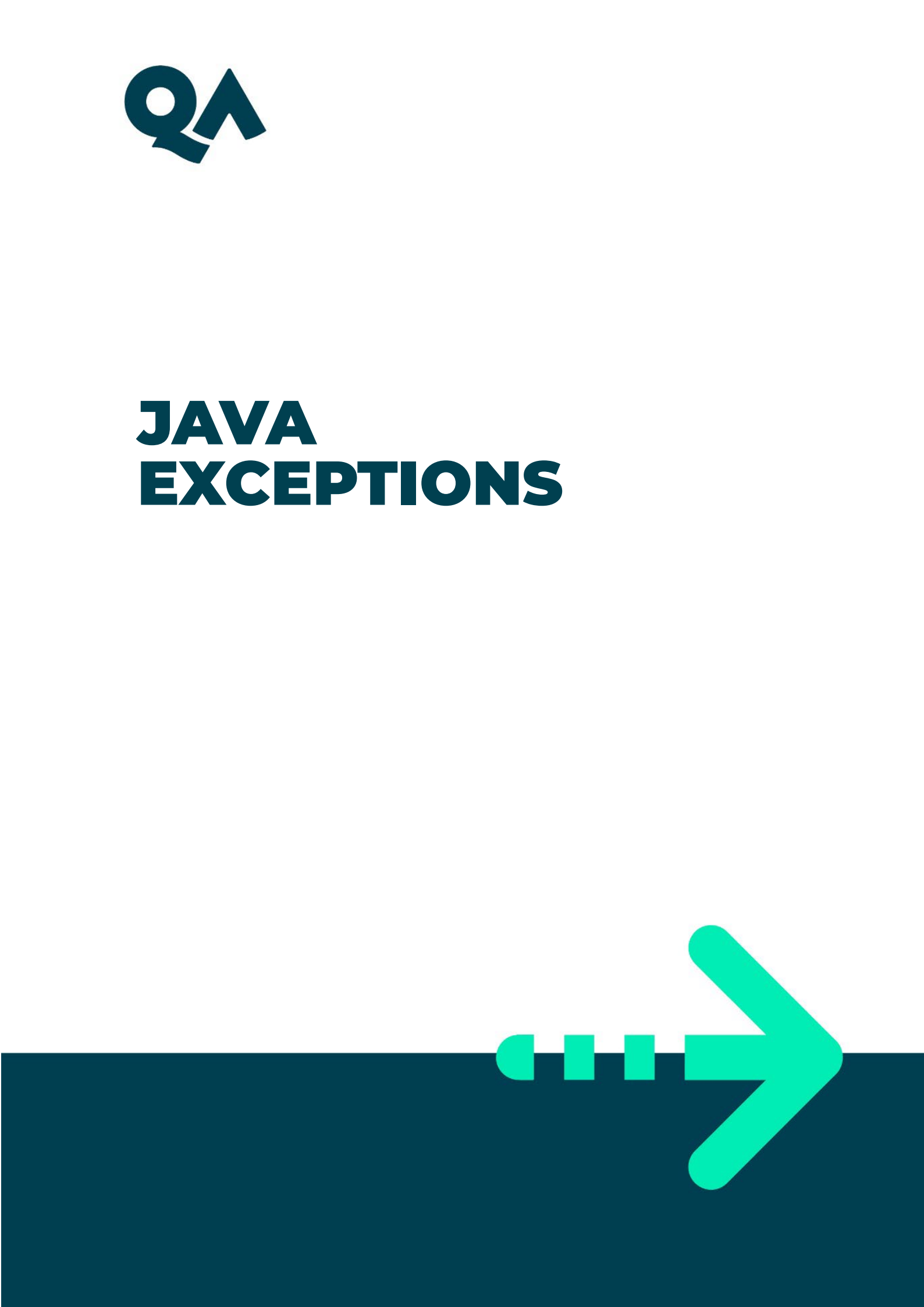
JAVA
EXCEPTIONS

Objective

The primary objectives for this lab are to be able to throw and catch exceptions and have a full understanding of the Java runtime's propagation mechanism for exceptions until a handler is found.

Overview

Firstly, you will handle the exceptions by throwing an exception object in your Account class if balance goes into red.

In a second practical you will enhance the MotorwaySimulator to throw an exception when the RegistrationFactory runs out of registration plates.

In the third part of this lab, you'll handle Java's checked exceptions.

Part 1

Using try/throw/catch in the Account class

1. Locate the **Account** class you created earlier (exercise2).
2. Add a **close()** method to this class which displays a simple message.
3. Back in the main() method create an instance of Account with a balance of £100.
4. Withdraw £50 and then display the account's details using the getDetails() method.
5. Withdraw £600 and display the account's details.
   Please note there no withdrawal limit in our friendly bank!
6. Fix the withdrawal() method so that it throws an `illegalArgumentException` when the balance becomes negative.
7. Catch this exception in main() and display a suitable user-friendly message.
8. Build and run the program now. Make sure you get the Account class to throw the exception and make sure it is handled in main().
9. Add a **finally** clause to the exception handler in main() to close the account no matter if there is an exception or not.

Part 2

10. Let's get back in the motorway simulator lab which you created earlier.
11. The factory class will at some time run out of registration plates.
    How would you handle this situation?
    How would you inform the Vehicle class that we've ran out of reg plates?

    How would the Vehicle class inform the caller we cannot create a vehicle?

    Implement the required exception mechanisms the best you can.

Part 3

Let's use this part to practise exception handling and remind ourselves of how to write text (like an error message) to a file.

12. Create a method in the Program class to write a message to a log file.
13. Type the following code:

```
private static void log(String msg){
        File file = new File("log.txt");
        FileWriter fr = new FileWriter(file, true);
        BufferedWriter br = new BufferedWriter(fr);
        br.write(msg + "\r\n");
        br.close();
        fr.close();
}
```

14. Call the log method from main()
15. Your code will not compile unless you change the signature of the log() method. It should indicate that it throws IOException.
16. Catch this exception in main() and display a IOException's message, should the log() method fail.