Informatics 1: Object Oriented Programming

Getting Started

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

Before you start writing your own code, you will need to make sure that you are comfortable with the basics of the development tools that we will be using. This note describes how to download and run a very simple Java example using these tools. This will check that you have everything correctly installed. You will then be able to create and run a simple program using IntelliJ.

1 Development Tools

Task 1 - Read the note which describes the Development Tools ◁ Task

If you intend to use your own machine, you will need to download and install these tools. If you only intend to use the Informatics DICE machines, then this software is already installed - but you should still read the note so that you understand the tools that you will be using, and any necessary configuration.

2 Using IntelliJ

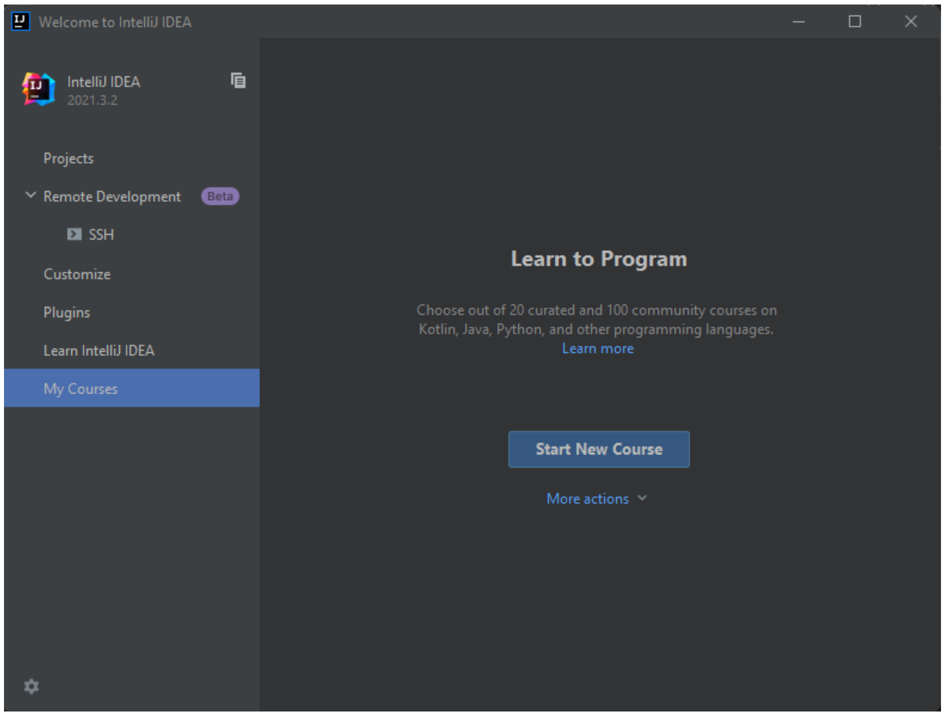
The instructions below will walk through getting started step-by-step and some screenshots illustrating the steps are included further below.

Task 2 - Open the 'Java Programming Basics' course in IntelliJ ◁ Task

From the IntelliJ welcome menu, if the *JetBrains Academy* plugin is correctly installed, you will be able to select 'My Courses'.

If you have previously already opened a project, the welcome menu won't appear when opening IntelliJ, but you can get back to it by closing the current project by selecting File > Close Project.

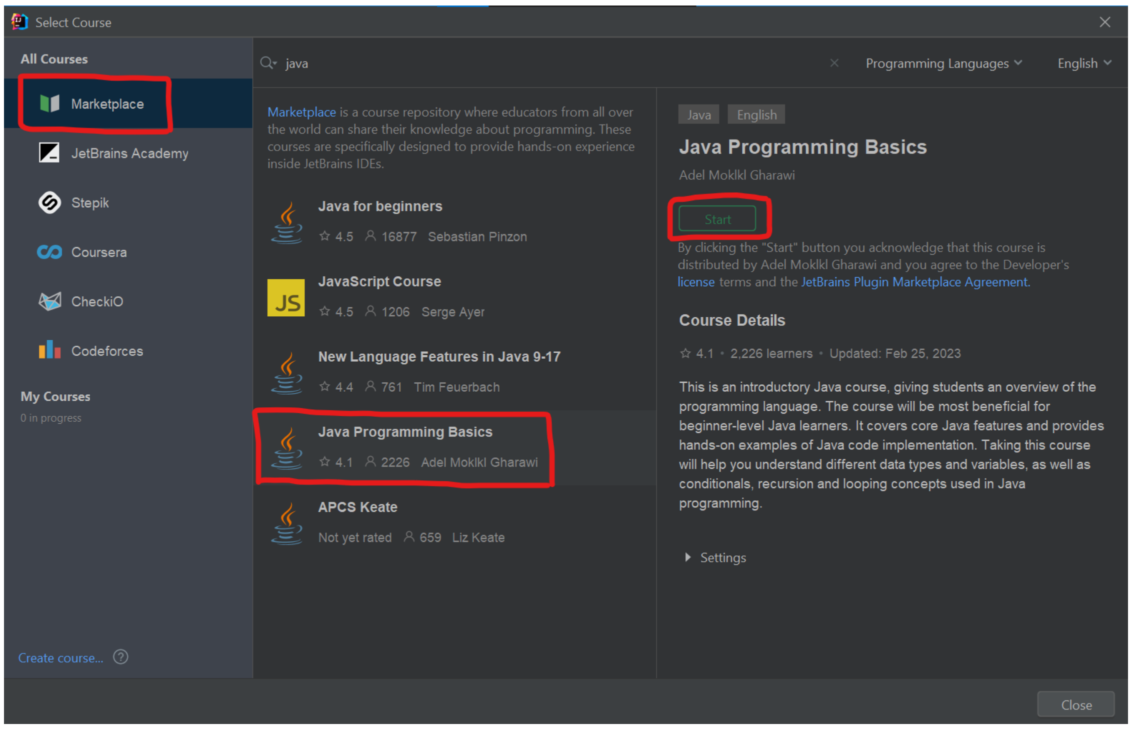
From here, click Start New Course and find the 'Java Programming Basics' course under the 'Marketplace' tab. You may need to use the search bar to find it, e.g., by typing "Java". If prompted whether to trust and open the project, it should be safe to click Trust Project and proceed.



Learn to Program

Choose out of 20 curated and 100 community courses on Kotlin, Java, Python, and other programming languages.
Learn more

Figure 1: IntelliJ welcome menu



Marketplace is a course repository where educators from all over the world can share their knowledge about programming. These courses are specifically designed to provide hands-on experience inside JetBrains IDEs.

Java for beginners
☆ 4.5 16877 Sebastian Pinzon

JavaScript Course
☆ 4.5 1206 Serge Ayer

New Language Features in Java 9-17
☆ 4.4 761 Tim Feuerbach

Java Programming Basics
☆ 4.1 2226 Adel Moklkl Gharawi

APCS Keate
Not yet rated 659 Liz Keate

Java Programming Basics
Adel Moklkl Gharawi

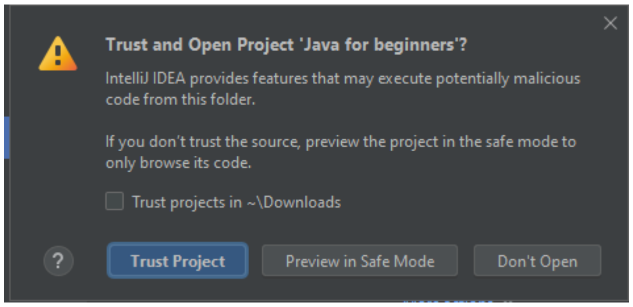
By clicking the "Start" button you acknowledge that this course is distributed by Adel Moklkl Gharawi and you agree to the Developer's license terms and the JetBrains Plugin Marketplace Agreement.

Course Details

☆ 4.1 • 2,226 learners • Updated: Feb 25, 2023

This is an introductory Java course, giving students an overview of the programming language. The course will be most beneficial for beginner-level Java learners. It covers core Java features and provides hands-on examples of Java code implementation. Taking this course will help you understand different data types and variables, as well as conditionals, recursion and looping concepts used in Java programming.

Figure 2: IntelliJ course selection


Trust and Open Project 'Java for beginners'?

IntelliJ IDEA provides features that may execute potentially malicious code from this folder.

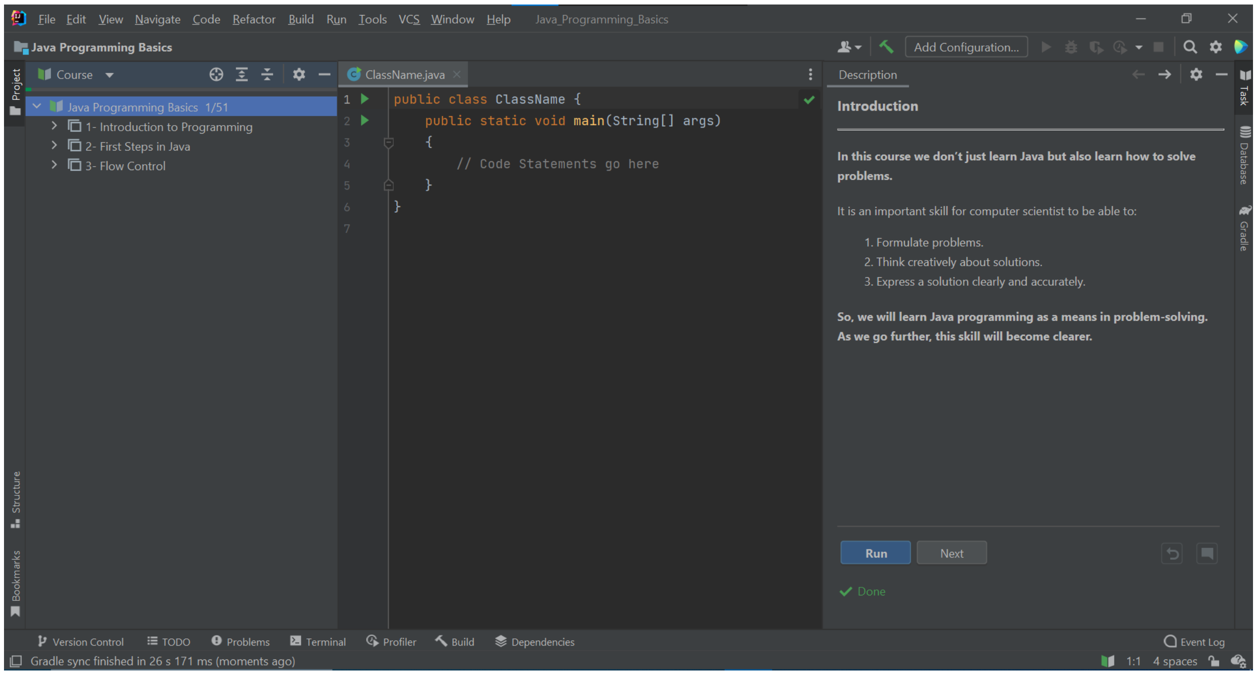
If you don't trust the source, preview the project in the safe mode to only browse its code.

Figure 3: IntelliJ trust project dialog

Task 3 - Write a small test program

◁ **Task**

When the course opens, you will see several panels. On the left is the project file explorer, it shows all the project files. We won't neeed this for now. In the middle is the text editor – it will start with one file open, which has a class with a main method with some comments and nothing else. On the right are some course notes.



```
public class ClassName {
    public static void main(String[] args)
    {
        // Code Statements go here
    }
}
```

Introduction

In this course we don't just learn Java but also learn how to solve problems.

It is an important skill for computer scientist to be able to:

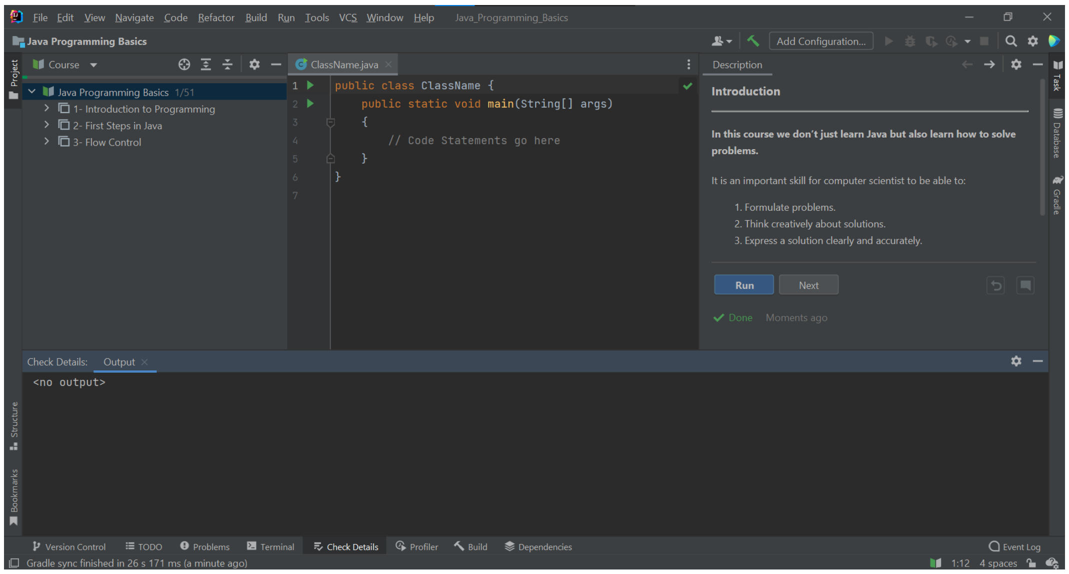
1. Formulate problems.
2. Think creatively about solutions.
3. Express a solution clearly and accurately.

So, we will learn Java programming as a means in problem-solving. As we go further, this skill will become clearer.

Gradle sync finished in 26 s 171 ms (moments ago)

Figure 4: IntelliJ course view

If you click Run in the course notes panel, the code you have open will be compiled and executed. A new view on the bottom of the screen will open, showing the output of your program. You can do this with the code as it is, but it will have no output.



File Edit View Navigate Code Refactor Build Run Tools VCS Window Help Java_Programming_Basics
Java Programming Basics
Add Configuration...
Course
Java Programming Basics 1/51
1- Introduction to Programming
2- First Steps in Java
3- Flow Control
ClassName.java
Description
Run
Next
Done Moments ago
Check Details: Output
Version Control TODO Problems Terminal Check Details Profiler Build Dependencies Event Log
Gradle sync finished in 26 s 171 ms (a minute ago)
1:12 4 spaces

```java
public class ClassName {
    public static void main(String[] args)
    {
        // Code Statements go here
    }
}
```

Introduction

In this course we don't just learn Java but also learn how to solve problems.

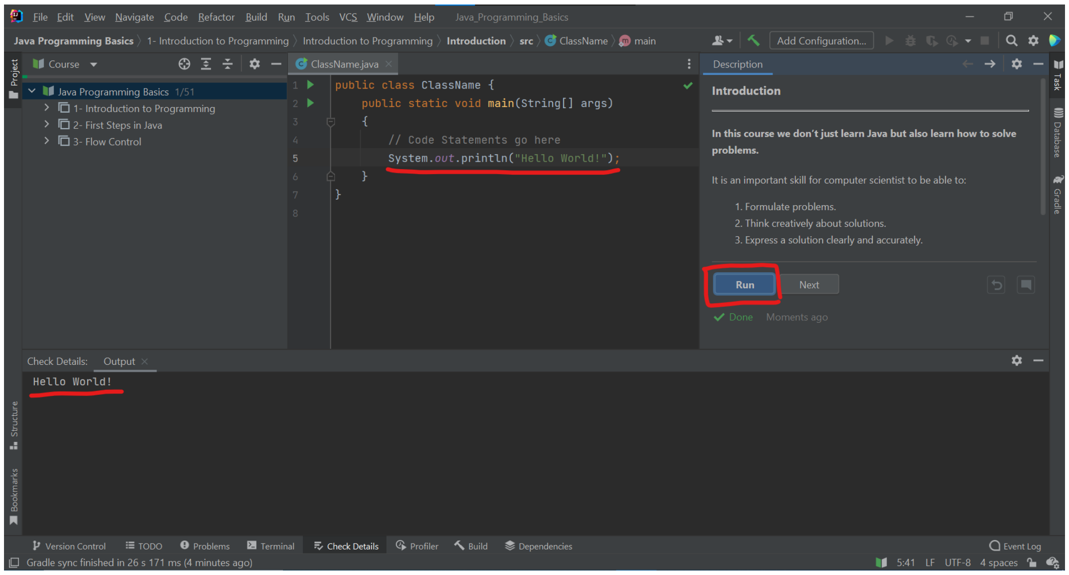
It is an important skill for computer scientist to be able to:

1. Formulate problems.
2. Think creatively about solutions.
3. Express a solution clearly and accurately.

```
<no output>
```

Figure 5: Blank program output screen

Traditionally, the first program that you write in a new language should just print the message "Hello World"[1]. Add the line `System.out.println("Hello World!");` to the main method and click Run. Check that the output says "Hello World!".



File Edit View Navigate Code Refactor Build Run Tools VCS Window Help Java_Programming_Basics
Java Programming Basics 〉 1- Introduction to Programming 〉 Introduction to Programming 〉 Introduction 〉 src 〉 ClassName 〉 main
Add Configuration...
Course
Java Programming Basics 1/51
1- Introduction to Programming
2- First Steps in Java
3- Flow Control
ClassName.java
Description
Run
Next
Done Moments ago
Check Details: Output
Version Control TODO Problems Terminal Check Details Profiler Build Dependencies Event Log
Gradle sync finished in 26 s 171 ms (4 minutes ago)
5:41 LF UTF-8 4 spaces

```java
public class ClassName {
    public static void main(String[] args)
    {
        // Code Statements go here
        System.out.println("Hello World!");
    }
}
```

Introduction

In this course we don't just learn Java but also learn how to solve problems.

It is an important skill for computer scientist to be able to:

1. Formulate problems.
2. Think creatively about solutions.
3. Express a solution clearly and accurately.

```
Hello World!
```

Figure 6: "Hello World!" program output screen

Task 4 - Complete part of the course

◁ Task

Section 1. Introduction to Programming covers a bit about Java and coding in general. Section 2. First

[1] https://en.wikipedia.org/wiki/%22Hello,_World!%22_program

steps in Java includes exercises on class syntax and printing statements. You don't need to do the exercises starting from variables as we will cover them in course lectures (unless you want to look ahead).

☞ Do make full use of the Piazza forum...
This is a valuable resource – if you get stuck, course staff and other students are often able to help. But make sure that you describe your problem clearly and include all of the necessary information. It is hard to answer a question which just asks *"why does my program not work?"*

You won't need to finish this course, but it might be a helpful additional practice tool alongside the lab exercises if you need it. You can always pick up from where you left off by opening the course from your recent projects list in the welcome menu.

3 Creating a Project From Scratch

Now that you have the tools set up and have feel for basic usage of IntelliJ, you should be ready to start writing your own code. For some exercises from labs, tutorials and assignments you might be provided with a full project template that you can just open.

For other cases you might only get a few Java source files and unit tests (which are also only Java source files) or nothing at all. In that situation, you can create your own blank project and add new files and provided files as required. The following section will show you how you can do that.

Task 5 - Create a new Java Project

◁ **Task**

1. Open IntelliJ and create a new project from the startup screen, or from the File 〉 New Project menu.
2. Select Java. Make sure the selected Project SDK is the JDK version you have installed. Leave the additional libraries at their default settings (nothing ticked) and press Next.
3. Do not create a project from template and select Next.
4. Give the project a name and choose a directory to store it on your hard disk. Leave the rest as it is and click Finish.

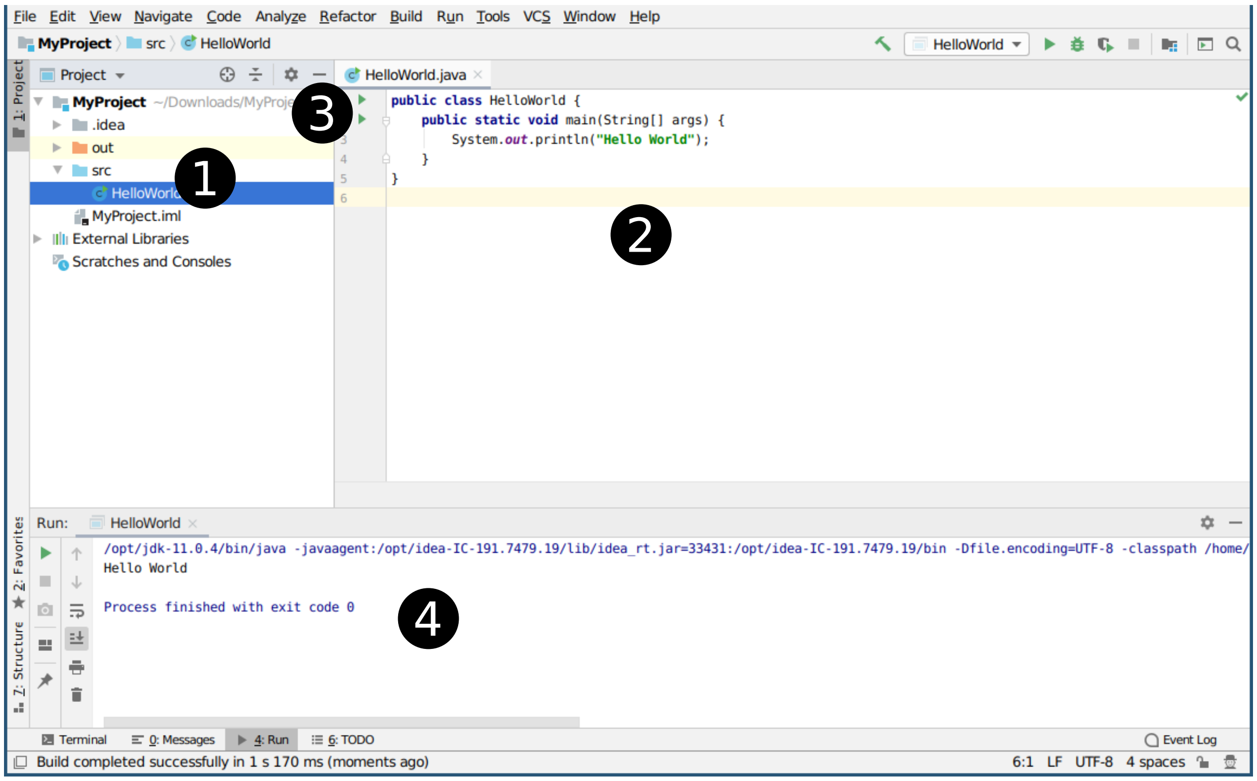
This will create an empty project with a `src` folder, IntelliJ configuration files (`.idea` and `.iml`) and nothing else. You can now add provided source files or create your own as described below.

Task 6 - Write a simple test program

◁ **Task**

1. Right-click on the source folder in the Project panel. For a Java project it is called src ❶.
2. Select New 〉 Java Class from the popup menu.
3. You can give the new class any name that you like. But if we call it `HelloWorld`, it will work for the rest of this tutorial. Make sure that whatever name you use starts with a capital letter.

This will have created a skeleton source file for the main class. This is where you should put your own code.



```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

```
/opt/jdk-11.0.4/bin/java -javaagent:/opt/idea-IC-191.7479.19/lib/idea_rt.jar=33431:/opt/idea-IC-191.7479.19/bin -Dfile.encoding=UTF-8 -classpath /home/
Hello World

Process finished with exit code 0
```

Build completed successfully in 1 s 170 ms (moments ago)

Figure 7: Creating a new project from scratch.

Edit the code ❷ so that the file contains the following . . .

```
public class HelloWorld {
  public static void main(String[] args) {
    System.out.println("Hello World");
  }
}
```

If you try to copy-paste this code, you will notice that the line numbers will get copied too. You could delete these, but then you would be missing out on a critical learning opportunity! Retyping code from memory is a really useful exercise for learning, so avoid copy pasting code whenever you can, especially when learning a new language or library.

While you are typing, notice the red marks which indicate syntax errors. These should disappear when you have entered the complete program. You may also see pop-up menus offering suggested completions. The first lab exercise will lead you through this code in more detail.

Finally . . .

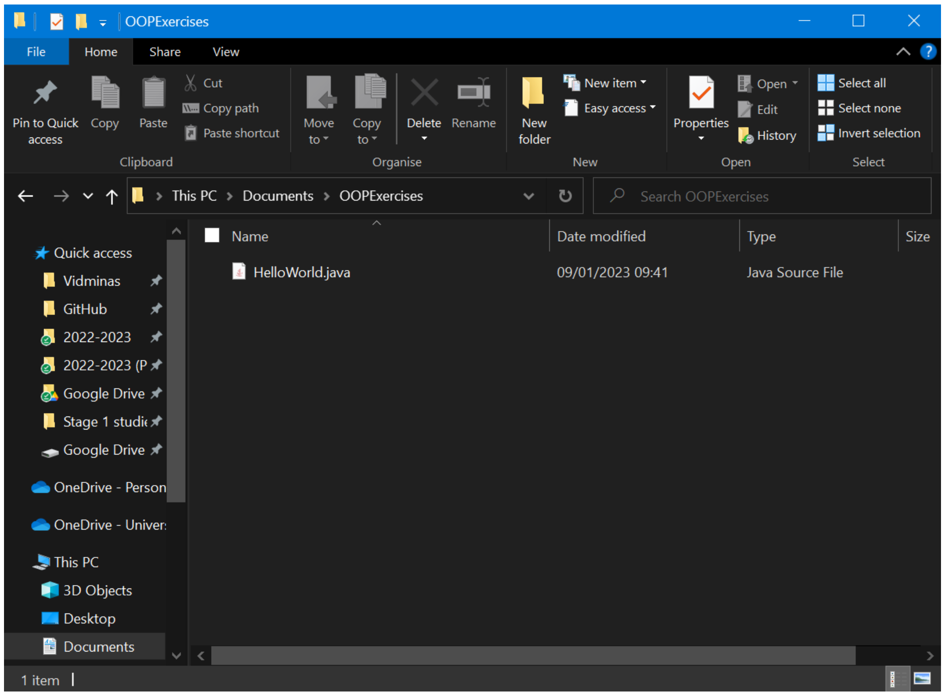
There should now be a green triangle ❸ to the left (in the gutter) of the Main method. Click this and select $\boxed{\text{run HelloWorld.main()}}$ to run the program.

The message should appear in the *Console* ❹ at the bottom of the window.

4 Running programs using the command line

The lab exercises include instructions for running single-class Java programs using the command line. This is equivalent to running code using IntelliJ's user interface (with the green triangle button), so you don't need to use it, but if you want to try, this is how:

1. Create a new text file in a directory of your choice (e.g., 'Documents/OOPExercises') and call it 'HelloWorld.java'
2. Open the file with any text editor like Notepad or Gedit
3. Write the same Hello World program code as in the instructions above. Save and close the file
4. Open a terminal window and navigate to the same directory using a 'cd' command (more details about this in the first lab sheet)
5. Run 'ls' to double check that you are in the correct directory – your file 'HelloWorld.java' should show up
6. Run 'javac HelloWorld.java' to compile the code into a bytecode '.class' file
7. Run 'java HelloWorld' to run the program



Name	Date modified	Type	Size
HelloWorld.java	09/01/2023 09:41	Java Source File	

Figure 8: Step 1 is to create a 'HelloWorld.java' file

The benefit of running code from the command line is that you don't need to create a whole new IDE project to run your code. On the other hand, you won't get any of the benefits of using an IDE like automatic code completion and error highlighting, and running unit tests is more difficult (this is not covered in the instructions, please use IntelliJ to work with unit tests).

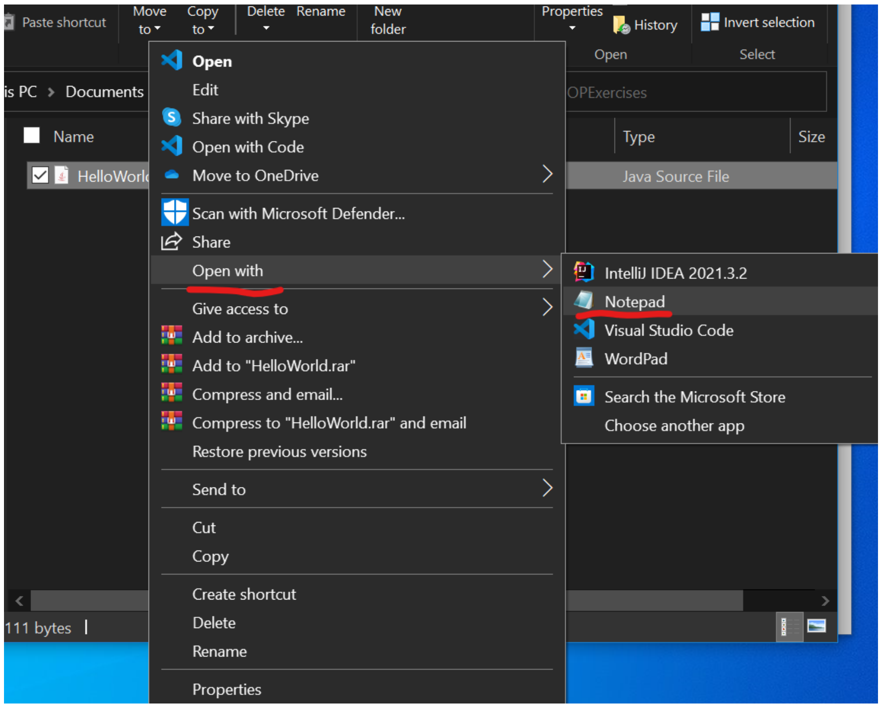
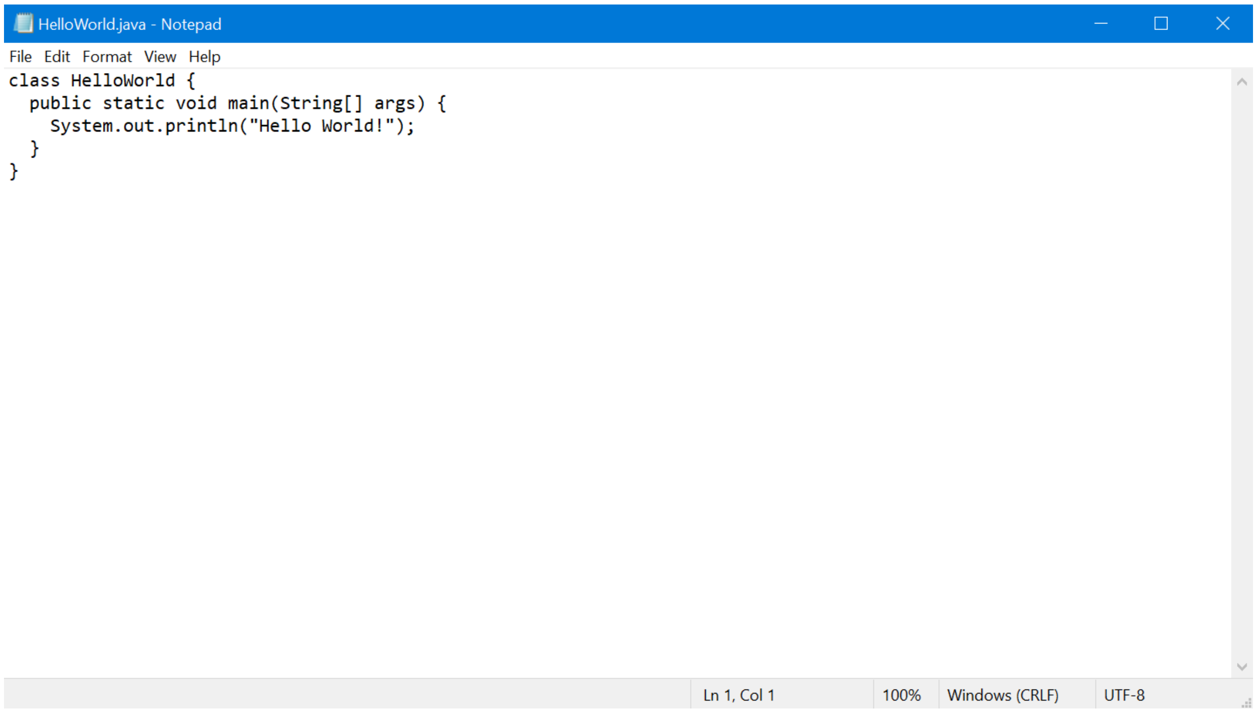
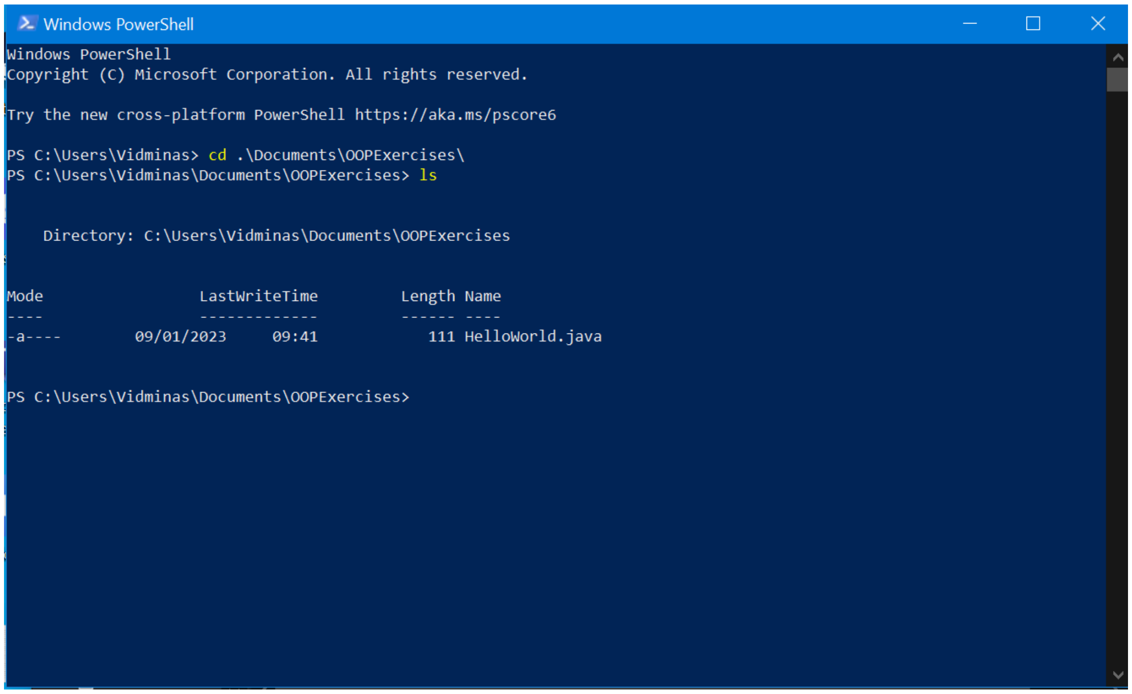


Figure 9: Step 2 is to open the file with a text editor



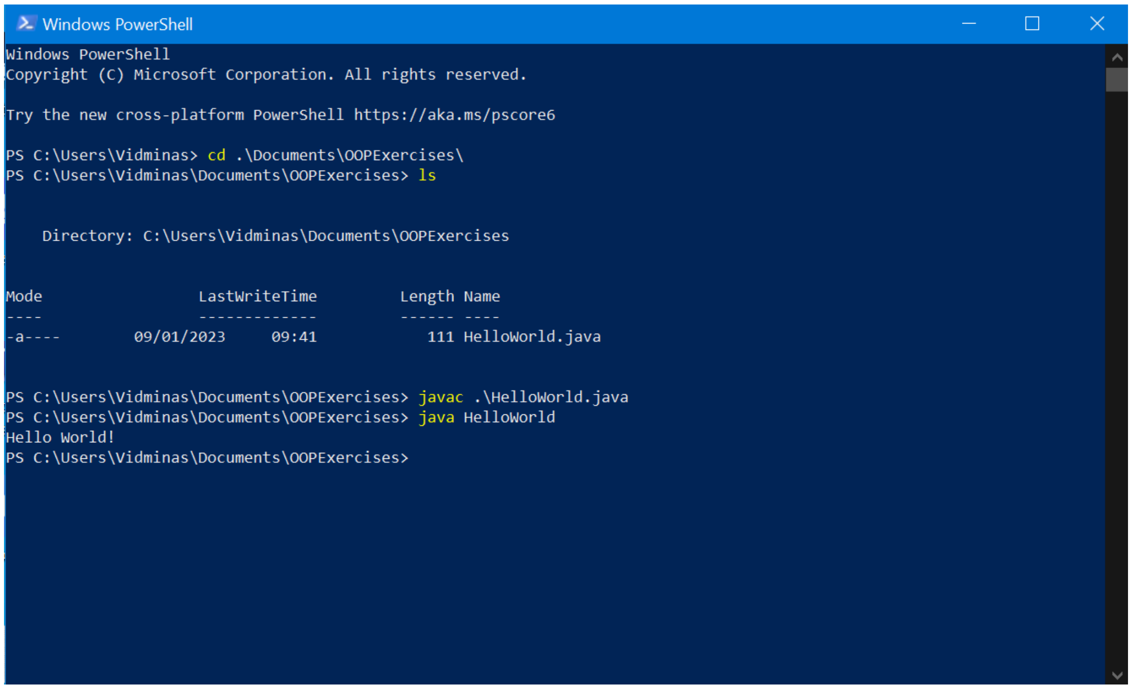
```
class HelloWorld {
  public static void main(String[] args) {
    System.out.println("Hello World!");
  }
}
```

Figure 10: Step 3 is to write the Hello World program code



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Vidminas> cd .\Documents\OOPExercises\
PS C:\Users\Vidminas\Documents\OOPExercises> ls


    Directory: C:\Users\Vidminas\Documents\OOPExercises


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        09/01/2023     09:41            111 HelloWorld.java


PS C:\Users\Vidminas\Documents\OOPExercises>
```

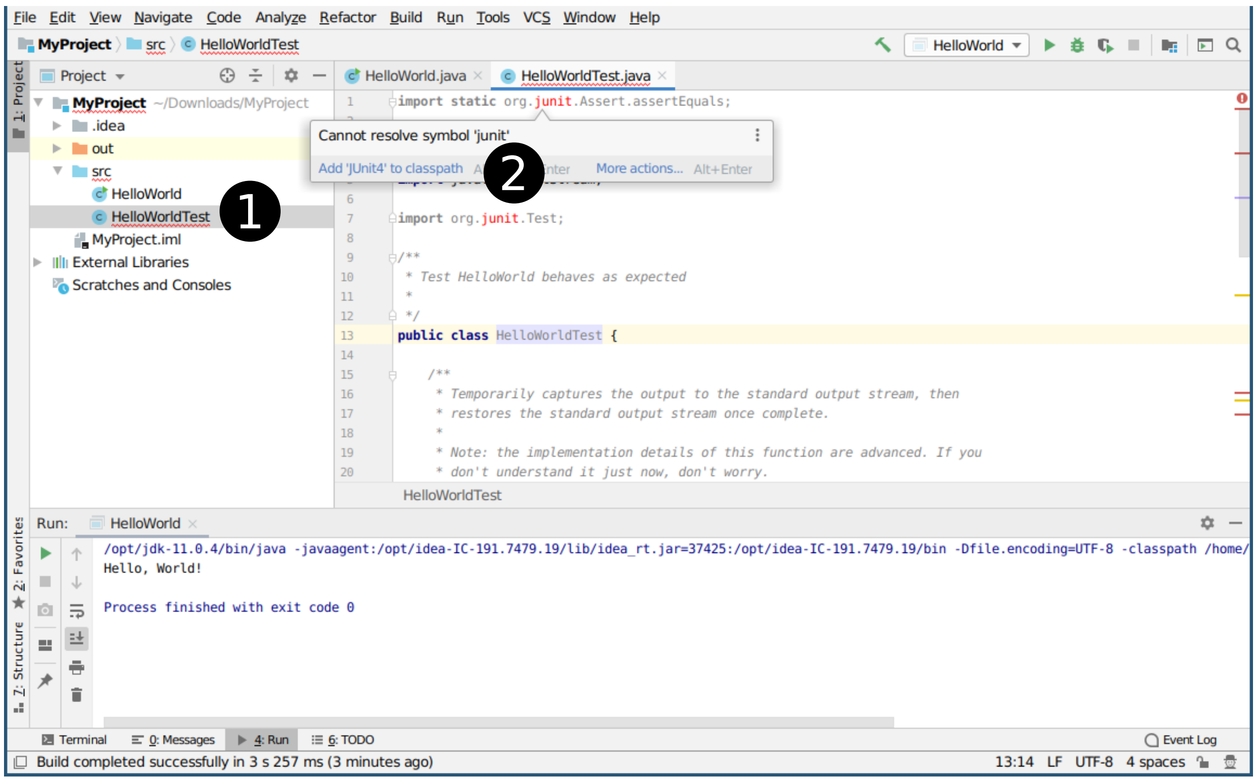
Figure 11: Step 4 is to run ‘cd’ to navigate to the directory with your code. Step 5 is to check whether the ‘HelloWorld.java’ file is there using ‘ls’



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Vidminas> cd .\Documents\OOPExercises\
PS C:\Users\Vidminas\Documents\OOPExercises> ls


    Directory: C:\Users\Vidminas\Documents\OOPExercises


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        09/01/2023     09:41            111 HelloWorld.java


PS C:\Users\Vidminas\Documents\OOPExercises> javac .\HelloWorld.java
PS C:\Users\Vidminas\Documents\OOPExercises> java HelloWorld
Hello World!
PS C:\Users\Vidminas\Documents\OOPExercises>
```

Figure 12: Step 6 is to compile the Java code using ‘javac’ and step 7 is to run the program using ‘java’

5 Automatic Testing with JUnit



```
import static org.junit.Assert.assertEquals;

import org.junit.Test;

/**
 * Test HelloWorld behaves as expected
 *
 */
public class HelloWorldTest {

    /**
     * Temporarily captures the output to the standard output stream, then
     * restores the standard output stream once complete.
     *
     * Note: the implementation details of this function are advanced. If you
     * don't understand it just now, don't worry.
```

Cannot resolve symbol 'junit'

Add 'JUnit4' to classpath   More actions... Alt+Enter

Run: HelloWorld

```
/opt/jdk-11.0.4/bin/java -javaagent:/opt/idea-IC-191.7479.19/lib/idea_rt.jar=37425:/opt/idea-IC-191.7479.19/bin -Dfile.encoding=UTF-8 -classpath /home/
Hello, World!

Process finished with exit code 0
```

Build completed successfully in 3 s 257 ms (3 minutes ago)   13:14 LF UTF-8 4 spaces

Figure 13: Configuring a Java Project for JUnit.

For nearly all exercises, you will be provided with JUnit tests so that you can quickly verify your own solutions. JUnit tests are the usual `.java` source files which have JUnit test implementations. IntelliJ integrates well with unit tests and has specific panels to show you a summary of test results, etc.

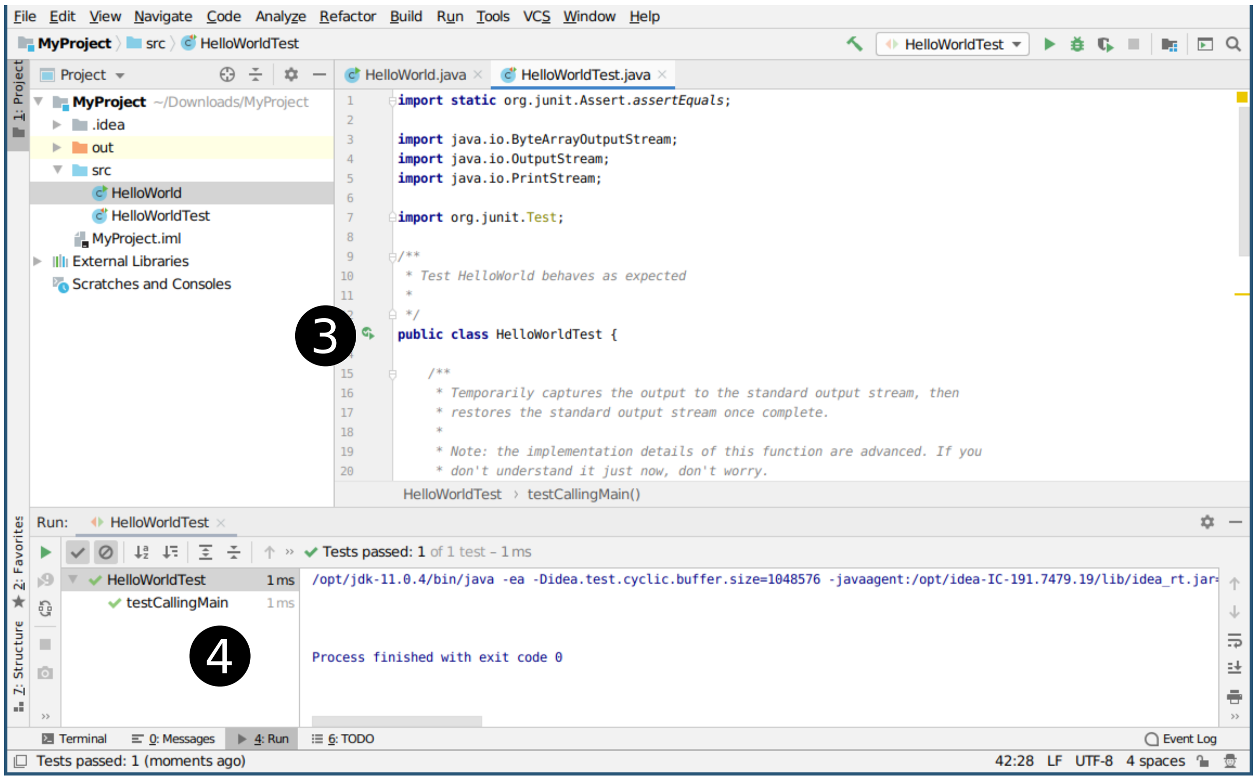
Task 7 - Add provided JUnit tests to your project

◁ **Task**

Let's run a test for the code you have just written:

- Download the test code from the initial lab exercise called `HelloWorldTest.java`. You can find it on the lab pages under the menu entry Java Files 〉 Automated Tests on the bottom left of the page.
- Copy this file into the src folder of your project using the file explorer of your operating system or the command line. IntelliJ will automatically update the Project Panel to display the added file. ❶
- For a Java project, the JUnit library needs to be included into the classpath. If not, you will see red error messages in the code Panel. When you hover your mouse over the `junit` import, you can see error details and a suggestion to fix it, Add 'JUnit5' to classpath (You may have to click on More actions... to see it if "Add 'Junit4' to classpath" appears instead like in the image above). ❷ Click on the quick fix and confirm the following dialog.
- Now you can see a green arrow symbol next to the name of the test class `HelloWorldTest`. Once you click and confirm, the tests will be executed for your code. ❸

- On the bottom of the screen, a Test Panel will open showing you green ticks if the tests passed. ❹



```java
import static org.junit.Assert.assertEquals;

import java.io.ByteArrayOutputStream;
import java.io.OutputStream;
import java.io.PrintStream;

import org.junit.Test;

/**
 * Test HelloWorld behaves as expected
 *
 */
public class HelloWorldTest {

    /**
     * Temporarily captures the output to the standard output stream, then
     * restores the standard output stream once complete.
     *
     * Note: the implementation details of this function are advanced. If you
     * don't understand it just now, don't worry.
```

Tests passed: 1 of 1 test – 1 ms

✓ HelloWorldTest	1 ms
✓ testCallingMain	1 ms

```
/opt/jdk-11.0.4/bin/java -ea -Didea.test.cyclic.buffer.size=1048576 -javaagent:/opt/idea-IC-191.7479.19/lib/idea_rt.jar

Process finished with exit code 0
```

Tests passed: 1 (moments ago)

Figure 14: Executing JUnit tests in a Java Project.

You can find more information on how to include libraries into Java projects in the IntelliJ documentation[2].

6 Conclusion

Tasks 1–7 will confirm that you have Java correctly installed and configured. If you have any problems at this stage, it is likely due to the installation or configuration of the development tools, and you should ask a lab demonstrator or post on the forum for help.

Congratulations on completing getting started and we hope you enjoy the course!

[2] https://www.jetbrains.com/help/idea/configuring-testing-libraries.html