

Introduction to Java

The Java Language

- ❖ Java was published in 1995 by James Gosling of Sun Microsystems
- ❖ Java is strongly typed (data types required) like C and C++ but utilizes a virtual machine (JVM) to run on multiple devices (portability). Also has 'automatic garbage collection'
- ❖ C# was later announced by Microsoft in 2000, many say to compete with Java!



A History of Java



The Java programming language was designed and implemented by a small team of people headed by **James Gosling** at Sun Microsystems in Mountain View, California throughout the early 1990s.



Java/Sun was acquired by Oracle Corporation in 2010.

<https://www.oracle.com/java/moved-by-java/timeline/>

A History of Java

The original team worked on designing software for consumer electronics.

They quickly found that existing programming languages, e.g. C and C++ were not adequate.

James was deterred by the lack of memory deallocation (garbage collection), use of pointers (access to memory addresses) in C++, and lack of portability.

Programs written in C and C++ had to be compiled for a particular computer chip. When a new chip came out the software had to be re-compiled to make full use of new features in the chip.



<https://www.oracle.com/java/moved-by-java/timeline/>



A History of Java

In 1990 James Gosling started the design of a new programming language that was meant to be more appropriate for consumer electronics, without the problems of traditional languages such as C and C++.

This project, called the Green Project, resulted in the development of a computer language which Gosling called Oak after an oak tree outside his office window at Sun.

But it was later discovered that there was already a computer language called Oak (Oak Technologies). When a group of Sun people visited a local coffee shop, the name Java was suggested - it stuck and the rest is history.

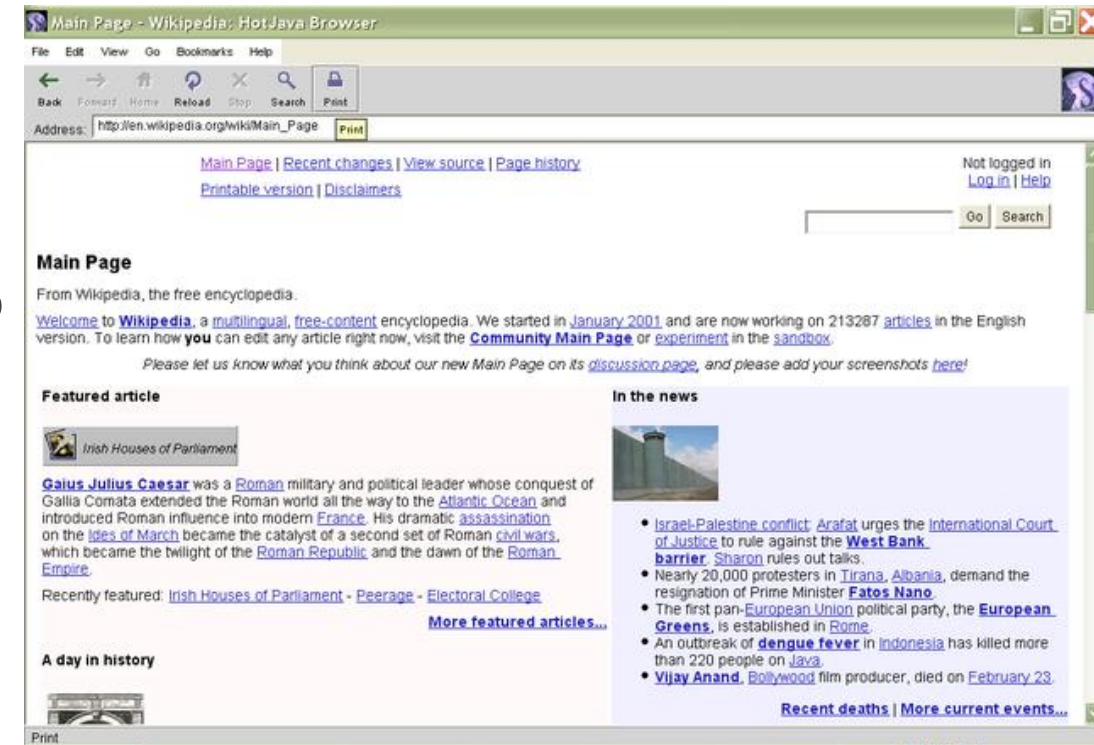
<https://www.oracle.com/java/moved-by-java/timeline/>

Java takes off

By sheer good fortune, the World Wide Web www exploded in popularity in 1993 and Sun saw the immediate potential of using Java to create Web pages (later Java Applets).

The team created 'WebRunner', named as homage to the movie Blade Runner. The browser was created using the Oak programming language and ran Oak applications. It later became 'HotJava'.

Sun formally announced Java at a conference in May 1993, and Java (JDK) 1.0 was released in 1996.



<https://www.oracle.com/java/moved-by-java/timeline/>

Why Java?

- Java is one of the most popular programming languages, alongside C++, C#, and Python, and web programming languages (HTML, CSS and JavaScript).
- According to Oracle, **Three billion** devices run Java.
- Furthermore, Android apps are also developed using Java.

Download Java from Oracle:

Java 21 and Java 17 available now

JDK 21 is the latest long-term support release of Java SE Platform.

[Learn about Java SE Subscription](#)

JDK 21 JDK 17 GraalVM for JDK 21 GraalVM for JDK 17

JDK Development Kit 21 downloads

JDK 21 binaries are free to use in production and free to redistribute, at no cost, under the [Oracle No-Fee Terms and Conditions](#) (NFTC).

JDK 21 will receive updates under the NFTC, until September 2026, a year after the release of the next LTS. Subsequent JDK 21 updates will be licensed under the [Java SE OTN License](#) (OTN) and production use beyond the [limited free grants](#) of the OTN license will [require a fee](#).

Linux macOS Windows

Product/file description	File size	Download
ARM64 Compressed Archive	186.35 MB	https://download.oracle.com/java/21/latest/jdk-21_linux-aarch64_bin.tar.gz (sha256)
ARM64 RPM Package	186.05 MB	https://download.oracle.com/java/21/latest/jdk-21_linux-aarch64_bin.rpm (sha256) (OL 8 GPG Key)
x64 Compressed Archive	188.04 MB	https://download.oracle.com/java/21/latest/jdk-21_linux-x64_bin.tar.gz (sha256)

<https://www.oracle.com/java/technologies/downloads/>

Why Java?

There are five main design goals that informed the creation of Java (Oracle 1999):

1. It must be simple, object-oriented, and familiar
2. It must be robust and secure (no pointers)
3. It must be architecture-neutral and portable (JVM)
4. It must execute with high performance (automatic memory management)
5. It must be interpreted, threaded and dynamic

Java Virtual Machine (JVM)

The Java Virtual machine

A **Java virtual machine (JVM)** is an abstract computing machine.

There are three notions of the JVM:

- specification,
- implementation
- instance

An instance of the JVM can execute any executable computer program compiled into Java bytecode. It is the code execution component of the Java platform.

Java Bytecode

Java bytecode is the instruction set of the Java virtual machine.

Each bytecode is composed by one, or in some cases two, bytes that represent the instruction (opcode), along with zero or more bytes for passing parameters.

- Essentially it is created when the high level language is converted into binary.

The Java Virtual Machine

The process of converting your source code into machine code is a **two stage process** in Java.

First the program is compiled into what is called Java Byte Code (essentially binary)

Providing the machine that you are working on has a Java Virtual Machine (JVM) the program can then be interpreted/linked and run.

This is what makes the Java platform independent. A Java program can be run on any platform, Unix, Windows, Mac etc

Compilers and Interpreters

Compilers

- operate on the entire program;
- provide a permanent binary file which may be executed (or run).

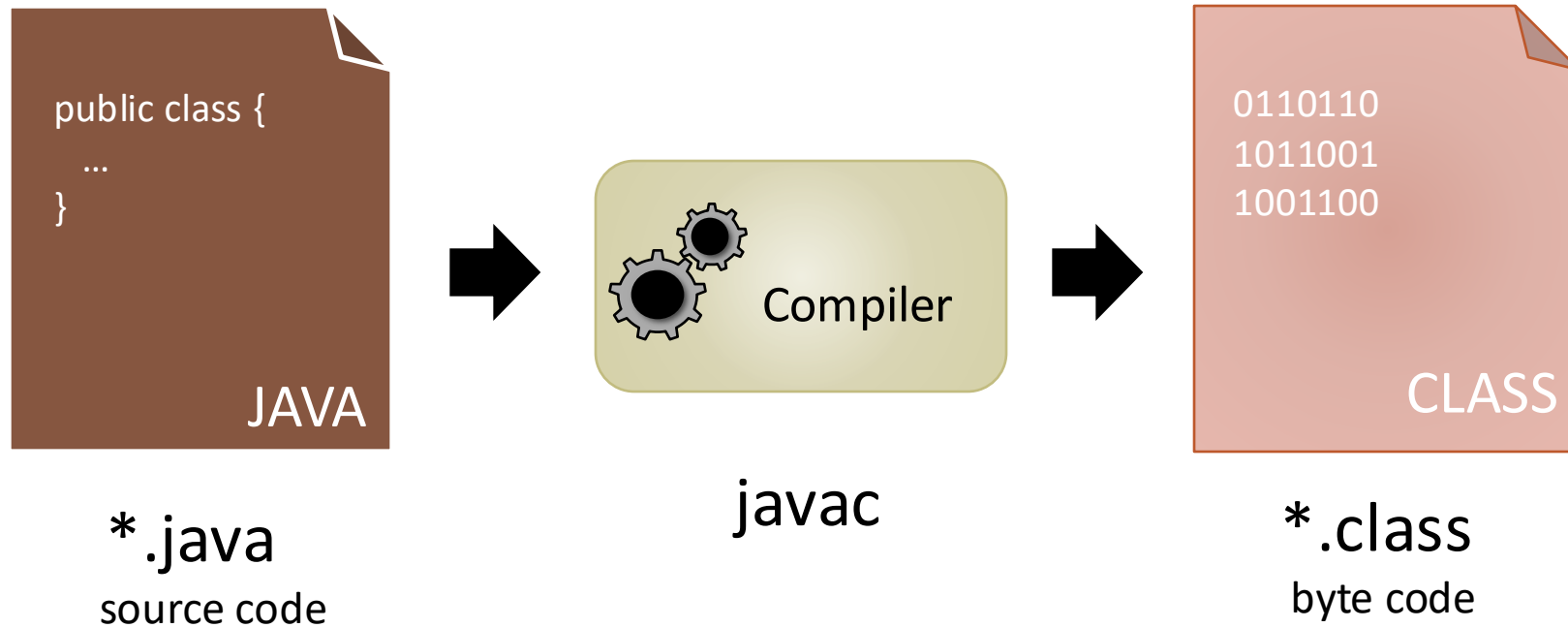
Interpreters:

- translate and execute the program one line at a time.

With Java, the processes of compilation and interpretation are combined.

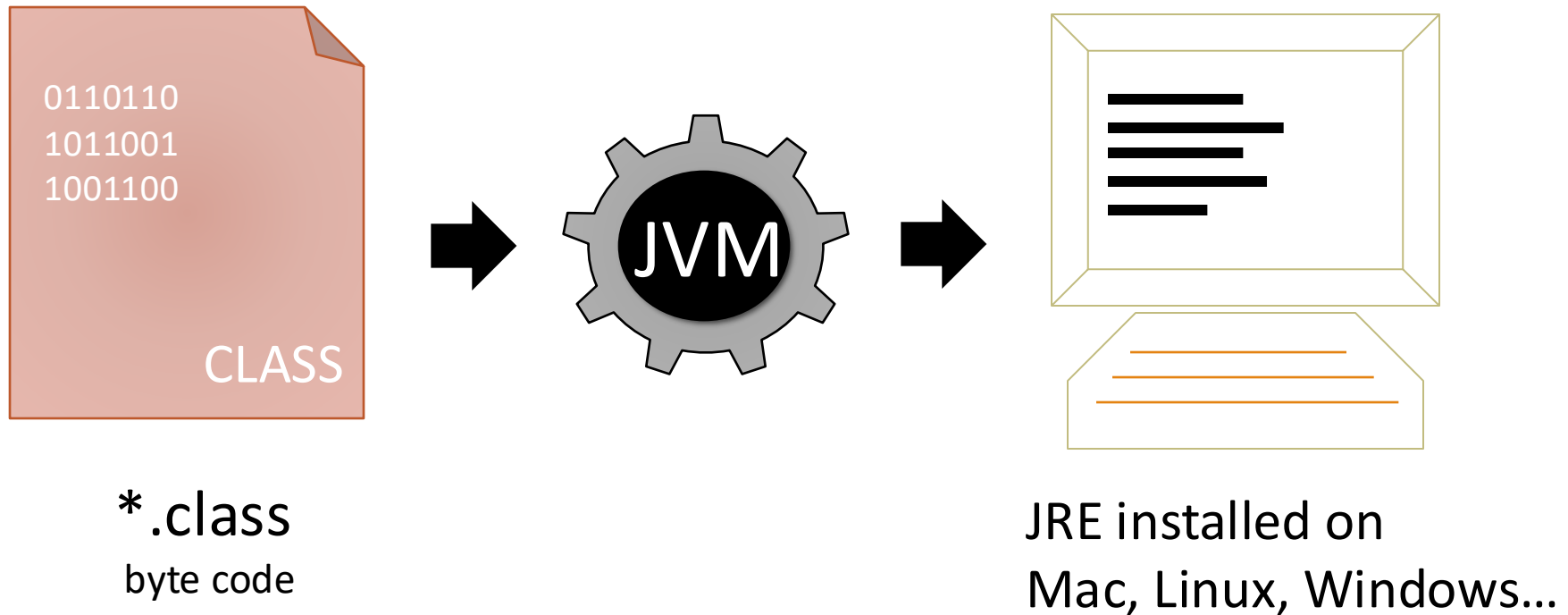
Compilation process (javac)

Compilation

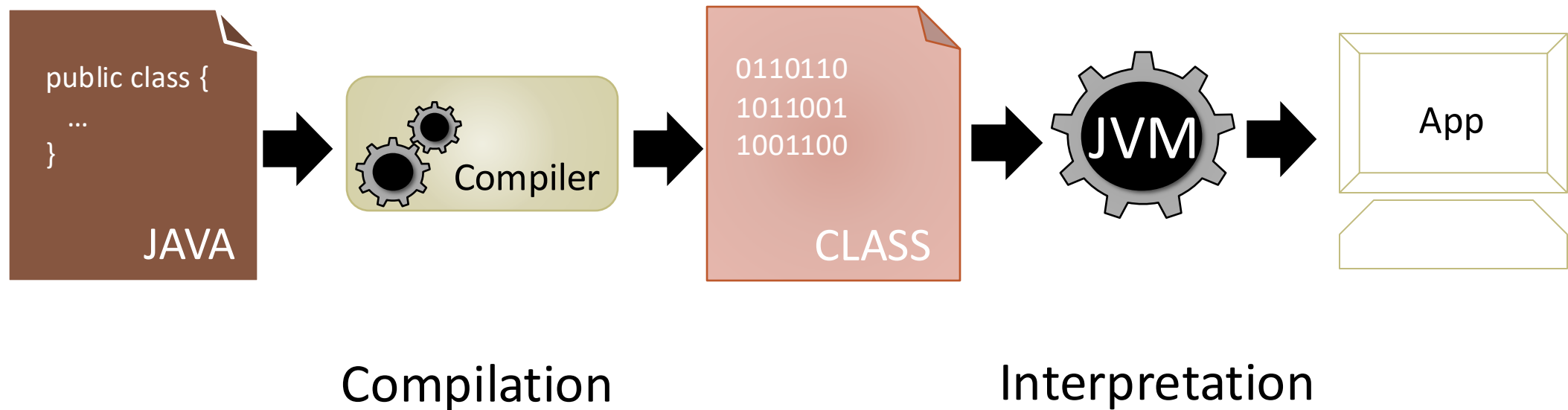


Interpretation process (JVM & JRE)

Interpretation



Compilation and interpretation



Java Acronyms

- JDK: Java Development Toolkit (for compiling applications)
- JRE: Java Runtime Environment (for running applications)
- JVM: Java Virtual Machine (for interpreting java code)
- Java API: Application Programming Interface
- JIT: Just In Time compilation
- IDE: Integrated Development Environment

The Java Language

Exploration of the differences

In the following section we shall explore some of the differences between the Java and Python language.

It is not intended to be an exhaustive coverage of all Java language features, but highlight some of the key differences in approach.

The intention is to also reinforce the core programming concepts. Same concepts, different syntax/expression across language.

Output

Output (Hello World) in Java vs Python



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

```
def main():  
    print("Hello, World!")  
  
if __name__ == "__main__":  
    main()
```

Input

String Input in Java vs Python



```
import java.util.*; //to get access to Scanner
public class Input{
    public static void main(String[] args){
        System.out.println("Enter a string: ");
        Scanner reader = new Scanner(System.in);
        String input = reader.nextLine();
        reader.close(); //release resources
        System.out.println("You entered "+ input);
    }
}
```



```
input = input("Enter a string: ")
print("You entered:", input)
```

Numerical Input in Java vs Python



```
import java.util.*; //to get access to Scanner
public class Input{
    public static void main(String[] args){
        System.out.println("Enter an integer: ");
        Scanner reader = new Scanner(System.in);
        int number = reader.nextInt();
        reader.close(); //release resources
        System.out.println("You entered "+ number);
    }
}
```



```
number = int(input("Enter an integer: "))
print("You entered:", number)
```

Strings

A String object is an array

A String object is an **immutable array of characters**.

Each character has a numbered position in the array (index):

String name = "Nick";

[0]	[1]	[2]	[3]
'N'	'i'	'c'	'k'

String code = "CO452";

[0]	[1]	[2]	[3]	[4]
'C'	'O'	'4'	'5'	'2'

Referring to characters in a String

You can refer to letters of a String through the index value.
In Java you can pass the index value as a parameter to the method **charAt()**

```
String name = "Nick";
```

[0]	[1]	[2]	[3]
'N'	'i'	'c'	'k'

```
System.out.println(name.charAt(0)); // displays 'N'
```

Java's equals() method

Whilst the equality operator (==) can be applied to primitive data (`int`, `char`, `boolean`), Strings are classes, so **the equality operator would compare memory addresses of String objects** rather than the values stored in each object

Use the method **equals** to compare the values stored at String variables rather than comparing memory addresses

```
if(name.equals("Nick"))
```

Comments

JavaDoc comments

```
/**
 * This is a Javadoc comment for class Hello
 */
public class Hello
{
    public static void main(String[] args)
    {
        System.out.println("Hello World");
    }
}
```

Constructor Summary

Constructors	
Constructor and Description	
<code>ArrayList()</code>	Constructs an empty list with an initial capacity of ten.
<code>ArrayList(Collection<? extends E> c)</code>	Constructs a list containing the elements of the specified collection, in the order they are returned by the collection's iterator.
<code>ArrayList(int initialCapacity)</code>	Constructs an empty list with the specified initial capacity.

Method Summary

Methods	
Modifier and Type	Method and Description
boolean	<code>add(E e)</code> Appends the specified element to the end of this list.
void	<code>add(int index, E element)</code> Inserts the specified element at the specified position in this list.
boolean	<code>addAll(Collection<? extends E> c)</code> Appends all of the elements in the specified collection to the end of this list, in the order that they are returned by the specified collection's Iterator.
boolean	<code>addAll(int index, Collection<? extends E> c)</code> Inserts all of the elements in the specified collection into this list, starting at the specified position.
void	<code>clear()</code> Removes all of the elements from this list.
Object	<code>clone()</code> Returns a shallow copy of this ArrayList instance.
boolean	<code>contains(Object o)</code> Returns true if this list contains the specified element.

Variables and Types

Data types

Java type	Description	Range of values
byte	Very small integers	-128 to 127
short	Small integers	-32,768 to 32,767
int	Big integers	-2,147,483,648 to 2,147,483,647
long	Very big integers	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	Real numbers	+/- $1.4 * 10^{-45}$ to $3.4 * 10^{38}$
double	Very big real numbers	+/- $4.9 * 10^{-324}$ to $1.8 * 10^{308}$
char	Characters	Unicode character set
boolean	Either true or false	true, false

Naming Variables in Java

You can choose any name for variables as long as

- the name is not already a word in the Java language (such as **class**, **void**);
- the name has no spaces in it;
- the name does not include operators such as + and -;
- the name starts either with a letter, an underscore (_), or a dollar sign (\$).

The convention in Java programs is to begin the name of a variable with a *lowercase* letter.

Variables in Java vs Python



```
public class Variables{  
    public static void main(String[] args){  
        char letter = 'A';  
        String name = "Nick";  
        int id = 9876;  
        double pi = 3.14;  
    }  
}
```



```
letter = 'A'  
name = "Nick"  
id = 9876  
pi = 3.14
```

Constants

Constants in Java vs Python



```
public class Constants
{
    public static void main(String[] args)
    {
        final int MAX_MARK = 100;
        final double PI = 3.14;
    }
}
```



```
MAX_MARK = 100
PI = 3.14
```

Selection

If statements in Java vs Python



```
...
int number = 5;
if (number > 0){
    System.out.println("The number is positive.");
}
else if(number < 0){
    System.out.println("The number is negative.");
}
else{
    System.out.println("The number is 0.");
}
...
```



```
number = 5
if number > 0:
    print("The number is positive.")
elif (number < 0):
    print("The number is negative.")
else:
    print("The number is 0.")
```

switch statement

The switch statement selects between cases

```
char grade = 'C';  
  
switch(grade) {  
    case 'A' : System.out.println("You achieved an A grade"); break;  
    case 'B' : System.out.println("You achieved a B grade"); break;  
    case 'C' : System.out.println("You achieved a C grade"); break;  
    case 'D' : System.out.println("You achieved a D grade"); break;  
    case 'F' : System.out.println("You achieved a F grade"); break;  
    default : System.out.println("Invalid grade");  
}
```

The '?' Operator: an example

```
int a = 18;  
int b = 7;  
int highest;  
highest = (a > b) ? a : b;    //assign highest  
System.out.println("The highest number is " + highest);
```

If the comparison evaluates to **true**, the **?** operator returns the value (**a**) to the **left of the : (colon)**

If the comparison evaluates to **false**, the **?** operator returns the value (**b**) to the **right of the : (colon)**

Iteration



...

```
public class Program{  
    public static void main(String[] args){  
        int[] numbers = {1,2,3,4,5};  
        for (Integer i : numbers){  
            System.out.println(i);  
        }  
    }  
}
```

```
for i in range(1,6):  
    print(i)
```

Java do while loop

The do while loop repeats whilst true

```
char response;  
do  
{  
    // program instructions go here  
    System.out.print("another go (y/n)?");  
    response = reader.nextChar(); // Java input  
}  
while (response == 'y');
```

Functions



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



```
def say_hello():  
    print("Hello, World!")  
  
say_hello()
```

Arrays

For loop in Java vs Python



...

```
public class Program{  
    public static void main(String[] args){  
        int[] numbers = {1,2,3,4,5};  
        for (int i = 0; i < 5; i++){  
            System.out.println(numbers[i]);  
        }  
    }  
}
```



```
numbers = [1,2,3,4,5]  
  
print(numbers)
```

Java's Collections Framework

Java Collections

Java created classes around key data structures:

- ❖ List
- ❖ LinkedList
- ❖ ArrayList
- ❖ Set
- ❖ HashSet
- ❖ Map

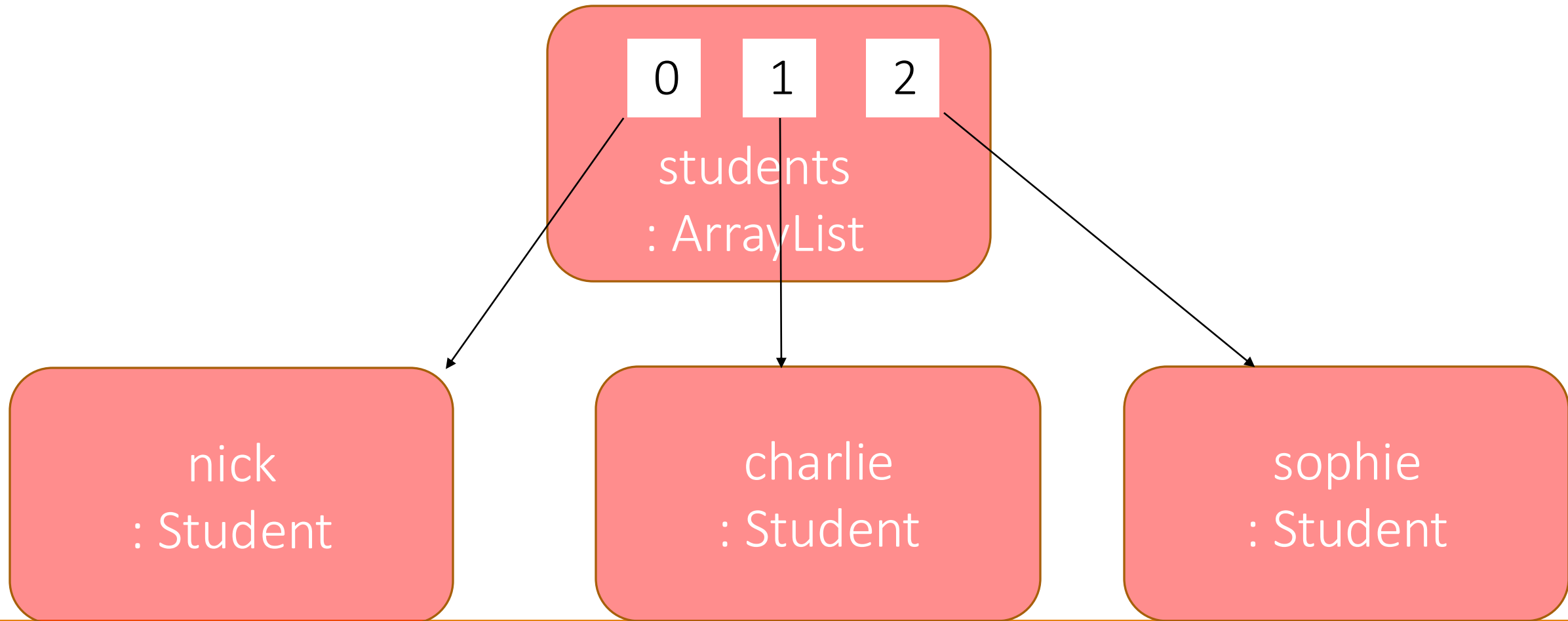
ArrayList

An instantiation of an ArrayList allows objects to be added (appended) to the end of the list. An ArrayList has no fixed-size so more memory can be allocated if more objects are added to the list.

Objects added do have an index position.

Can find objects by iterating through the list, or if the index is known, can be accessed directly through the index position.

Visualisation of an ArrayList



Syntax for instantiating an ArrayList

The syntax for creating objects of Collection classes is similar to creating objects of classes. **However, the difference is that Collections are Generic and require the type of objects to be stated in the angle brackets <>**

```
private ArrayList<Student> students = new ArrayList<>();
```

Scope Collection < type of objects > identifier = new constructor<>();

Reminder of syntax for creating objects

Similar to creating arrays (last week), we have to declare objects (variables) of a class type (data type). Then the **new** keyword instructs the compiler to allocate the appropriate amount of memory for an object (variable) of this type. Then call the constructor (method with the same name as the class).

Student nick = new Student();

classname objectname = new classname();
(aka. 'constructor')

Syntax for instantiating an ArrayList

Remember to import the ArrayList Class from the java.util package above the class definition.

```
import java.util.ArrayList;  
/**  
 * Class comment...  
 */  
public class StudentTester  
{  
    private ArrayList<Student> students = new ArrayList<>();  
}
```

Some methods of the ArrayList

add()

remove()

clear()

get()

size()

Adding objects through the method

Conveniently, we can invoke the 'add' method through an ArrayList object to append the items to the list:

```
students.add(nick);
```

```
students.add(charlie);
```

```
students.add(sophie);
```


For each loop with collection

The **for each** loop can be used to iterate through collections of objects.

Requires an object to be declared of the type of item that is in the collection:

```
for(Student student : students)
{
    student.print();
}
```

Finding an item in an ArrayList

... and can check that sought after value matches an item of the ArrayList:

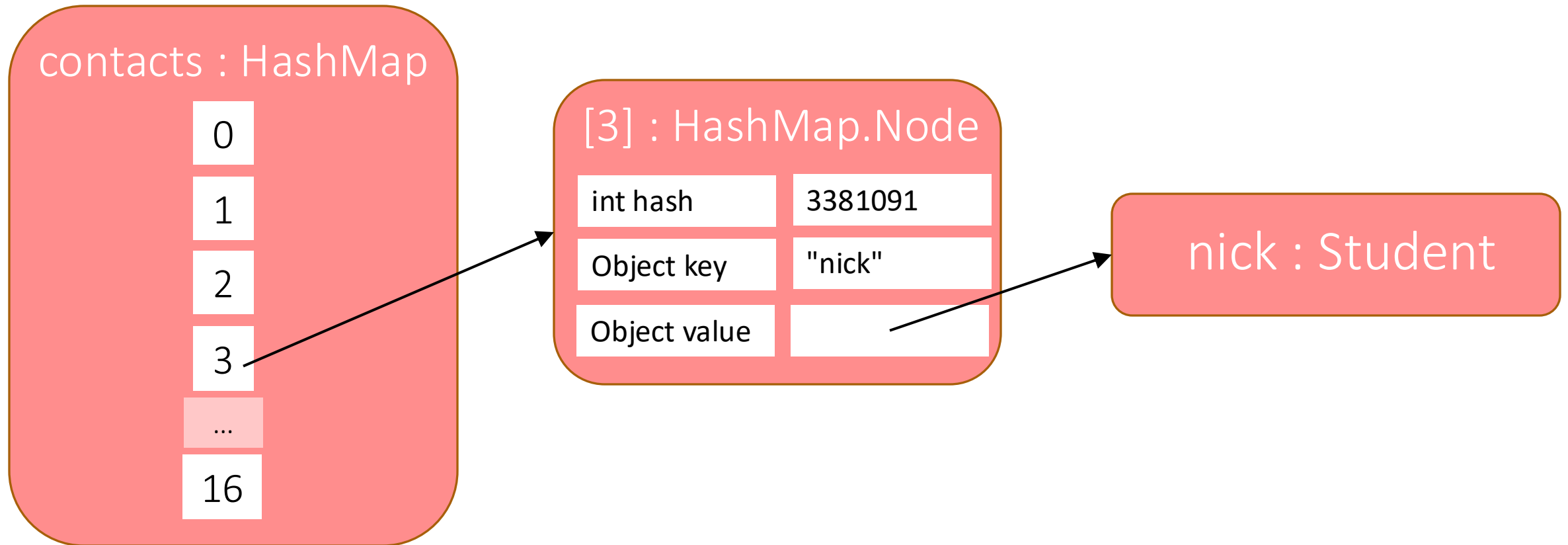
```
public Student findByID(int id)
{
    for(Student student : students)
    {
        if(student.getID() == id)
            return student;
    }
    return null;
}
```

Hash Map

Items are added to an ArrayList in the order that the 'add' method is called (added to the end of the list). LinkedLists also append or prepend items.

HashMaps, however, will 'put' items (values) in a position that corresponds with their 'key'. Items (values) can subsequently be retrieved by their key (e.g a String literal). A String can be converted to an integer position by the process of Hashing. Each character has an integer ASCII value.

Visualisation of a HashMap



Classes

Classes in Java vs Python



```
class Student{  
    private int id;  
    private String name;  
  
    public Student(int id, String name){  
        this.id = id;  
        this.name = name;  
    }  
}
```



```
class Student:  
    def __init__(self, id, name):  
        self.id = id  
        self.name = name
```

Objects



```
...  
public class Program{  
    public static void main(String[] args){  
        Student nick = new Student(1234, "Nick");  
        nick.print();  
    }  
}
```

```
from student import Student  
  
nick = Student(1234, "Nick")  
nick.print()
```


Inheritance



```
class Child extends Parent {  
    private string name;  
  
    public Child(int id, String name){  
        super(id); //call parent constructor  
        this.name = name;  
    }  
}
```



```
class Child(Parent):  
    def __init__(self, id, name):  
        super().__init__(id)  
        self.name = name
```

Summary

Summary

Java went public in 1995 (JDK 1.0 released January 1996)

Java Virtual Machine (JVM) was a transition from the 'compilation' to hardware specs (C/C++) and towards interpretation by VM (Python/Java/C#) now as there is more need for cross platform technology.

Java known for its portability (JVM), automatic 'garbage collection', Collections (generics), and for underpinning apps made in Android Studio.

Java Graphics: JavaFX

Java Graphics – AWT to Swing to JavaFX

In the earliest versions of Java, graphical programming could only be achieved through use of the Abstract Window Toolkit (AWT) package.

From 2000-2014, Swing was the main platform for producing Java Graphics. But has since become outdated (OS' have moved on and changed their styles).

With the release of Java 8 in 2014 came JavaFX.

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help demo [C:\Users\nickd\Desktop\demo] - index.html

demo

Project

- demo C:\Users\nickd\...
- .gradle
- .idea
- .mvn
- src
 - main
 - java
 - resources
 - static
 - index
 - template
 - applicati
 - test
 - target
 - .gitignore
 - HELP.md
 - mvnw
 - mvnw.cmd
 - pom.xml
- External Libraries
- Scratches and Console

Terminal: Local x +

Version Control TODO Problems Terminal Dependencies

New Project

Java

Maven

Gradle

Android

IntelliJ Platform Plugin

JavaFX

Groovy

Kotlin

Multi-module Project

Empty Project

Desktop application with **JavaFX** toolkit.

Name: JavaFXTest

Location: ~\Desktop\JavaFXTest

Language: Java Kotlin Groovy

Build system: Maven Gradle

Test framework: JUnit TestNG

Group: com.example

Artifact: JavaFXTest

Project SDK: 17 Amazon Corretto version 17.0.2

Previous Next Cancel Help

FileEditViewNavigateCodeRefactorBuildRunToolsVCSWindowHelpJavaFXTest - HelloApplication.java

JavaFXTest > src > main > java > com > example > javafxtest > HelloApplication > start

Project

JavaFXTest C:\Users\nickd\Desktop\JavaFXTest

src

main

java

com.example.javafxtest

module-info.java

resources

com.example.javafxtest

hello-view.fxml

target

JavaFXTest.iml

pom.xml

External Libraries

Scratches and Consoles

m pom.xml (JavaFXTest)

hello-view.fxml

HelloController.java

HelloApplication.java

12

13

14

15

16

17

18

19

20

21

22

23

@

public void start(Stage stage) throws IOException {

FXMLLoader fxmlLoader = new FXMLLoader(HelloApplication.class.getResource("hello-view.fxml"), v: 320, v1: 240);

Scene scene = new Scene(fxmlLoader.load(), 320, 240);

stage.setScene(scene);

stage.show();

}

public static void main(String[] args) {

launch(args);

}

Hello!

Welcome to JavaFX Application!

Hello!

Run: Unnamed

"C:\Program Files\Amazon Corretto\jdk17.0.2_8\bin\java.exe" ...

Version Control

Run

TODO

Problems

Terminal

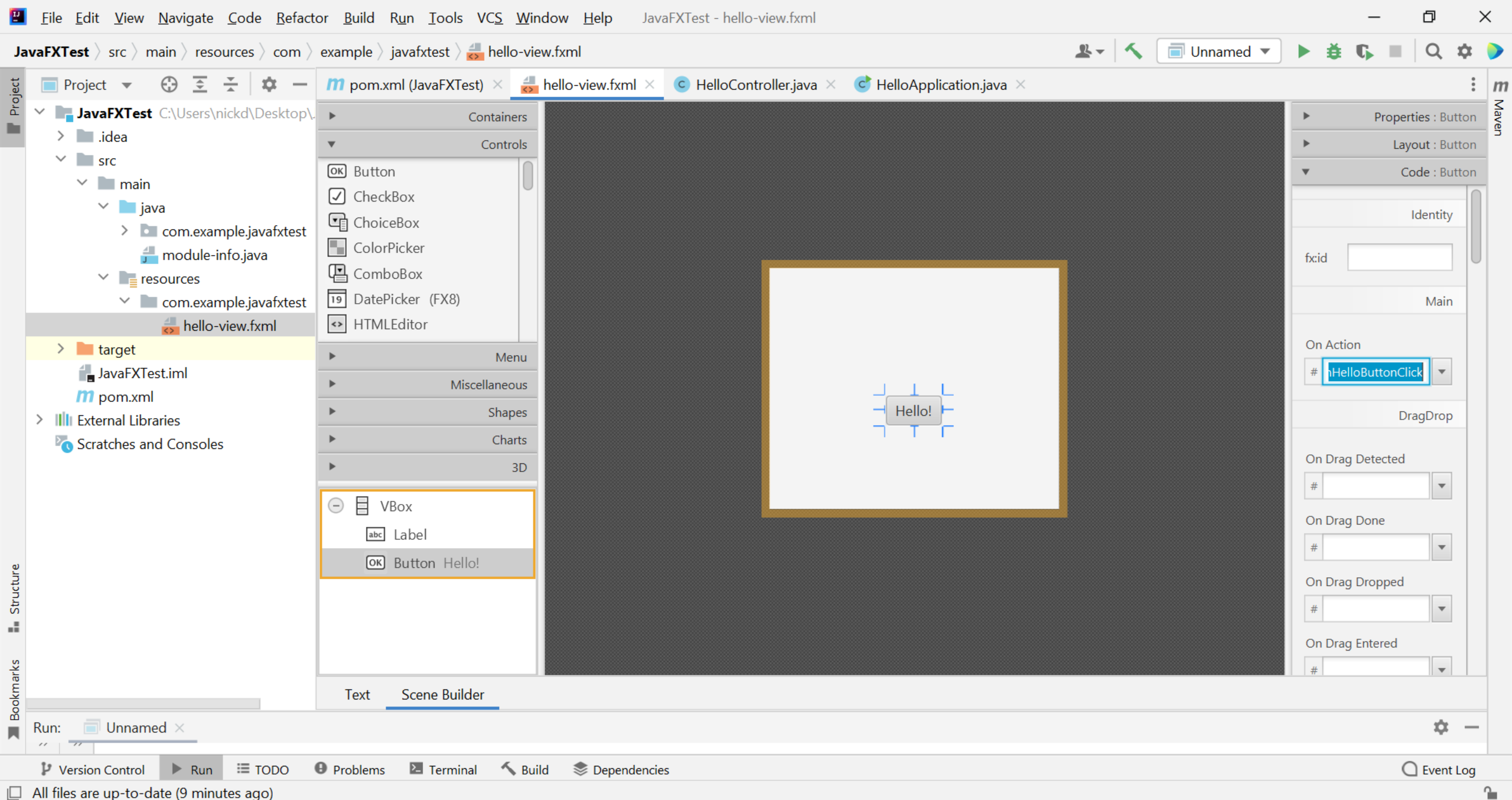
Build

Dependencies

Event Log

All files are up-to-date (moments ago)

16:31 LF UTF-8 4 spaces



FileEditViewNavigateCodeRefactorBuildRunToolsVCSWindowHelpJavaFXTest - HelloController.java

JavaFXTest > src > main > java > com > example > javafxtest > HelloController

Projectpom.xml (JavaFXTest)hello-view.fxmlHelloController.javaHelloApplication.java

JavaFXTest C:\Users\nickd\Desktop\> .idea> src> main> java> com.example.javafxtestHelloApplicationHelloControllermodule-info.javaresources> com.example.javafxtesthello-view.xmltarget> JavaFXTest.imlpom.xml> External LibrariesScratches and Consoles

1package com.example.javafxtest;
2
3import ...
4
5
6public class HelloController {
7 @FXML
8 private Label welcomeText;
9
10 @FXML
11 protected void onHelloButtonClick() {
12 welcomeText.setText("Welcome to JavaFX Application!");
13 }
14
15}

Version ControlTODOProblemsTerminalBuildDependenciesEvent Log

10:1 (120 chars, 3 line breaks) LF UTF-8 4 spaces

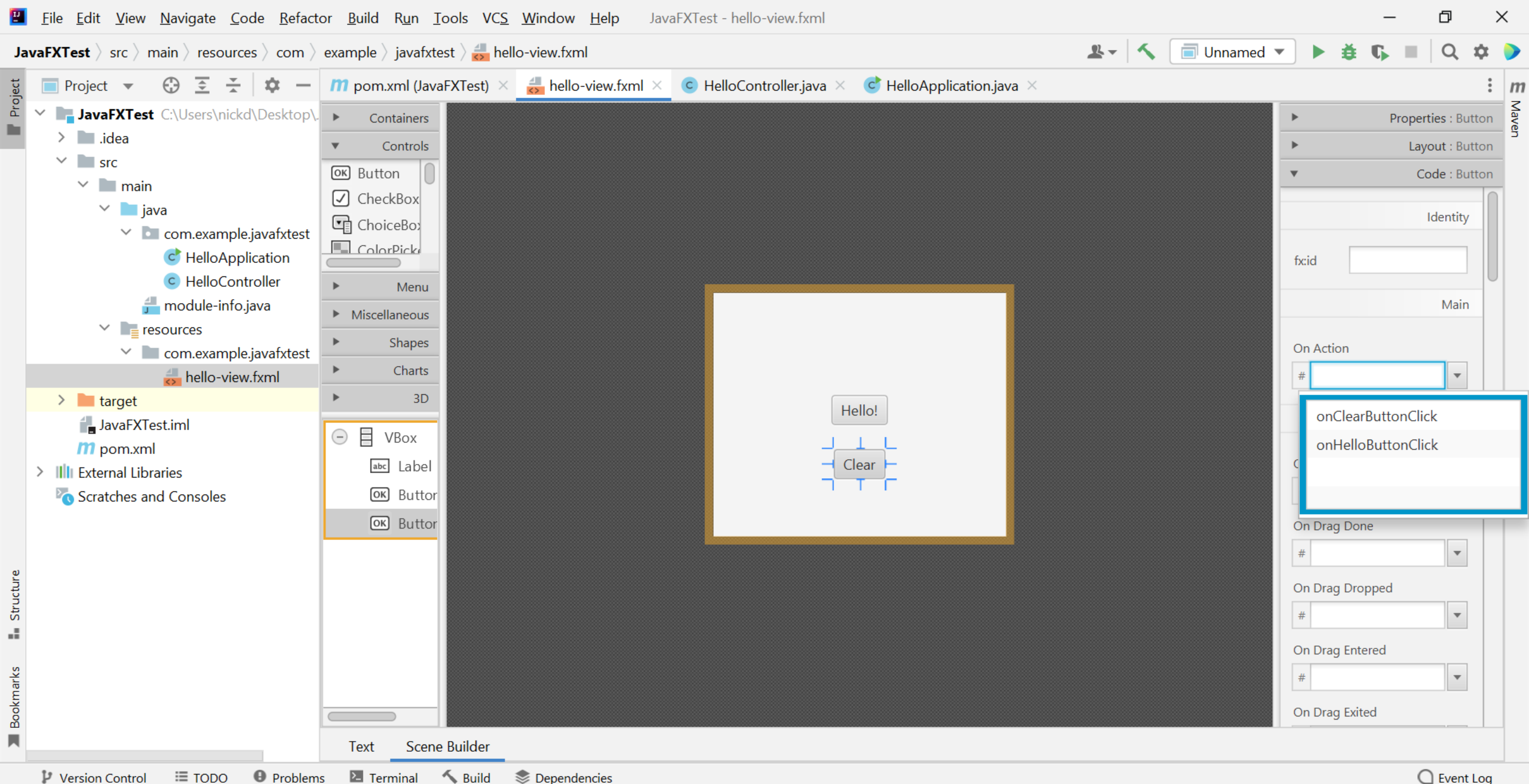
Project

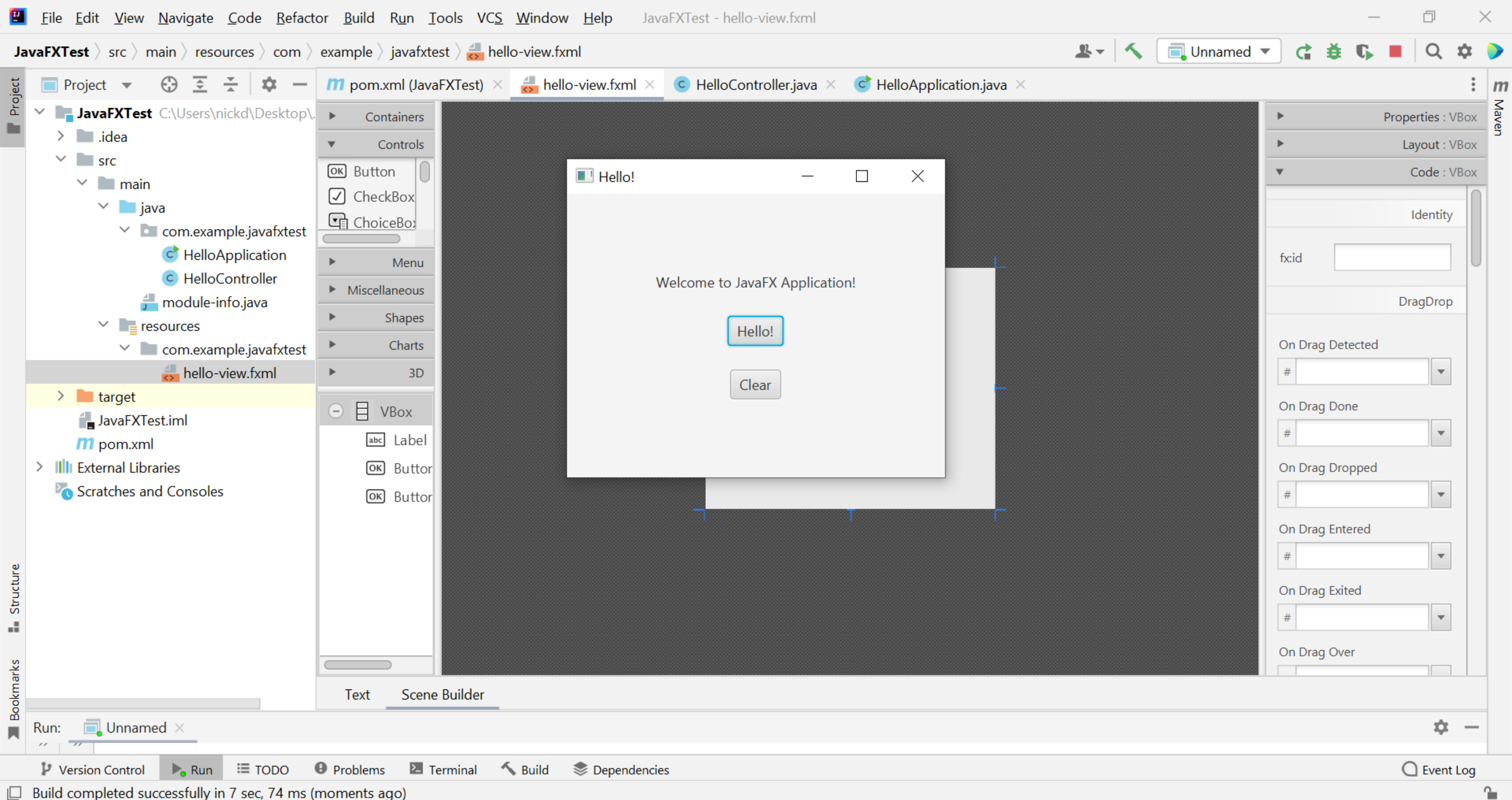
- JavaFXTest C:\Users\nickd\Desktop\
 - .idea
 - src
 - main
 - java
 - com.example.javafxtest
 - HelloApplication
 - HelloController**
 - module-info.java
 - resources
 - com.example.javafxtest
 - hello-view.fxml
 - target
 - JavaFXTest.iml
 - pom.xml
 - External Libraries
 - Scratches and Consoles

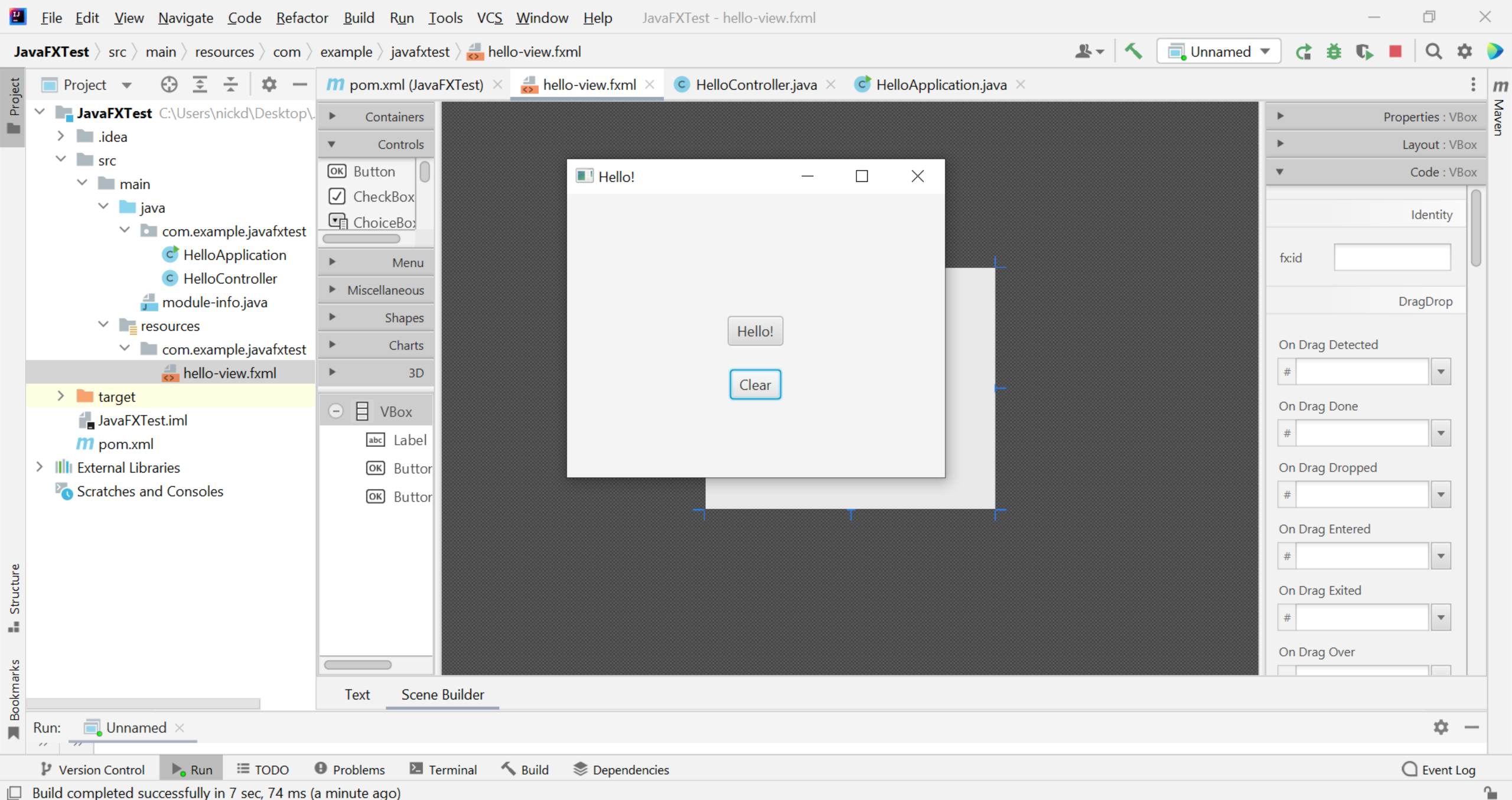
```

1 package com.example.javafxtest;
2
3 import ...
4
5
6 public class HelloController {
7     @FXML
8     private Label welcomeText;
9
10    @FXML
11    protected void onHelloButtonClick() {
12        welcomeText.setText("Welcome to JavaFX Application!");
13    }
14
15    @FXML
16    protected void onClearButtonClick() {
17        welcomeText.setText("");
18    }
19 }

```







Android Studio

Structure

1: Project

2: Structure

3: Layout Captures

4: Build Variants

5: Resource Manager

6: MainActivity

Create New Project

Select a Project Template

Phone and Tablet Wear OS TV Automotive Android Things

No Activity

Basic Activity

Empty Activity

Bottom Navigation Activity

Bottom Navigation Activity
Creates a new activity with bottom navigation

Previous Next Cancel Finish

What's New

4.2 Now ...

Android Studio 4.2 is now available in the stable channel. Update and restart the IDE to take advantage of the latest performance improvements and features described below.

[Read in a browser](#)

Update Now

[Update and Restart](#)

You can update later by selecting **Help > Check for updates** from the menu bar.

Upgrade Assistant...

New Upgrade Assistant for version for your project.

Built on top of the existing A rough project-wide update help prevent potential bre

FileEditViewNavigateCodeAnalyzeRefactorBuildRunToolsVCSWindowHelp

TestApp [C:\Users\nickd\AndroidStudioProjects\TestApp] - ...res\layout\fragment_home.xml [app]

TestApp > app > src > main > res > layout > fragment_home.xml

Android > mobile_navigation.xml > fragment_home.xml > activity_main.xml

Project
Resource Manager
Structure
Layout Captures
Variants
Build

app

manifests

java

java (generated)

res

drawable

layout

activity_main.xml

fragment_dashboard.xml

fragment_home.xml

fragment_notifications.xml

menu

bottom_nav_menu.xml

mipmap

navigation

values

Gradle Scripts

Palette

Common

Ab TextView

Button

ImageView

RecyclerView

<> <fragment>

ScrollView

Switch

Text

Buttons

Widgets

Layouts

Containers

Google

Legacy

Component Tree

ConstraintLayout

Ab text_home

helloButton "Hello"

Pixel 29 AppTheme Default (en-us)

0dp

HELLO

Attributes

helloButton Button

End → EndOf parent (0dp)

Top → BottomOf text_home (0dp)

Bottom → BottomOf parent (0dp)

layout_width 358dp

layout_height 107dp

visibility

visibility

Common Attributes

style

onClick

background

text

text

contentDescription

textAppearance

alpha

All Attributes

alpha

AppCompat

Body1

Body2

Display1

Display2

Display3

Display4

Large

Medium

Run

TODO

Profiler

Logcat

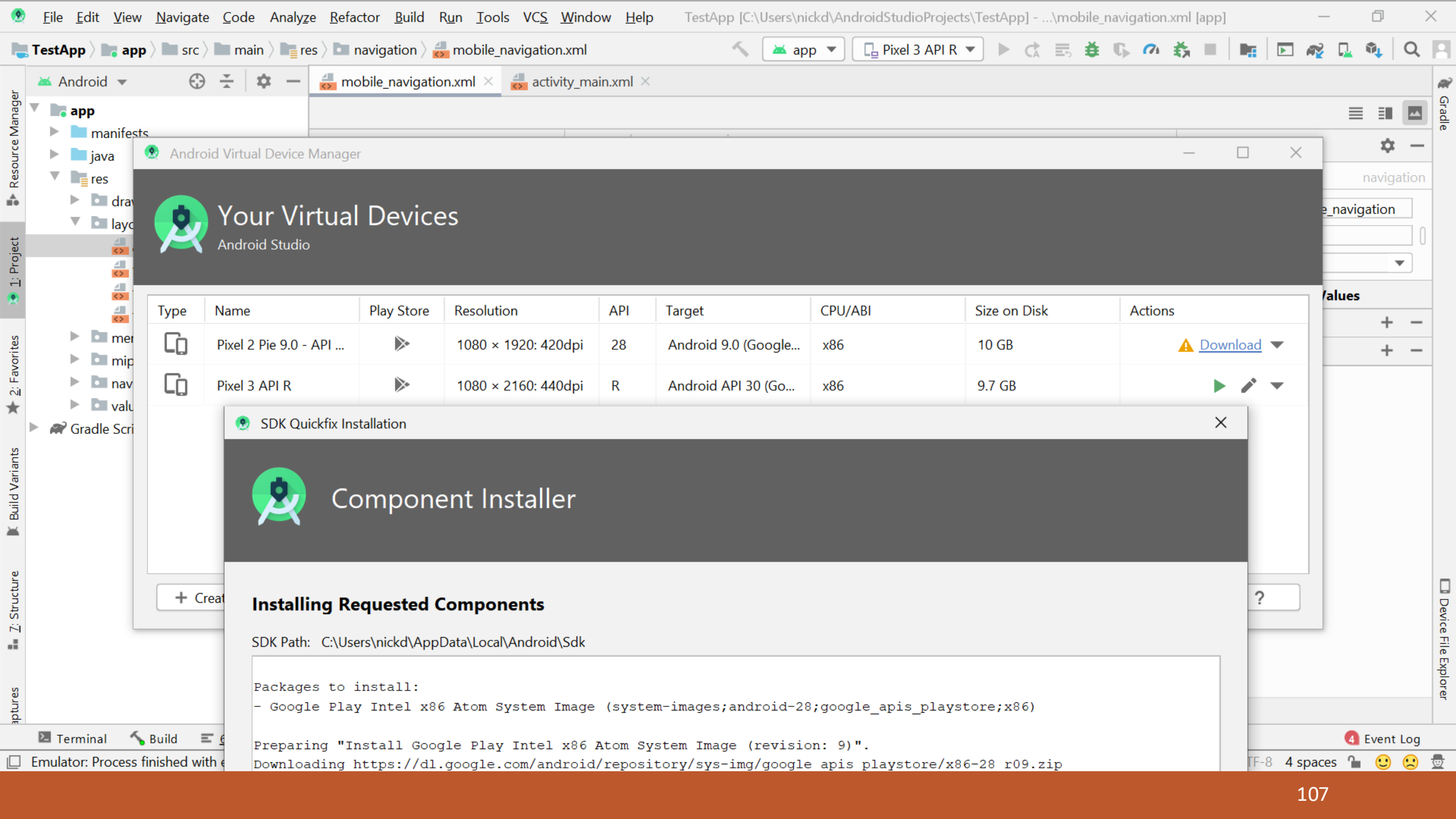
Build


Terminal

Event Log

Install successfully finished in 337 ms.: App restart successful without requiring a re-install. (11 minutes ago)








CRLF UTF-8 4 spaces






Your Virtual Devices

Android Studio

Type	Name	Play Store	Resolution	API	Target	CPU/ABI	Size on Disk	Actions
	Pixel 2 Pie 9.0 - API ...		1080 × 1920: 420dpi	28	Android 9.0 (Google...)	x86	10 GB	 Download ▼
	Pixel 3 API R		1080 × 2160: 440dpi	R	Android API 30 (Go...)	x86	9.7 GB	  ▼



Component Installer

Installing Requested Components

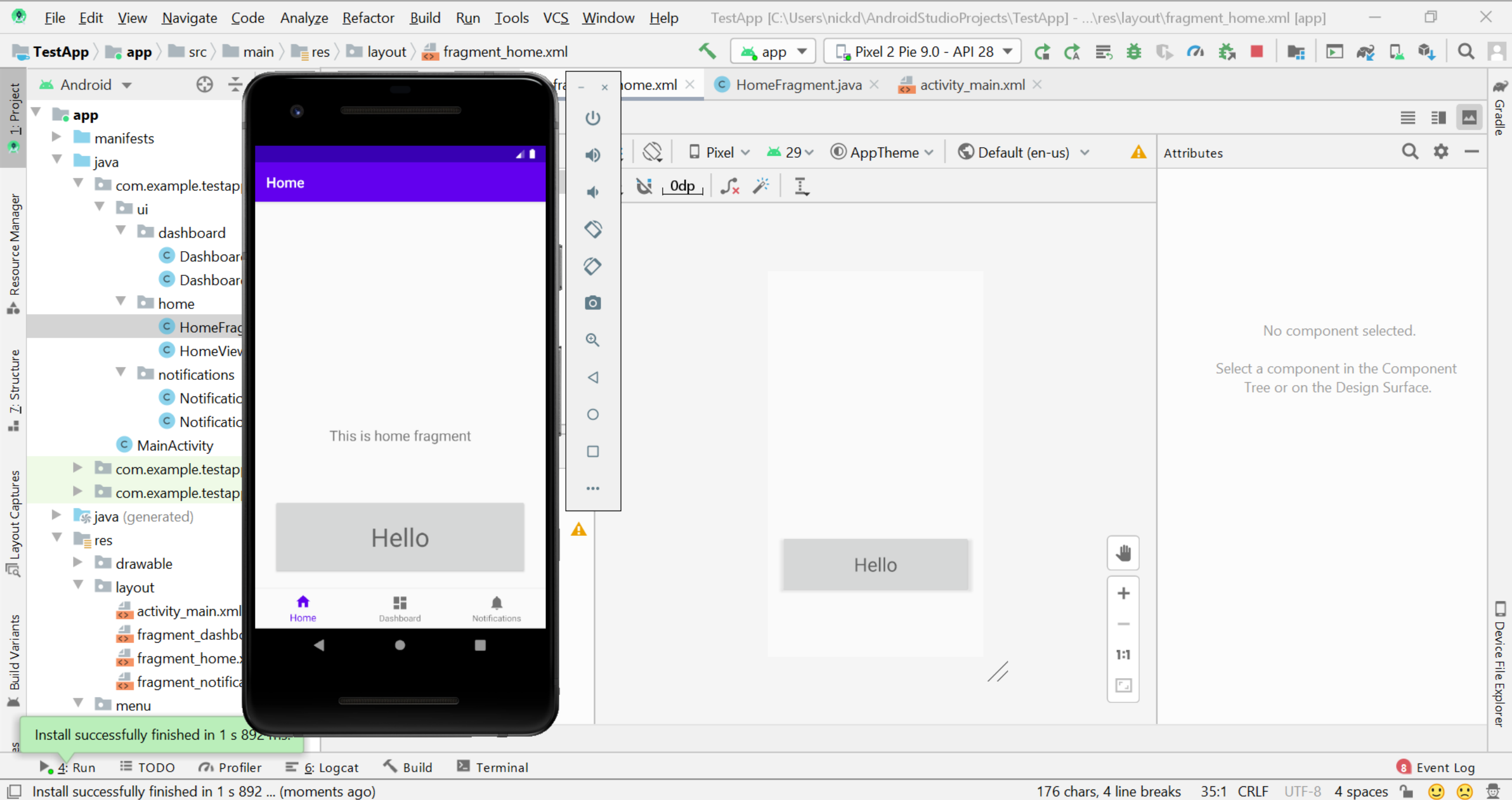
SDK Path: C:\Users\nickd\AppData\Local\Android\Sdk

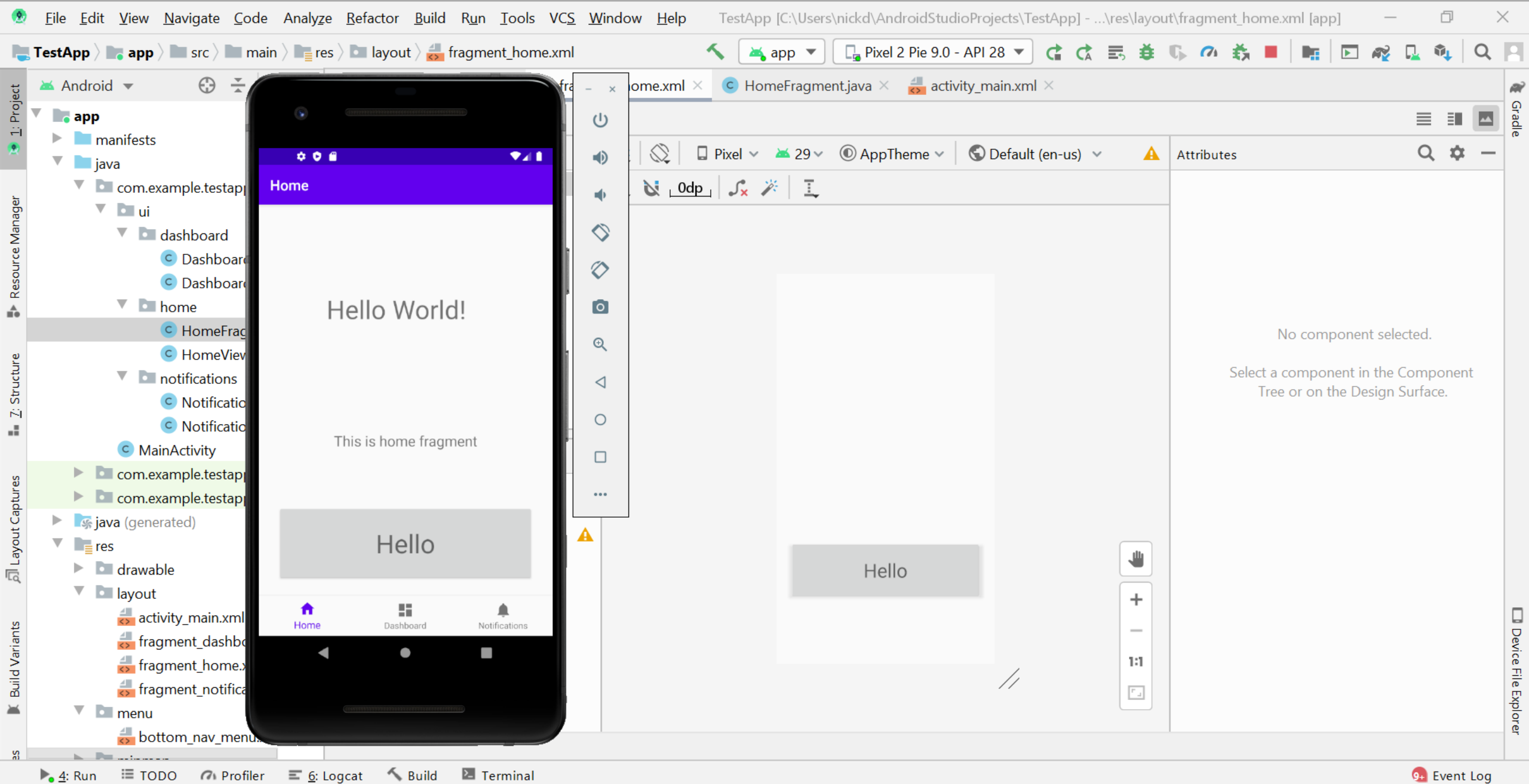
Packages to install:

- Google Play Intel x86 Atom System Image (system-images;android-28;google_apis_playstore;x86)

Preparing "Install Google Play Intel x86 Atom System Image (revision: 9)".

Downloading https://dl.google.com/android/repository/sys-img/google_apis_playstore/x86-28_r09.zip





SpringBoot / SpringMVC



Spring Initializr Java Support v0.9.0

Microsoft | 1,434,201 | ★★★★★ (5)

A lightweight extension based on Spring Initializr to generate quick start Spring Boot Java projects.

Disable Uninstall Switch to Pre-Release Version ⚙️

Details Feature Contributions Changelog

Spring Initializr Java Support

VS Marketplace v0.9.2022032503 installs 1.43M rating 3.4/5 (5)

Overview

Spring Initializr is a lightweight extension to quickly generate a Spring Boot project in Visual Studio Code (VS Code). It helps you to customize your projects with configurations and manage Spring Boot dependencies.

Feature List

- Generate a Maven/Gradle Spring Boot project
- Customize configurations for a new project (language, Java version, group id, artifact id, boot version and dependencies)
- Search for dependencies

Categories

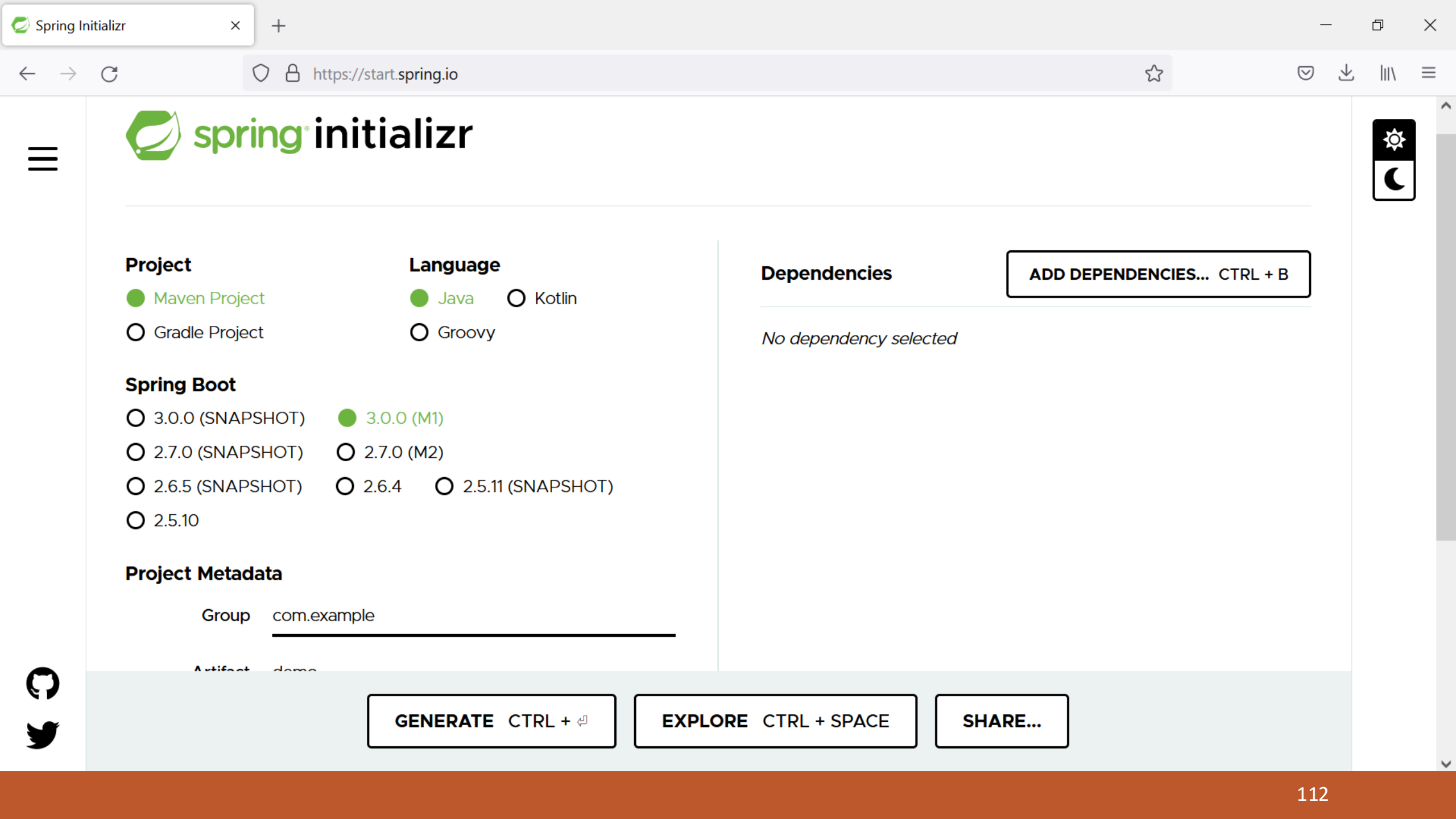
Other

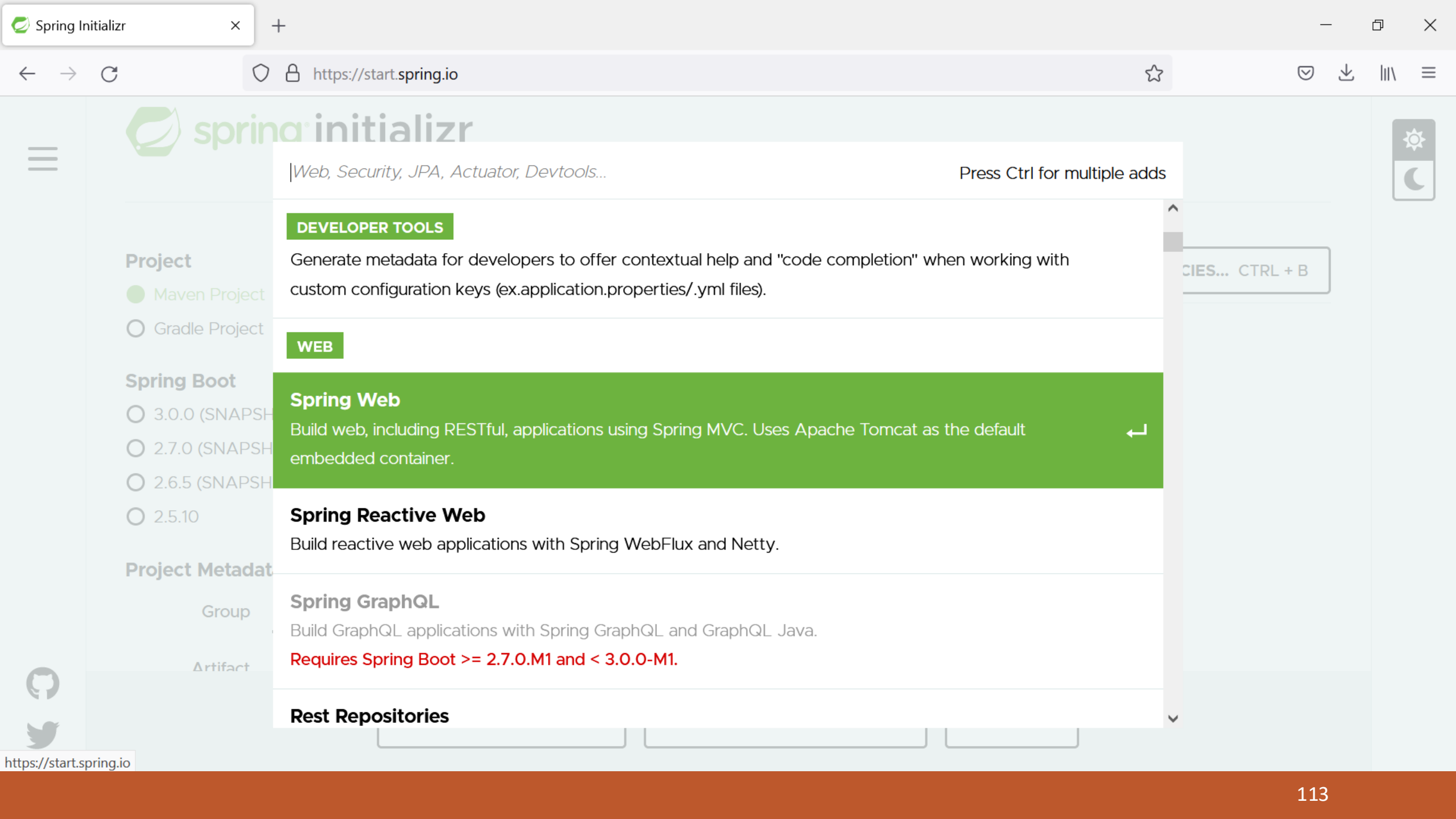
Resources

[Marketplace](#)
[Repository](#)
[License](#)
[microsoft.com](#)

Marketplace Info

Released on	17/01/2018, 07:11:18
Last updated	25/03/2022, 03:05:16
Identifier	vscjava.vscode-spring-initializr





|Web, Security, JPA, Actuator, Devtools...

Press Ctrl for multiple adds

DEVELOPER TOOLS

Generate metadata for developers to offer contextual help and "code completion" when working with custom configuration keys (ex.application.properties/.yml files).

WEB

Spring Web

Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Spring Reactive Web

Build reactive web applications with Spring WebFlux and Netty.

Spring GraphQL

Build GraphQL applications with Spring GraphQL and GraphQL Java.

Requires Spring Boot >= 2.7.0.M1 and < 3.0.0-M1.

Rest Repositories

CIES... CTRL + B

demo > src > main > java > com > example > demo > DemoApplication > hello

Project

- demo C:\Users\nickd\Desktop\demo
 - .idea
 - .mvn
 - src
 - main
 - java
 - com.example.demo
 - DemoApplication.java
 - resources
 - test
 - target
 - .gitignore
 - HELP.md
 - mvnw
 - mvnw.cmd
 - pom.xml
- External Libraries
- Scratches and Consoles

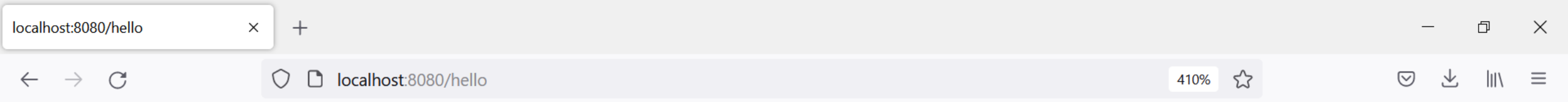
```

1 package com.example.demo;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5 import org.springframework.web.bind.annotation.GetMapping;
6 import org.springframework.web.bind.annotation.RequestMapping;
7 import org.springframework.web.bind.annotation.RestController;
8
9 @SpringBootApplication
10 @RestController
11 public class DemoApplication {
12     public static void main(String[] args) {
13         SpringApplication.run(DemoApplication.class, args);
14     }
15     @GetMapping("/hello")
16     public String hello(@RequestParam(value = "name", defaultValue = "World") String name) {
17         return String.format("Hello %s!", name);
18     }
19 }

```

Build: Sync x

✓ Sync: At 07/03/2022 13:48 2 min, 10 sec, 815 ms



Hello World!

FileEditViewNavigateCodeRefactorBuildRunToolsVCSWindowHelpdemo [C:\Users\nickd\Desktop\demo] - index.html

demo > src > main > resources > static > index.html

Project

demo C:\Users\nickd\Desktop\c

> .gradle

> .idea

> .mvn

> src

> main

> java

> resources

> static

index.html

templates

application.properties

> test

> target

.gitignore

HELP.md

mvnw

mvnw.cmd

pom.xml

> External Libraries

Scratches and Consoles

DemoApplication.java

index.html

1<!DOCTYPE HTML>

2<html>

3<head>

4<title>This is a HTML front-end for the Java logic</title>

5<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

6</head>

7<body>

8<h2> Enter Student ID: </h2>

9<input type = "text"/>

10<h2> Enter Student name: </h2>

11<input type = "text"/>

12<p>

13<input type = "button" value = "Sign in"/>

14</p>

15</body>

16</html>

html > head > title

Terminal: Local

Version ControlRunTODOProblemsTerminalBuildDependenciesEvent Log

Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built Maven library shared indexes // Always download // Download once // Don't show a... (28 minutes ago)4:59CRLFUTF-84 spaces

116

Enter Student ID:

Enter Student name:

Sign in