Informatics 1: Object Oriented Programming

Tutorial 02

Week 4: Debugging 03/02 - 07/02

Volker Seeker (`volker.seeker@ed.ac.uk`)
Vidminas Mikucionis (`vmikucio@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. This process 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 win from 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. This is ok and intentional. You will face similar situations more often then you would think as a developer and you still have to manage working with it.

[1] Image source: `https://www.thisiscolossal.com/wp-content/uploads/2016/06/bug-5.jpg`

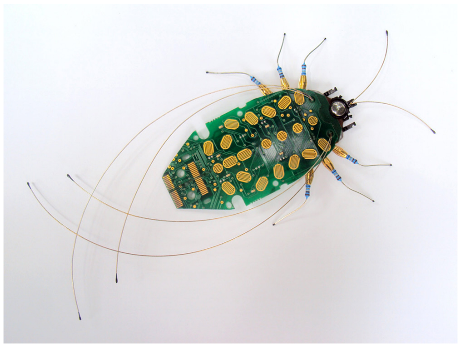

Figure 1: Hardwired bug[1]

2 Bug hunt time

Task 1 - Let's put search engines to good use

◁ **Task**

Suppose Tyrone was developing software for Sum Macrosystems, who needed some innovative Java solutions. Unfortunately, like every programmer, Tyrone ran into lots of errors while writing code. Here are some of the outputs he got. Do you have any ideas about what might be causing them? 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?

Task 2 - 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
1 public class MyClass {
2     public static void main(String[] args) {
3         int[] myList = { 1, 2, 3 };
4         int sum = myList[1] + myList[2] + myList[3];
5         System.out.println("1 + 2 + 3 = " + sum);
6     }
7 }
```

```java
1 public class MyClass {
2     public static void main(String[] args) {
3         if (args.length == 2) {
4             int a = Integer.parseInt(args[0]);
5             int b = Integer.parseInt(args[1]);
6             System.out.println("a + b = " + (a + b));
7         } else {
8             System.out.println("give me 2 arguments, pretty please :3");
9         }
```

```
    }
}
```

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

```
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;
    }
}

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

Task 3 - Time to save the day

◁ **Task**

Figure 2: Save the World[2]

Now, in groups, see if you can help Tyrone save his job and fix these issues in the code. If you don't have a Java IDE, you can use this online tool: https://www.jdoodle.com/online-java-compiler/

[2]Image source: https://images-na.ssl-images-amazon.com/images/I/41mhjyz73fL._SX342_.jpg

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!

Task 2 - 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?

```java
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`

Task 3 - 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		

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.

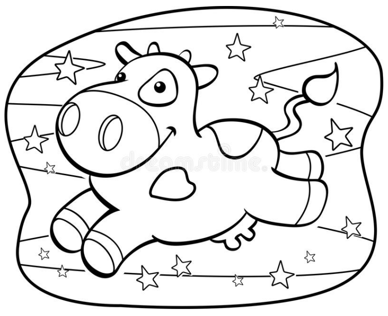

Figure 3: MilkySoftwareWay[3]

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.

[3]Image source: https://thumbs.dreamstime.com/b/chocolate-cow-happy-cartoon-milky-way-47328690.jpg

Figure 4: Time to start hacking![4]

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 Learn page: `https://course.inf.ed.ac.uk/inf1b`

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

[4]Image source: `https://i.pinimg.com/736x/b8/b0/e6/b8b0e67e49bc4965a8fe4e6760a8468c.jpg`