Informatics 1: Object Oriented Programming

Tutorial 02

Week 3: Debugging

Volker Seeker (volker.seeker@ed.ac.uk)
Vidminas Vizgirda (s1750767@ed.ac.uk)

1 Introduction

In this tutorial, you will practice dealing with errors of all kinds in code. There will be some compilation errors, where the code does not produce a working program; runtime errors, where the program crashes while running; and logic errors, where the program runs fine, but its outputs are different to what was expected. The process of finding and fixing errors is called debugging.

The terms "bug" and "debugging" are often attributed to Admiral Grace Hopper in the 1940s. One day, while she was working at Harvard University, her associates discovered a moth stuck in a relay, which was causing it to not run properly. She remarked that they were "debugging" the system.

Side note: if you are interested in finding out more about the story of how computer errors came to be called bugs, check out the article "Stalking the elusive computer bug" by P. A. Kidwell.

Finding bugs can be a very challenging task, and some companies have bug bounty schemes where they reward people who inform them about bugs in their systems. If you find a way to break something, you might be able to get a prize for it, just make sure you are not doing anything illegal!

> ✎ You might encounter some code samples in this tutorial that you do not fully understand. Do not panic! This is intentional. Even expert programmers constantly face code they do not understand at work. Most of the time, understanding everything in a program is not necessary to be able to work with it. Instead, a good strategy is to focus on just the bits you need for your task and ignore the rest until you need it.

Figure 1: "Grace Hopper and UNIVAC" by Unknown (Smithsonian Institution) is licensed under CC BY 2.0 DEED.

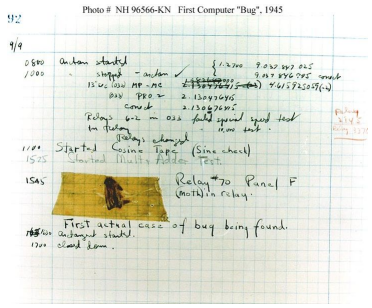

Figure 2: "First Computer Bug" by flanker27 is licensed under CC BY-NC-SA 2.0.

2 Bug hunt time

Task 1 - Program syntax

◁ Task

Syntax errors are a kind of compilation-time error. In Java, code with syntax errors does not produce a valid program at all. Discuss the below snippets with peers at your tables and try to spot what might be wrong with their syntax.

```
1 public class MyClass {
2     private static int ageToInt(String age) {
3         int parsed = Integer.parseInt(age);
4     }
5
6     public static void main(String[] args) {
7         System.out.println(ageToInt(args[0]));
8     }
9 }
```

The ageToInt function is missing a return statement. Attempting to compile/run this program will show a corresponding compiler error:

```
MyClass.java:4:5 java:  missing return statement
```

To fix this, you'd need to add `return parsed;` at the end of the `ageToInt` function.

```
1 public class MyClass {
2     private static void sayHappyBirthday(String name, int age) {
3         System.out.println("Happy " + age + "th birthday, dear " + name + "!");
4     }
5
6     public static void main(String[] args) {
7         sayHappyBirthday("Zhang Ming");
8     }
9 }
```

The function `sayHappyBirthday` has 2 parameters – `name` and `age`, but is given only one argument ("Zhang Ming") when called in `main`. The compiler shows the error as follows:

```
MyClass.java:7:9
java:  method sayHappyBirthday in class MyClass cannot be applied to given types;
required:  java.lang.String,int
found:  java.lang.String
reason:  actual and formal argument lists differ in length
```

To fix this, you'd need to add a second argument, e.g., 20, when calling `sayHappyBirthday`.

Task 2 - Let's put search engines to good use

◁ Task

Below are some common Java errors, that come up when running faulty code. Do you have any ideas about what might be causing them? What kind of errors are these (between compile-time errors, runtime errors, and logical errors)? Discuss with peers at your tables.

- Exception in thread "main" java.lang.ArithmeticException: / by zero
  at MyClass.main(MyClass.java:6)

- Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 3 out of bounds for length 3
  at MyClass.main(MyClass.java:5)
- Exception in thread "main" java.lang.NullPointerException
  at MyClass.main(MyClass.java:20)
- Exception in thread "main" java.lang.NumberFormatException: For input string: "Hello"
  at java.base/java.lang.NumberFormatException.forInputString(NumberFormatException.java:65)
  at java.base/java.lang.Integer.parseInt(Integer.java:652)
  at java.base/java.lang.Integer.parseInt(Integer.java:770)
  at MyClass.main(MyClass.java:4)

You can use the help of a search engine (Ecosia, DuckDuckGo, Google, Bing, whatever is your favourite) to find potential culprits.

Looking back at the introduction section, what kind of errors are these?

These are runtime exceptions, thrown during the program execution, which causes the program to crash.

Task 3 - The scapegoats

◁ **Task**

Now that you have some ideas in mind, here are the actual code pieces that are behind all this mess. Again on your tables, try to figure out which piece of code is causing which problem. Perhaps you can even point to the exact line, that is the troublemaker here?

```java
public class MyClass {
    public static void main(String[] args) {
        int[] myList = { 1, 2, 3 };
        int sum = myList[1] + myList[2] + myList[3];
        System.out.println("1 + 2 + 3 = " + sum);
    }
}
```

```java
public class MyClass {
    public static void main(String[] args) {
        if (args.length == 2) {
            int a = Integer.parseInt(args[0]);
            int b = Integer.parseInt(args[1]);
            System.out.println("a + b = " + (a + b));
        } else {
            System.out.println("give me 2 arguments, pretty please :3");
        }
    }
}
```

```java
public class MyClass {
    public static void main(String[] args) {
        System.out.println("30 / 9 = " + (30 / 0));
    }
}
```

```java
// Imagine this is in file Car.java
public class Car {
```

```
    int wheels;

    public Car() {
        this.wheels = 4;
    }

    public Car(int wheels) {
        if (wheels < 3) {
            System.out.println("Are you sure your car is okay?");
        }
        this.wheels = wheels;
    }
}

// Imagine this is in file MyClass.java
public class MyClass {
    public static void main(String[] args) {
        Car myCar;
        System.out.println("My car has " + myCar.wheels + " wheels :)");
    }
}
```

The first example creates an ArrayIndexOutOfBoundsException. This emphasises a common mistake of forgetting that list indices start at 0, not 1.

The second example produces a NumberFormatException, but only if 2 arguments are passed in and they are not integers. This might be a good point to talk about representing a number as a String and as an Integer type and how these are different.

The third example might have been produced by a typo - it tries division by 0, but the output line implies that the 0 was meant to be a 9.

Finally, the last example throws a NullPointerException, because myCar is not initialised. It is important not to forget to call the constructor with the "new" keyword.

The code snippets after debugging look as follows:

```
public class MyClass {
    public static void main(String[] args) {
        int[] myList = { 1, 2, 3 };
        int sum = myList[0] + myList[1] + myList[2];
        System.out.println("1 + 2 + 3 = " + sum);
    }
}
```

```java
public class MyClass {
    public static void main(String[] args) {
        if (args.length == 2) {
            try {
                int a = Integer.parseInt(args[0]);
                int b = Integer.parseInt(args[1]);
                System.out.println("a + b = " + (a + b));
            } catch (NumberFormatException e) {
                System.out.println("integers only, please");
            }
        } else {
            System.out.println("give me 2 arguments, pretty please :3");
        }
    }
}
```

```java
public class MyClass {
    public static void main(String[] args) {
        System.out.println("30 / 9 = " + (30 / 9));
    }
}
```

```java
// Imagine this is in file Car.java
public class Car {
    int wheels;

    public Car() {
        this.wheels = 4;
    }

    public Car(int wheels) {
        if (wheels < 3) {
            System.out.println("Are you sure your car is okay?");
        }
        this.wheels = wheels;
    }
}

// Imagine this is in file MyClass.java
public class MyClass {
    public static void main(String[] args) {
        Car myCar = new Car();
        System.out.println("My car has " + myCar.wheels + " wheels :)");
    }
}
```

3 Exercises

Imagine you are now an elite freelance hacker. MilkySoftwareWay Inc have called you in for your expertise in code - last night a black hat (criminal) cracker broke into their systems and pulled a prank: now different modules from their system no longer produce correct behaviour!

The in-house software developers in MilkySoftwareWay are at a loss about what changed... Can you help

them identify and fix the issues?

Task 1 - (In)secure program

◁ Task

The first and most important module in MilkySoftwareWay's systems is the authorisation module.

It uses a Java Scanner which you may not have seen before. Don't worry about the details of getting user input - the software engineers are sure that this part of the code has not been changed. The real problem must lie elsewhere.

```java
import java.util.Scanner; // Import the Scanner class

public class MyClass {
  public static void main(String[] args) {
    Scanner input = new Scanner(System.in); // Create a Scanner object
    System.out.println("Enter username");

    String userName = input.nextLine(); // Read user input
    input.close();
    System.out.println("Username is: " + userName); // Output user input

    if (userName == "admin") {
        System.out.println("You have full access to everything!");
    } else if (userName == "HRadmin") {
        System.out.println("You have access to all employee records");
    } else {
        System.out.println("You don't have access to anything");
    }
  }
}
```

This is a table that MilkySoftwareWay's developers have created, showing what output this code produces, when given a specific input, and what they would expect the output to be.

Discuss in groups at your table to see if you can find what the issue is and how it could be fixed.

Input	Output	Expected Output
test	You don't have access to anything	You don't have access to anything
Rachael Williams	You don't have access to anything	You don't have access to anything
Dolan Prumt	You don't have access to anything	You don't have access to anything
HRadmin	You don't have access to anything	You have access to all employee records
admin	You don't have access to anything	You have full access to everything!

This task demonstrates a very common beginner mistake of forgetting to compare strings in Java using the .equals() method. Using == actually compares the strings' memory addresses which aren't the same for the user input string and the string literals in the program.

Task 2 - Index out of bounds again?!

◁ Task

The following module prints out all the first names and roles of members of one team at MilkySoftwareWay. But now, it seems to crash after printing them all, which shouldn't be happening. What went wrong?

```java
public class MyClass {
    public static void main(String[] args) {
        String[] employeeRoles = { "Tyrone, Research Software Engineer", "Elizabeth, Researcher",
```

```
            "Wednesday, Engineering Manager" };

        for (int i = 0; i <= employeeRoles.length; ++i) {
            System.out.println("Employee " + i + ": " + employeeRoles[i]);
        }
    }
}
```

Output:

```
Employee 0: Tyrone, Research Software Engineer
Employee 1: Elizabeth, Scientist
Employee 2: Wednesday, Engineering Manager
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 3 out
 of bounds for length 3
        at MyClass.main(MyClass.java:6)
```

In this task, there are 3 strings in the employeeRoles array, and therefore, employeeRoles.length is 3. The for loop attempts to run 4 times, with i values 0, 1, 2, 3 and the last one causes an ArrayIndexOutOfBoundsException. This is the same issue as one of the previous examples, just disguised. The solution is to use `i < employeeRoles.length` instead of `i <= employeeRoles.length` in the loop condition.

Task 3 - It's a trap!

◁ **Task**

Another affected module is terrible news for MilkySoftwareWay - their main user interface menu no longer allows users to exit the program properly!

This is definitely going to cost the company some customers, can you help them identify the issue to resolve this?

```
import java.util.Scanner; // Import the Scanner class

public class MyClass {
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    int option = scanner.nextInt(); // Get an integer from user input (stdin)
    scanner.close();

    if (option < 1) {
        System.out.println("You chose doom");
    } if (option < 2) {
        System.out.println("You chose heaven");
    } if (option < 3) {
        System.out.println("You chose earth");
    } else {
        System.out.println("You chose space");
    }
  }
}
```

Input	Output	Expected Output
4	`You chose space`	`You chose space`
3	`You chose space`	`You chose space`
2	`You chose earth`	`You chose earth`
1	`You chose heaven` `You chose earth`	`You chose heaven`

0	`You chose doom`
	`You chose heaven`
	`You chose earth`
		`You chose doom`

This example showcases an easy oversight to make - the if statements in the middle should be "else if" instead, otherwise all of them are executed if their conditions are met.

Task 4 - Taking matters into your own hands

◁ **Task**

Whew! We are almost there. In this last affected module, a critical business decision making system has broken down.

Can you find what is the culprit?

```java
import java.util.Scanner;

public class MyClass {
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    int number = scanner.nextInt();
    scanner.close();

    boolean choice = number == 1;

    if (choice = true) {
        System.out.println("Choose 1");
    } else {
        System.out.println("Choose other");
    }
  }
}
```

MilkySoftwareWay's engineers have not had time to test this program yet - but at least they came up with some potential inputs that would be a good idea to try.

First look at the code and think of what output you would expect for each of the inputs and write them in the table below.

Next try running this program in a Java IDE (you can use `https://www.jdoodle.com/online-java-compiler/`) and see whether its actual outputs match up to your expected outputs.

Input	Output	Expected Output
-1		
0		
1		
x		
hello		

Here the mistake in spotlight is using an assignment = instead of a comparison == in "if (choice = true)".

Since this statement sets choice to true, for all numeric inputs, the program will print "Choose 1". However, for the last two inputs "x" and "hello" the program will throw an InputMismatchException, which is there to lead the students into beginning to think about handling bad inputs safely.

Hooray! The MilkySoftwareWay troubles have come to an end. Now it's time for the tutors to clarify and elaborate these code pranks.

They are actually very common mistakes in code, and as you may have experienced, they can be tricky to spot! Knowing exactly what to watch out for can really help.

Pair debugging

Find yourself another elite hacker partner - it is time for another job but this time you will be working in pairs!

A company Appdroid do their business by making apps that change the world. Currently, they are working on an app called "Sudoku World".

Appdroid have received numerous bug reports from multiple customers about "Sudoku World" since the recent release of version 1.9.

Having heard about your expertise from MilkySoftwareWay, they have decided to employ you to help track down and solve these issues.

Figure 3: "Time to start hacking!" generated with Bing Image Creator (DALL-E3)

Your job is to investigate the code and bug reports provided by Appdroid, find the causes of the issues and fix them. Appdroid have also provided you some guidelines about how to get started with the project in the "readme" file.

Since Appdroid are on a tight deadline to release "Sudoku World" 2.0 you will have only half an hour to finish this work. You may find that splitting up the work of looking at the code and the bug reports with your partner can allow you to work more effectively.

✎ You can download the required source files from the Inf1B course materials page: `https://course.inf.ed.ac.uk/inf1b`

The first thing to do is to extract the downloaded zip file somewhere. Then use IntelliJ's 'Open' menu, navigate to the extracted project folder, and open it.

Since this is the first time you are given an entire project and not just an individual file, you may encounter some new difficulties. A common one is when a project is created using a different JDK than what you have installed – but do not fret – this is really easy to fix. Java is a backwards compatible language, meaning a newer version of Java can run programs written for an older version of Java (although it does not work the other way round).

If you see a blue error strip at the top that says "JDK ... is missing", do *not* select the Download option – you don't need yet another JDK installed just because someone else used this version. Instead hit "Configure..." and select the JDK version you have installed on your system.

The screenshots below illustrate this issue and how to resolve it:

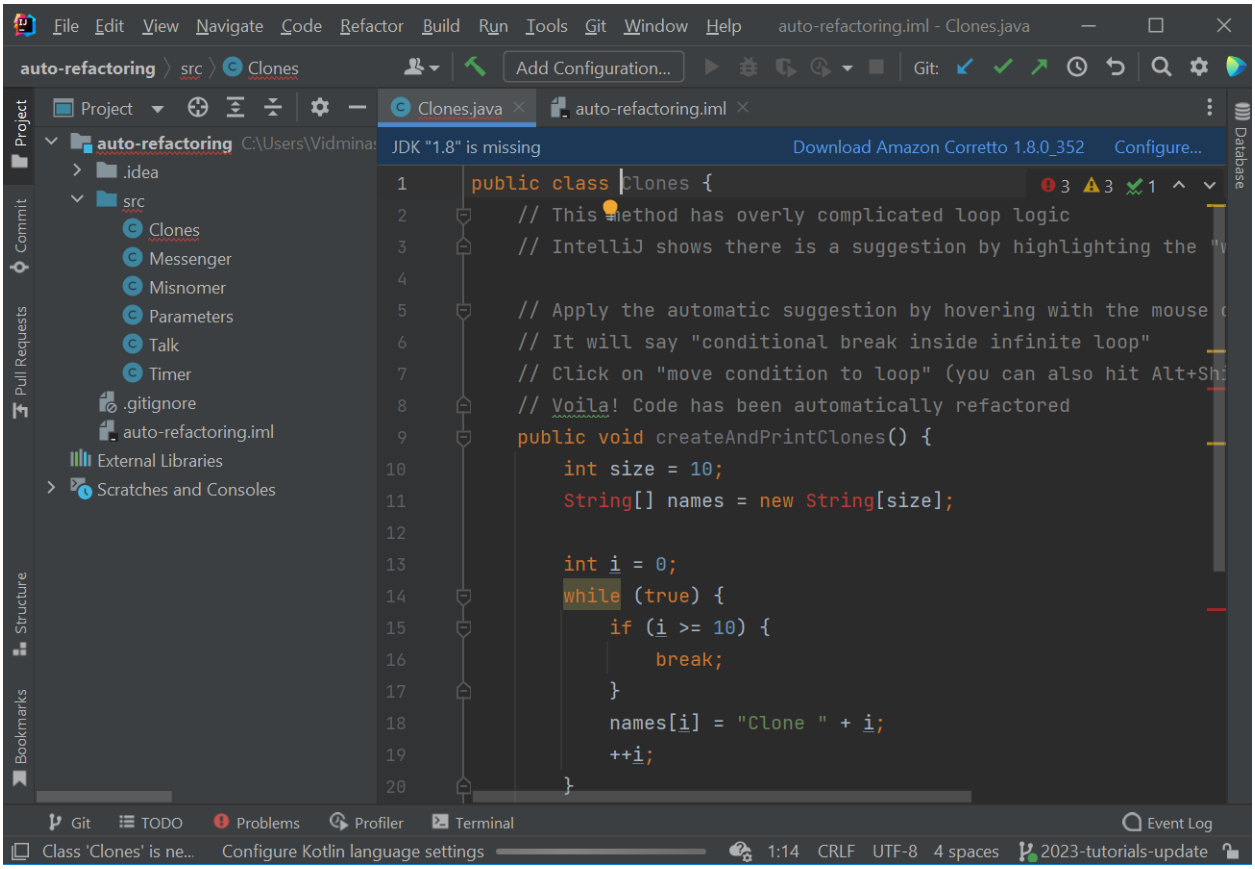

Figure 4: Example project (not the one we'll be using) with a JDK missing error

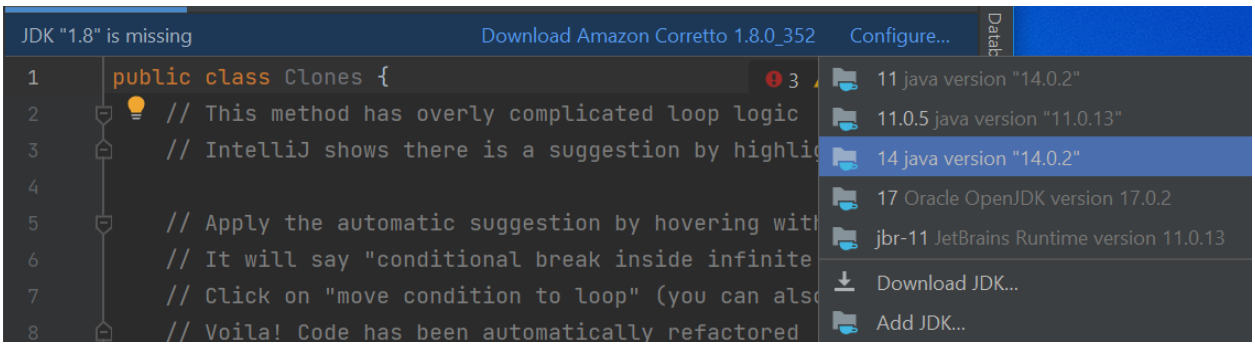

Figure 5: The configure context menu, where any of the options 11, 14, or 17 would work

If IntelliJ says "Module JDK is not defined", the fix is basically the same, just select a JDK version from the dropdown that appears after clicking "Setup SDK".

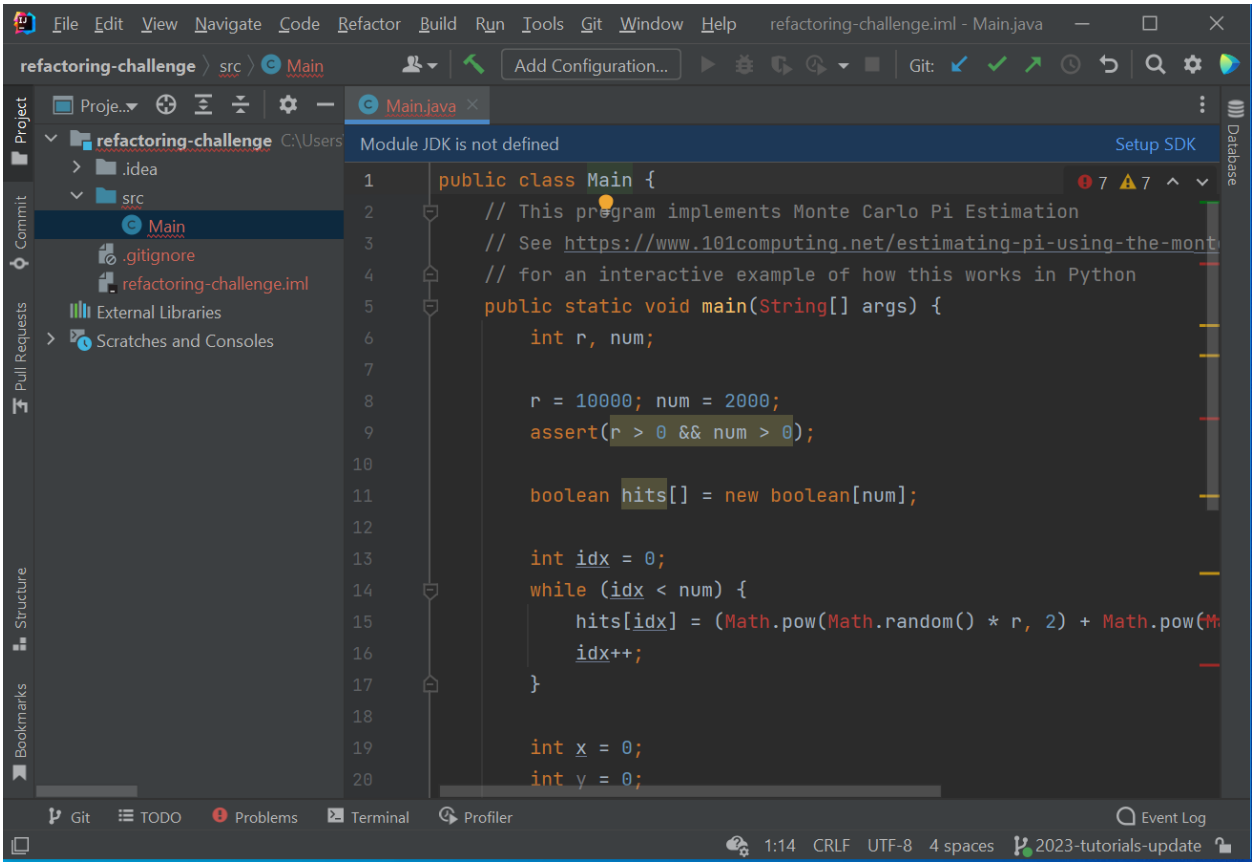

Figure 6: Another example project where the module JDK is not defined. Click 'Setup JDK' and choose a JDK version from the dropdown to resolve this.

If you accidentally select the wrong version or decide to change it later, you can do this from the project structure settings. Open `File > Project Structure...` and you can set the JDK from there.

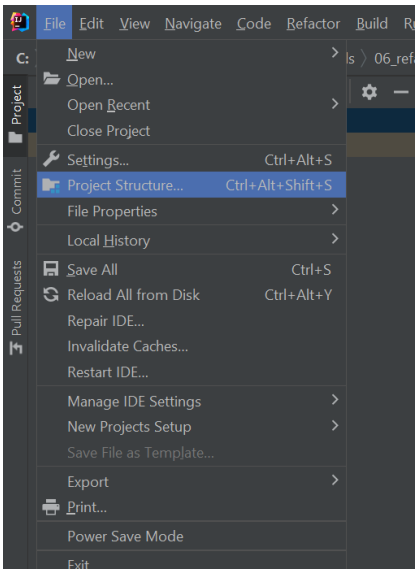

Figure 7: Another example project where the module JDK is not defined. Click 'Setup JDK' and choose a JDK version from the dropdown to resolve this.

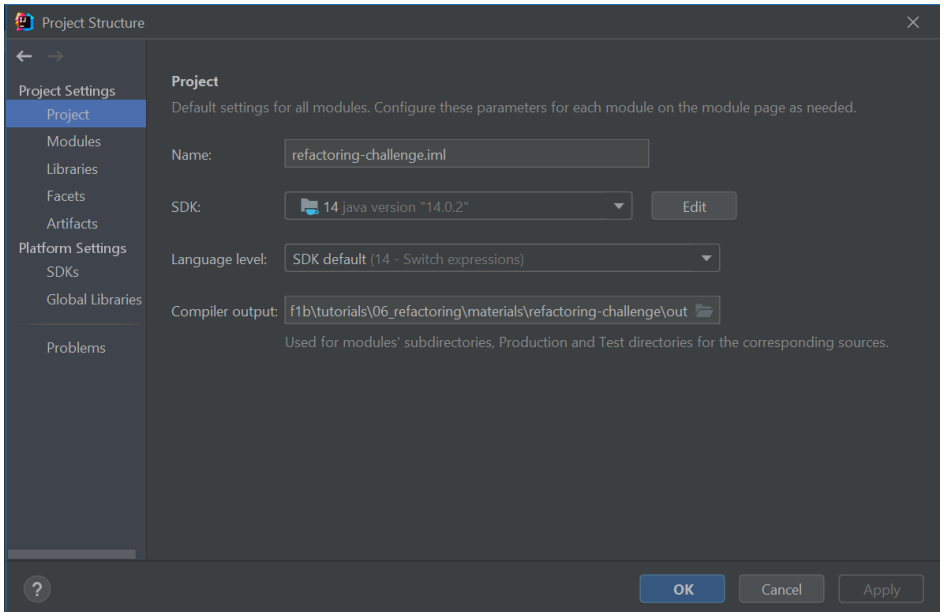

Figure 8: Where to find the Project Structure settings

Another issue that could come up is that the project configuration might get somehow lost, for example, if the project was renamed outside of IntelliJ. This could make the project open blank. If this happens, find the project structure settings again, but this time go to the "Modules" tab. There you need to click 'Add Content Root' and navigate it to the same project folder. This should make all the project files appear again.

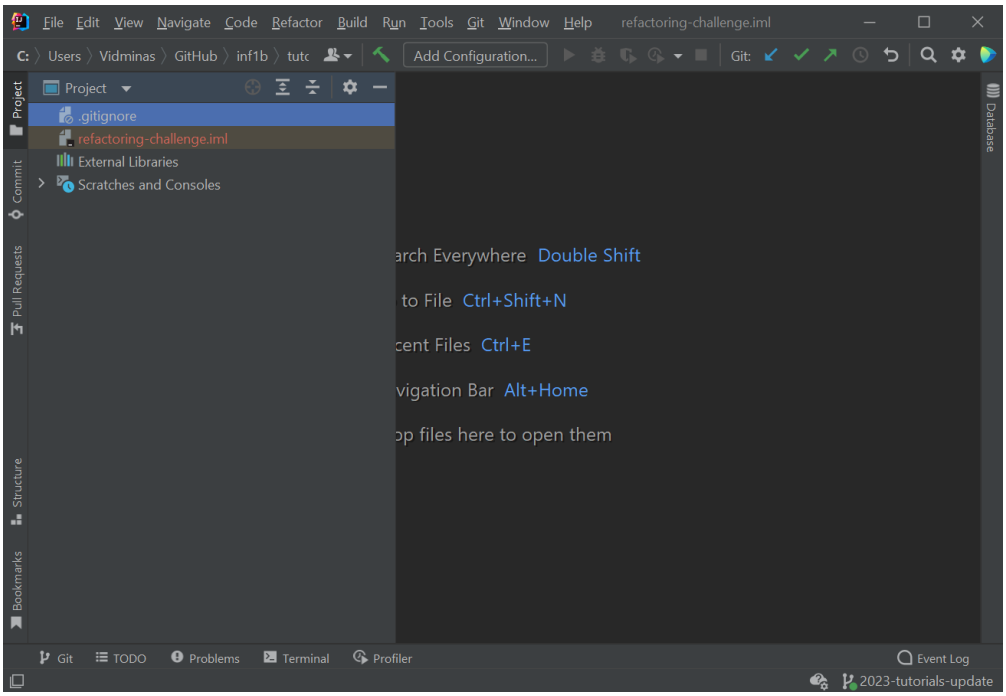

Figure 9: Another project that opens up blank, only .gitignore and the project .iml files are there, but the 'src' directory with the Java code does not show up...

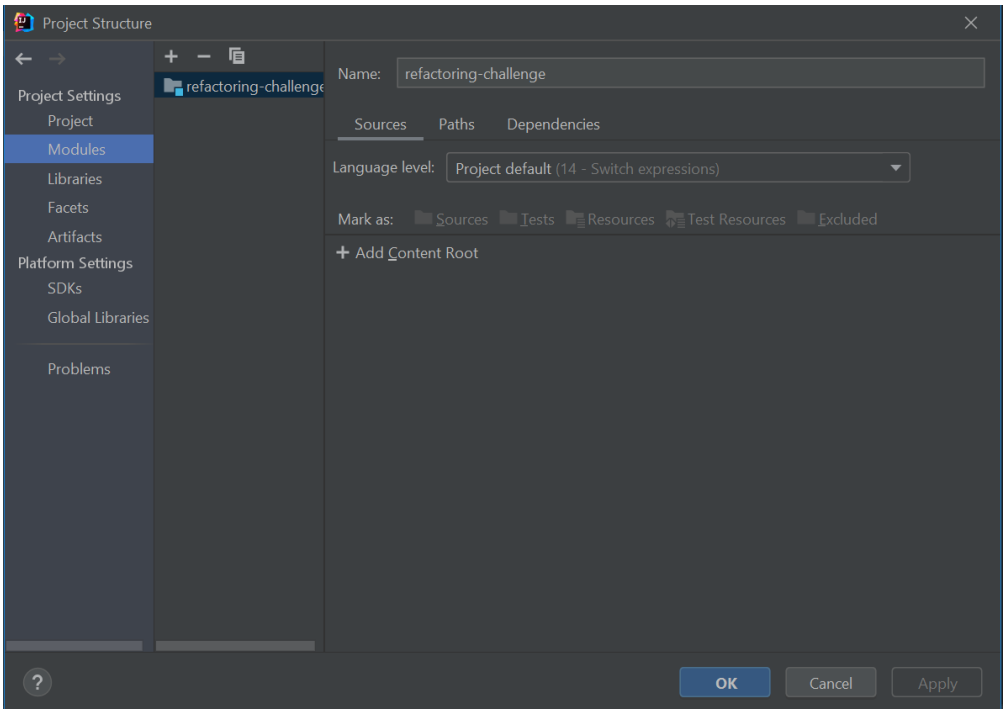

Figure 10: To fix blank projects, go to the Project Structure and find the Modules tab on the left

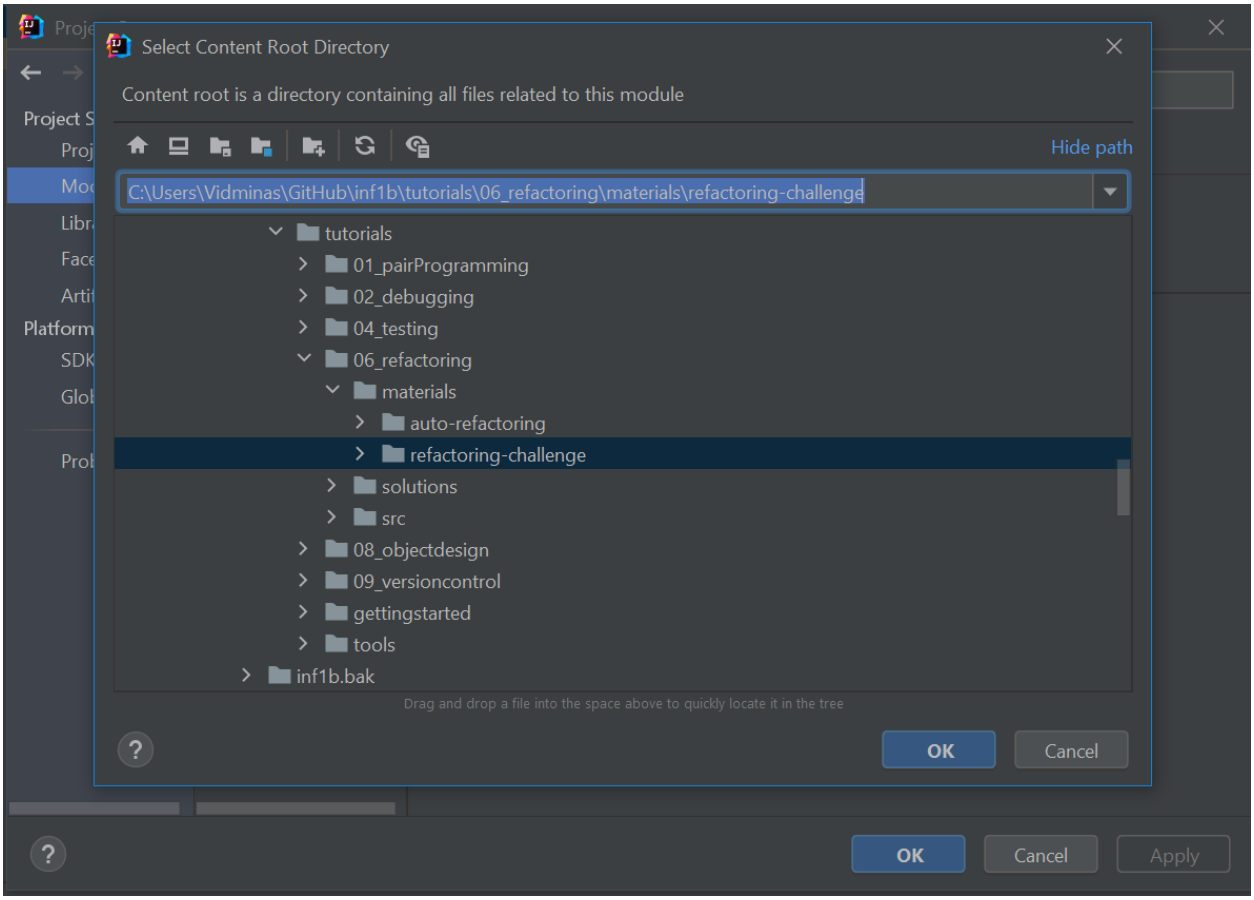

Figure 11: After clicking 'Add Content Root', select the project folder and press OK. This should make the files appear again.

✎ Solutions for this task are also provided on the Inf1B course materials page.

Sources: https://www.codejava.net/coding/10-common-mistakes-every-beginner-java-programmer-makes