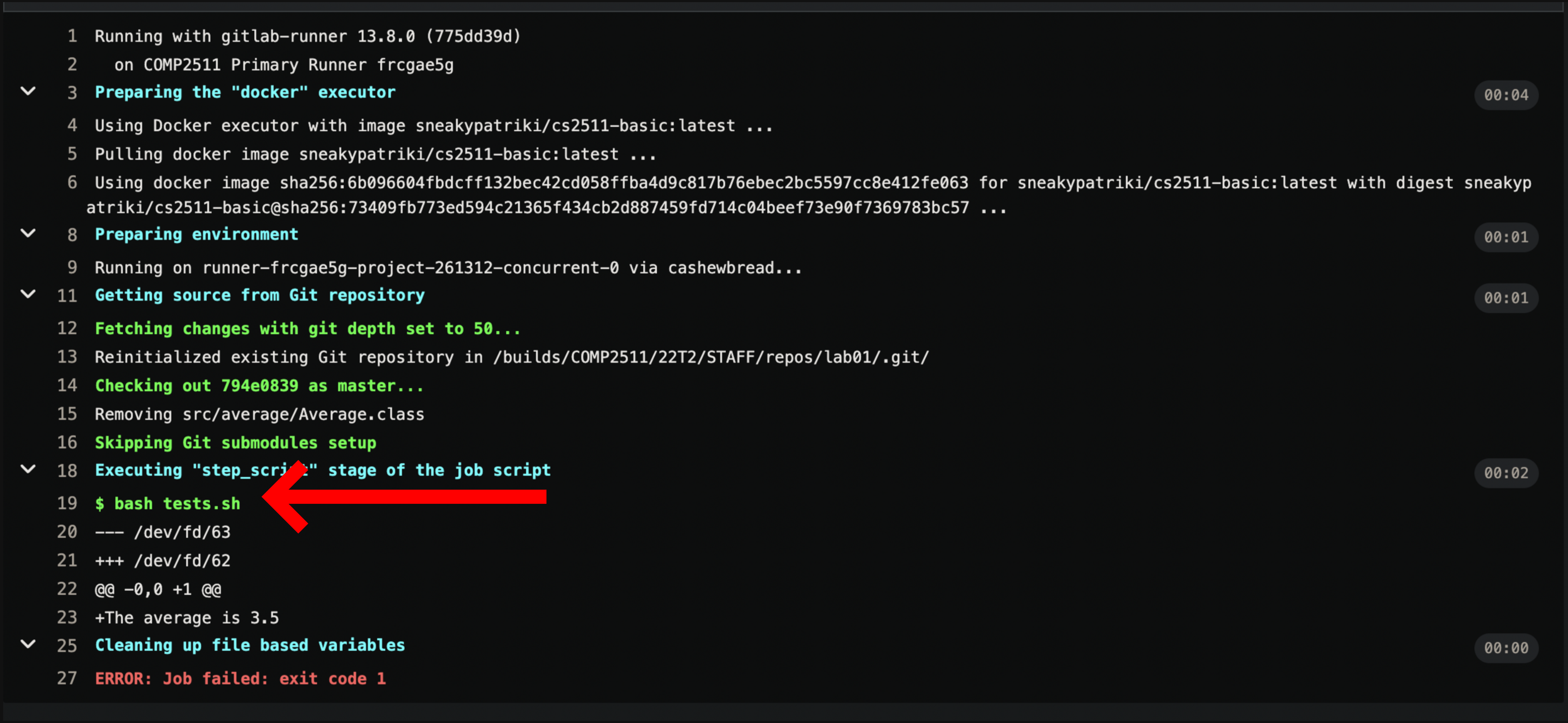
latest

`794e0839`

No related merge requests found.

# GitLab Tips

How to view pipeline output



```
Running with gitlab-runner 13.8.0 (775dd39d)
  on COMP2511 Primary Runner frcgae5g
Preparing the "docker" executor                                                                                                00:04
Using Docker executor with image sneakypatriki/cs2511-basic:latest ...
Pulling docker image sneakypatriki/cs2511-basic:latest ...
Using docker image sha256:6b096604fbdcff132bec42cd058ffba4d9c817b76ebec2bc5597cc8e412fe063 for sneakypatriki/cs2511-basic:latest with digest sneakyp
atriki/cs2511-basic@sha256:73409fb773ed594c21365f434cb2d887459fd714c04beef73e90f7369783bc57 ...
Preparing environment                                                                                                          00:01
Running on runner-frcgae5g-project-261312-concurrent-0 via cashewbread...
Getting source from Git repository                                                                                             00:01
Fetching changes with git depth set to 50...
Reinitialized existing Git repository in /builds/COMP2511/22T2/STAFF/repos/lab01/.git/
Checking out 794e0839 as master...
Removing src/average/Average.class
Skipping Git submodules setup
Executing "step_script" stage of the job script                                                                                00:02
$ bash tests.sh
--- /dev/fd/63
+++ /dev/fd/62
@@ -0,0 +1 @@
+The average is 3.5
Cleaning up file based variables                                                                                               00:00
ERROR: Job failed: exit code 1
```

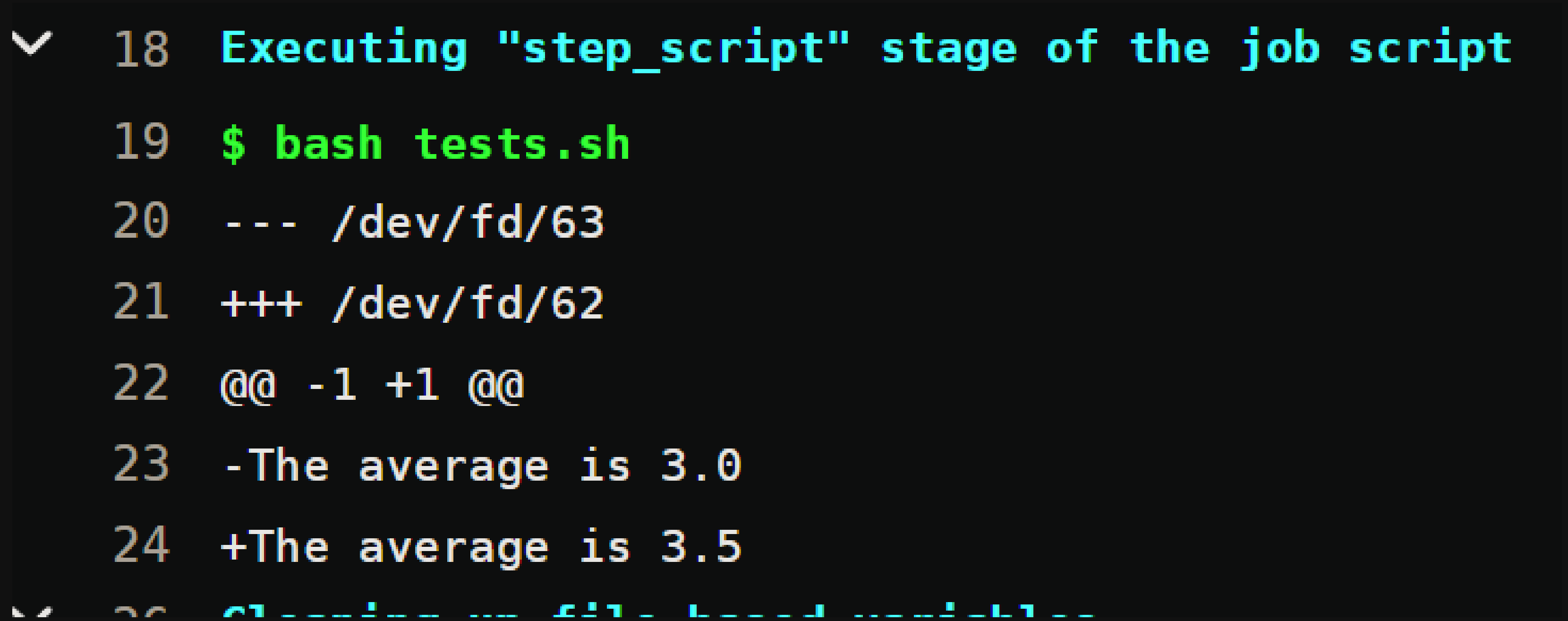
# Test File

How to view pipeline output

- Compares the difference between your output and what is expected
- Runs test in order. Average -> splitter -> satellite
- No output == good

```bash
#!/bin/bash
function run_junit() {
    exercise=$1
    rm -rf bin/$exercise
    javac -d bin -cp "$JUNIT" $(find src/$exercise -name "*.java")
    java -jar "$JUNIT" -cp bin:src/$exercise --scan-class-path --disable-ansi-colors --disable-banner 2>&1
}

cd src
# Average
javac average/Average.java || exit 1
diff <(java average/Average) <(echo "The average is 3.5") || exit 1

# Splitter
javac splitter/Splitter.java || exit 1
diff <(java splitter/Splitter <<< "Welcome to my humble abode") <(printf "Enter a message: \nWelcome\nto\nmy\nhumble\nabode\n") || exit 1

# Satellite
javac satellite/Satellite.java || exit 1
diff <(java satellite/Satellite) <(printf "I am Satellite A at position 122.0 degrees, 10000 km above the centre of the earth and moving at a velocity of 55.0 metres per second\n2.129301687433082\n0.04303052592865024\n4380.0\n") || exit 1

cd ..

JUNIT="lib/junit-platform-console-standalone-1.7.0-M1.jar"
```

# Test File



```
Executing "step_script" stage of the job script
$ bash tests.sh
--- /dev/fd/63
+++ /dev/fd/62
@@ -1 +1 @@
-The average is 3.0
+The average is 3.5
Cleaning up file based variables
```

- The '-' (minus) line is your output
- The '+' (plus) line is expected

# Git Config

When you **commit** something, you are effectively saving the staged files as a new snapshot and **signing it** with your **name** and **email**.

You have to configure your git identity if you haven't done it before

```
git config --global user.name "Your Name Here"
git config --global user.email "z555555@unsw.edu.au"
```

You then can add whatever email you have set in user.email on GitLab, so that it recognises all the commits that have been pushed to GitLab.

Please do this, its important Git etiquette

## Code Review

# Code Review

- What is the difference between **super** and **this**?
  - **super** refers to the immediate parent class whereas **this** refers to the current class
- What about **super(...)** and this(...)?
  - **super()** acts as a parent class constructor and should be the first line in a child class constructor
  - **this()** acts as a current class constructor (can be used for method overloading)

```java
package shapes;

public class Shape {
    public String color;

    public Shape(String color) {
        System.out.println("Inside Shape constructor");
        this.color = color;
    }
}
```

```java
public class Rectangle extends Shape {
    public int height;
    public int width;

    public Rectangle(String color) {
        super(color);
        System.out.println("Inside Rectangle constructor with one argument");
    }

    public Rectangle(String name, int width, int height) {
        this(name);
        this.width = width;
        this.height = height;
        System.out.println("Inside Rectangle constructor with three arguments");
    }

    public static void main(String[] args) {
        Rectangle r = new Rectangle("red", 10, 20);
        System.out.println(r.color);
        System.out.println(r.width);
        System.out.println(r.height);
    }
}
```

# Code Review

- What is the difference between **`super`** and **`this`**?
  - **`super`** refers to the immediate parent class whereas **`this`** refers to the current class
- What about **`super(...)`** and `this(...)`?
  - **`super()`** acts as a parent class constructor and should be the first line in a child class constructor
  - **`this()`** acts as a current class constructor (can be used for method overloading)

```java
package shapes;

public class Shape {
    public String color;

    public Shape(String color) {
        System.out.println("Inside Shape constructor");
        this.color = color;
    }
}
```

```java
public class Rectangle extends Shape {
    public int height;
    public int width;

    public Rectangle(String color) {
        super(color); // => Calling constructor of parent `Shape(String color)`
        System.out.println("Inside Rectangle constructor with one argument");
    }

    public Rectangle(String name, int width, int height) {
        this(name); // => Calling constructor `Rectangle(String color)`
        this.width = width;
        this.height = height;
        System.out.println("Inside Rectangle constructor with three arguments");
    }

    public static void main(String[] args) {
        // Rectangle(3 arguments) => Rectangle(1 argument) => Shape(1 argument)
        Rectangle r = new Rectangle("red", 10, 20);
        System.out.println(r.color);
        System.out.println(r.width);
        System.out.println(r.height);
    }
}
```

# Code Review

What are **static fields** and **methods**?

**Static fields** are variables that are common and available to all instances of a Class. They belong to the Class, rather than an instance.

**Methods** are a block of code that perform a task. You can think of them as functions of a class.

```java
package circle;

public class Circle extends Object {
    // Every class extends Object, it is not needed though
    private static final double pi = 3.14159;
    private int x, y;
    private int r;

    // Only 1 variable for all Circle objects
    static int no_circles = 0;

    public Circle() {
        super(); // not needed
        no_circles++;
    }

     public double circumference() {
        return 2 * pi * r;
    }
}
```

# Code Review

What are **static fields** and **methods**?

**Static fields** are variables that are common and available to all instances of a Class.

```java
package circle;

public class Circle extends Object {
    // Every class extends Object, it is not needed though
    private static final double pi = 3.14159;
    private int x, y;
    private int r;

    // Only 1 variable for all Circle objects
    static int no_circles = 0;

    public Circle() {
        super(); // not needed
        no_circles++;
    }

     public double circumference() {
        return 2 * pi * r;
    }
}
```

What are **static fields** and **methods**?

**Static fields** are variables that are common and available to all instances of a Class

**Methods** are a block of code that perform a task. You can think of them as functions of a class.

# Documentation

JavaDoc

# Documentation

- Why is documentation important? When should you use it
- What does the term "self-documenting" code mean?
  - **Code that documents itself**. It is readable inherently. Usually accomplished through variable name and function names
- When can comments be bad (code smell)?
  - Comments become **stale** & does not get updated with new changes
  - Possibly hinting that your design/code is **too complex**

# Documentation

Single Line

```
// Single line comment
```

Multi-line comment

```
/**
 * This is multi-line
 * documentation
 */
```

## JavaDoc

- JavaDoc is one way of documenting in Java.
- JavaDoc is a way of writing your comments
- It mainly targets class definitions and method/function definitions.
- in COMP2511, you will not have to use JavaDoc documentation unless asked.

# JavaDoc

```java
/**
 * File class that stores content under a file name
 */
public class File {
    /**
     * Constructor used to create a file
     * @param fileName the name of the file
     * @param content contents of the file
     */
    public File(String fileName, String content) {}

    /**
     * Constructor used to make a partial file when receiving a new file
     * I.e., content.length() != fileSize with no compression
     * @param fileName
     * @param fileSize
     */
    protected File(String fileName, int fileSize) {}

    /**
     * Checks if transfer has been completed
     * @return true if it has been completed
     */
    public boolean hasTransferBeenCompleted() {}
}
```

## Inheritance

## Inheritance

What is it?

In Java, a class can inherit attributes and methods from another class. The class that inherits the properties is known as the sub-class or the child class. The class from which the properties are inherited is known as the superclass or the parent class.

Known as a "is-a" relationship

## Code Demo

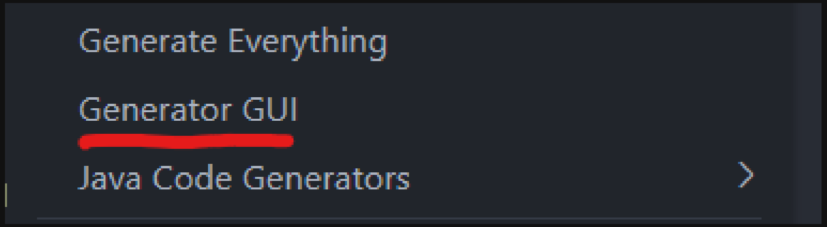
Employee.java & Manager.java

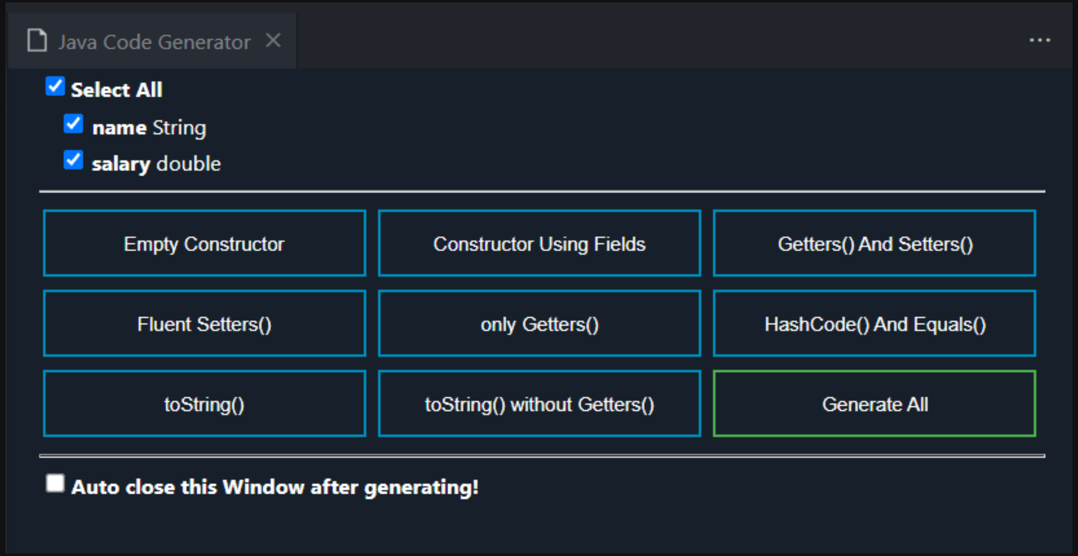
# Code Demo

1. Create a **Employee** class with a **name** and **salary**
2. Create setters & getters with JavaDoc
3. Create a **Manager** class that inherits **Employee** with a **hireDate**
4. Override **toString()** method

# Code Demo

The java extension packs come with some features you can use to generate boilerplate code





# Code Demo

How many constructors does a class need?

Technically none. If a class is defined without a constructor, Java adds a default constructor

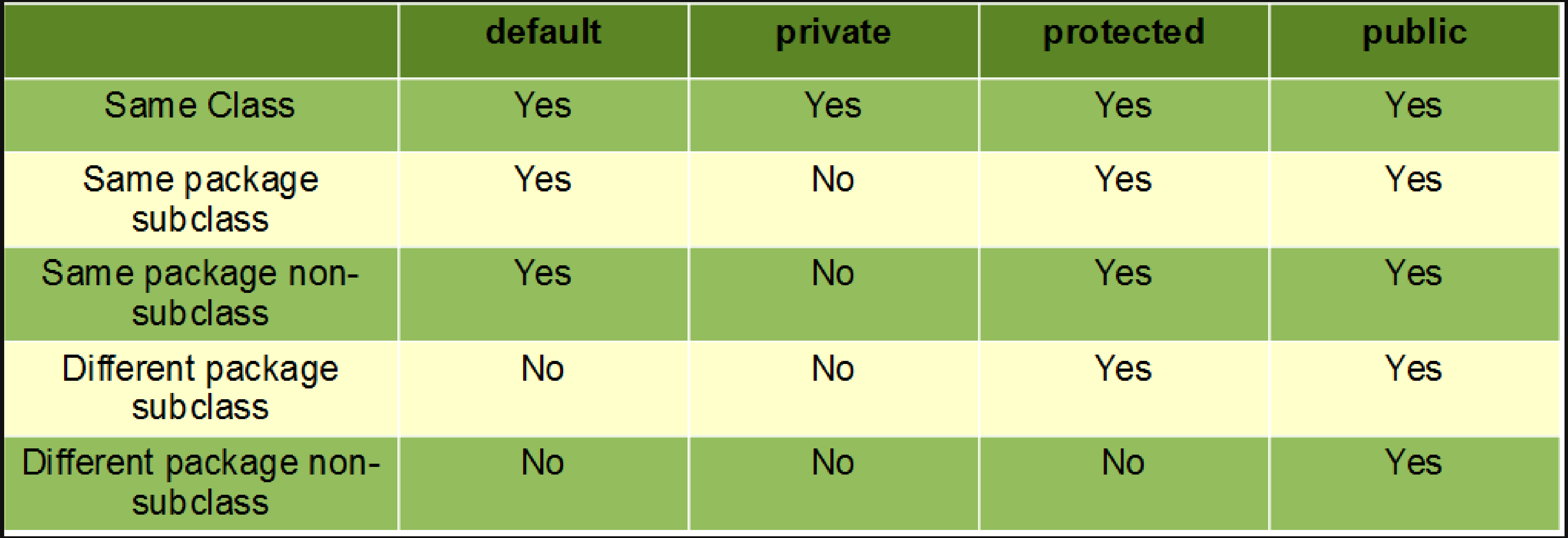
However, if a class needs attributes to be assigned (e.g., has a salary), then a constructor must be assigned

```java
package employee;

import java.time.LocalDate;

public class Employee {
    private String name;
    private double salary;

    /**
     * Creates an Employee with the given name and salary
     * @param name The full name of the employee
     * @param salary The employee's yearly salary
     */
    public Employee(String name, double salary) {
        this.name = name;
        this.salary = salary;
    }

    /**
     * Gets the name of the Employee
     * @return full name of employee
     */
    public String getName() {
        return this.name;
    }

    /**
     * Sets the name of the Employee
     * @param name new full name
     */
    public void setName(String name) {
        this.name = name;
    }

    public double getSalary() {
        return this.salary;
    }

    public void setSalary(double salary) {
        this.salary = salary;
    }

    @Override
    public String toString() {
        return getClass().getName() + "[name=" + name + ", salary=" + salary + "]";
    }
}
```

```java
package employee;

import java.time.LocalDate;

public class Manager extends Employee {
    private LocalDate hireDate;

    /**
     * Creates a manager with a given name, salary and hiring date
     * @param name
     * @param salary
     * @param hireDate
     */
    public Manager(String name, double salary, LocalDate hireDate) {
        super(name, salary);
        this.hireDate = hireDate;
    }

    public LocalDate getHireDate() {
        return this.hireDate;
    }

    @Override
    public String toString() {
        return super.toString() + "[hireDate=" + hireDate + "]";
    }

}
```

## Access Modifiers

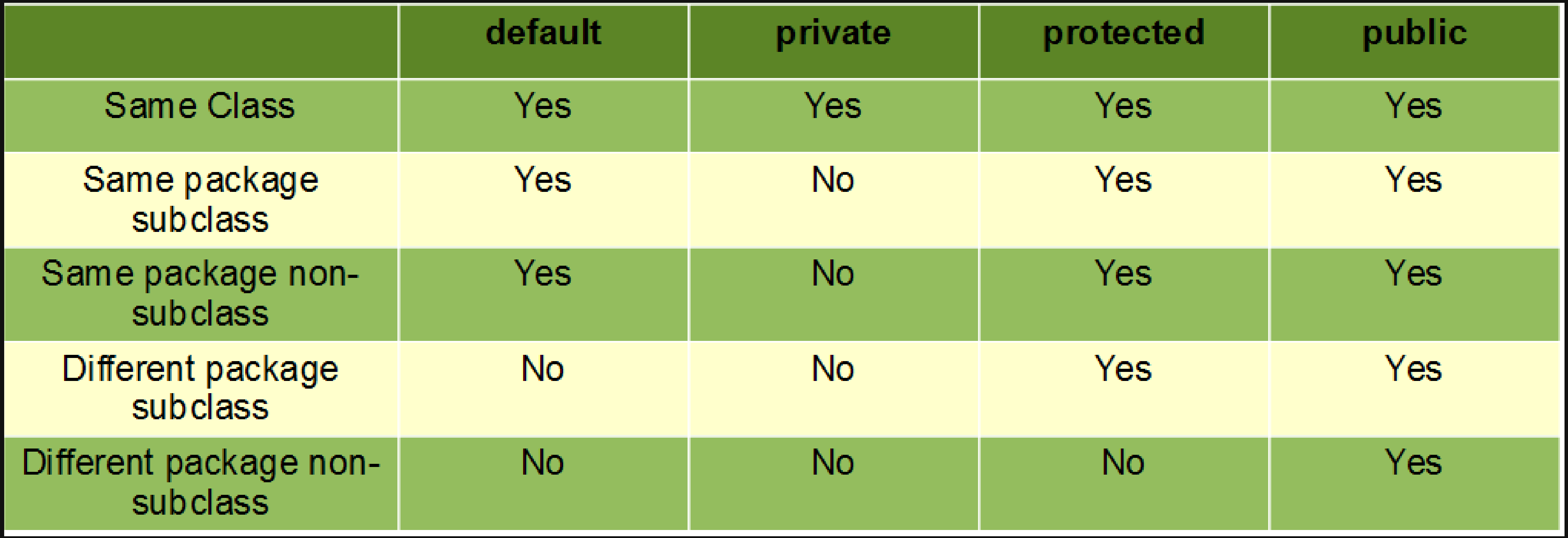
# Access Modifiers



| | default | private | protected | public |
|---|---|---|---|---|
| Same Class | Yes | Yes | Yes | Yes |
| Same package subclass | Yes | No | Yes | Yes |
| Same package non-subclass | Yes | No | Yes | Yes |
| Different package subclass | No | No | Yes | Yes |
| Different package non-subclass | No | No | No | Yes |

# Private

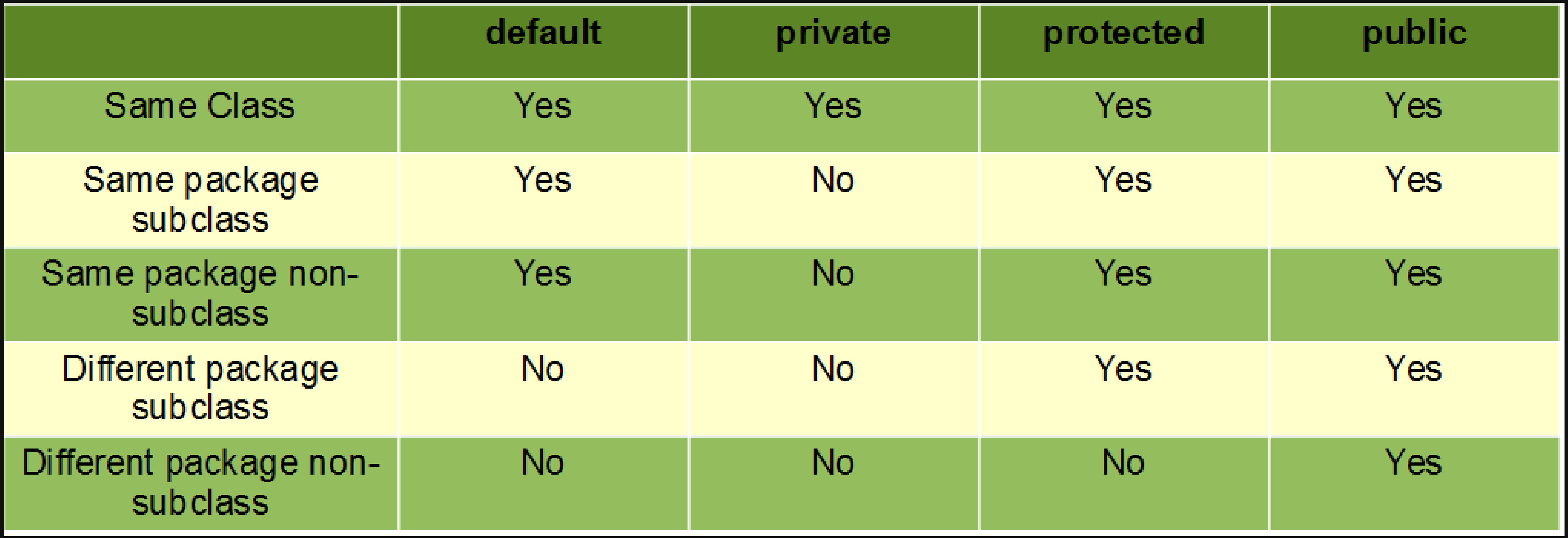
It is accessible only to the same class (not including main). The most restrictive modifier.



| | default | private | protected | public |
|---|---|---|---|---|
| Same Class | Yes | Yes | Yes | Yes |
| Same package subclass | Yes | No | Yes | Yes |
| Same package non-subclass | Yes | No | Yes | Yes |
| Different package subclass | No | No | Yes | Yes |
| Different package non-subclass | No | No | No | Yes |

# Public

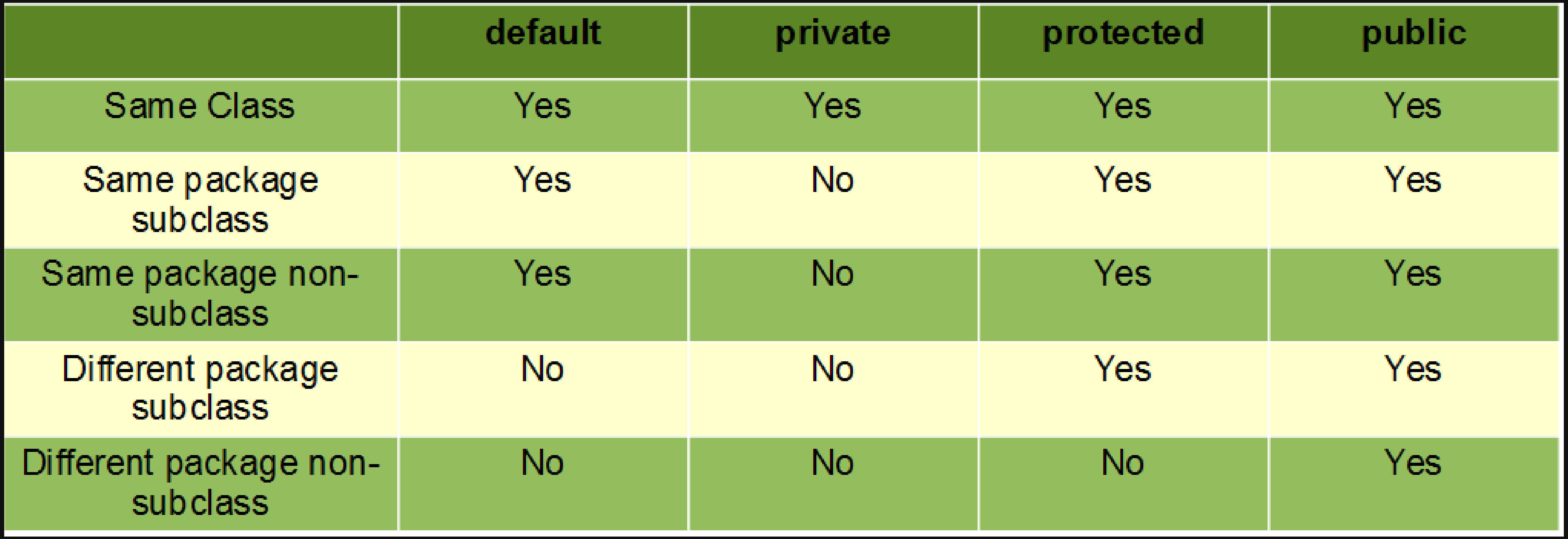
It is accessible to everything. The least restrictive modifier.



| | default | private | protected | public |
|---|---|---|---|---|
| Same Class | Yes | Yes | Yes | Yes |
| Same package subclass | Yes | No | Yes | Yes |
| Same package non-subclass | Yes | No | Yes | Yes |
| Different package subclass | No | No | Yes | Yes |
| Different package non-subclass | No | No | No | Yes |

# Protected

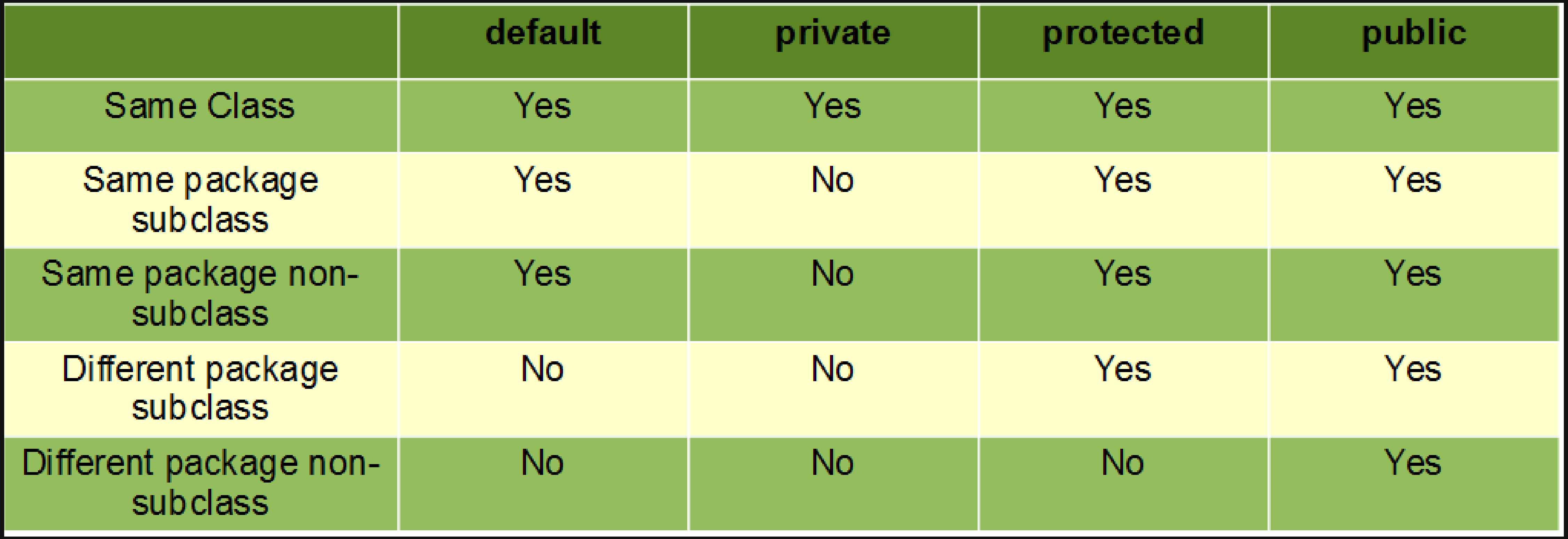
Can be accessed in the same package and in inheritance.



| | default | private | protected | public |
|---|---|---|---|---|
| Same Class | Yes | Yes | Yes | Yes |
| Same package subclass | Yes | No | Yes | Yes |
| Same package non-subclass | Yes | No | Yes | Yes |
| Different package subclass | No | No | Yes | Yes |
| Different package non-subclass | No | No | No | Yes |

# Default

The default access modifier is also called **package-private**, which means that all members are visible within the same package but aren't accessible from other packages



| | default | private | protected | public |
|---|---|---|---|---|
| Same Class | Yes | Yes | Yes | Yes |
| Same package subclass | Yes | No | Yes | Yes |
| Same package non-subclass | Yes | No | Yes | Yes |
| Different package subclass | No | No | Yes | Yes |
| Different package non-subclass | No | No | No | Yes |

## Attendance

## Feedback

https://forms.gle/R4sMTTQzPC4vqXSN8