

# Morse Code

**Convert English word to Morse code or Morse code to English**

**<https://github.com/capacytron/morse-java>**

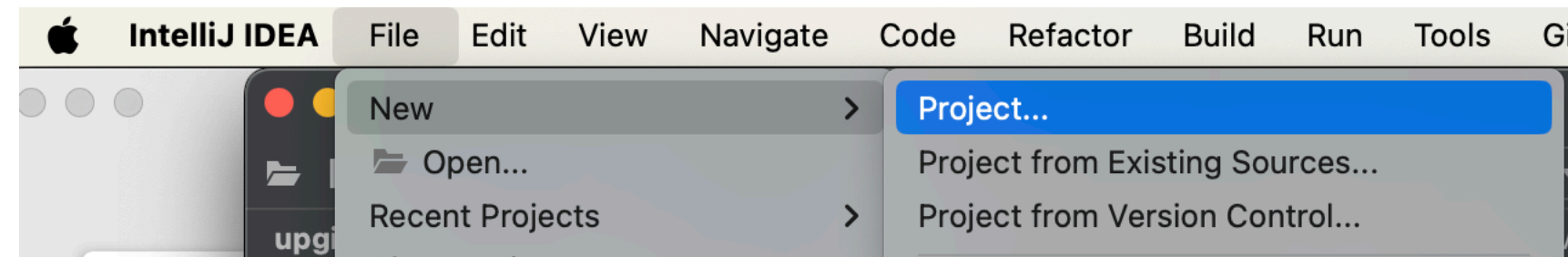
# Build systems

it helps to build complex software with multiple dependencies

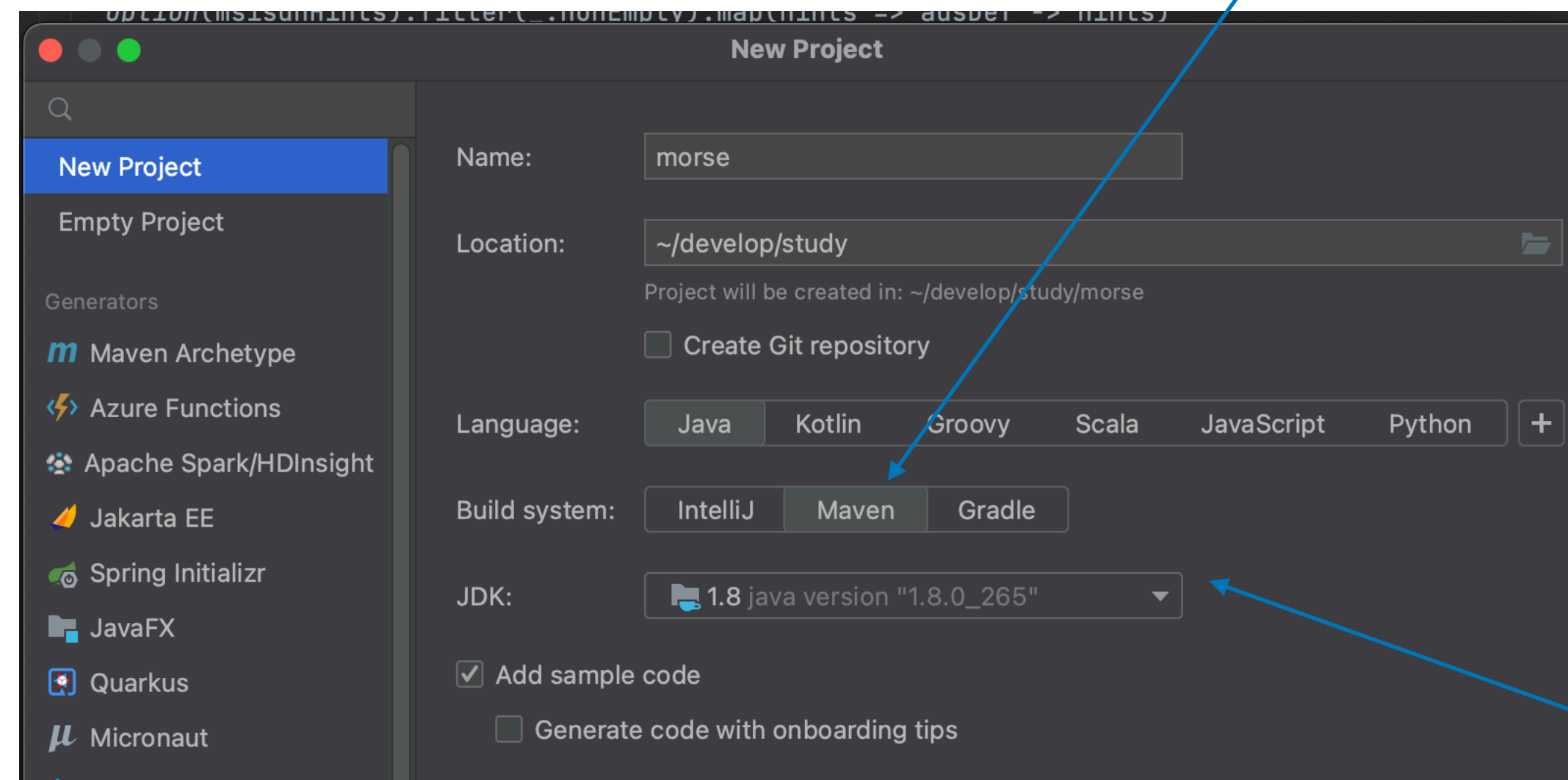
- They all have similar idea behind, but look quite different
  - think about Swedish, Danish, English. All these languages have nouns, verbs, adjectives, but look different
  - No matter what happens, all languages describe how objects (nouns) act (verbs) and look (adjectives)
- Most popular in java world:
  - Maven <https://maven.apache.org/>
  - Gradle <https://gradle.org/>

# Create Maven project

it's all about pom.xml (POM = Project Object Model)



Select Maven



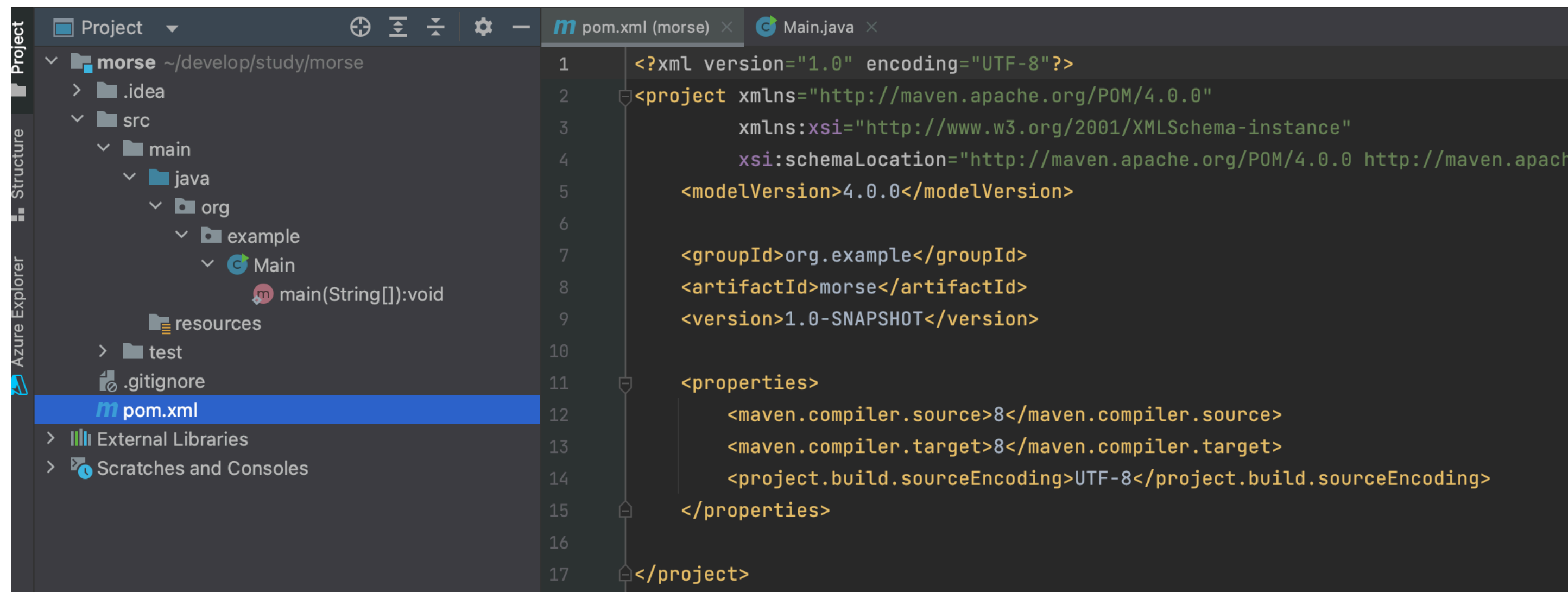
Set it at least to 11!

1.8 is ancient

Focus on Java 17

# Add dependencies to maven project

Do not invent wheels, just use them to drive



Default pom.xml that was created for you

groupId and artifactId make unique name of library

scope - tells build system where library is used

```
<dependency>
  <groupId>org.junit.jupiter</groupId>
  <artifactId>junit-jupiter-api</artifactId>
  <version>5.10.2</version>
  <scope>test</scope>
</dependency>
```

```
<dependencies>
  <!-- https://mvnrepository.com/artifact/commons-io/commons-io -->
  <!-- read from file -->
  <dependency>
    <groupId>commons-io</groupId>
    <artifactId>commons-io</artifactId>
    <version>2.15.1</version>
  </dependency>

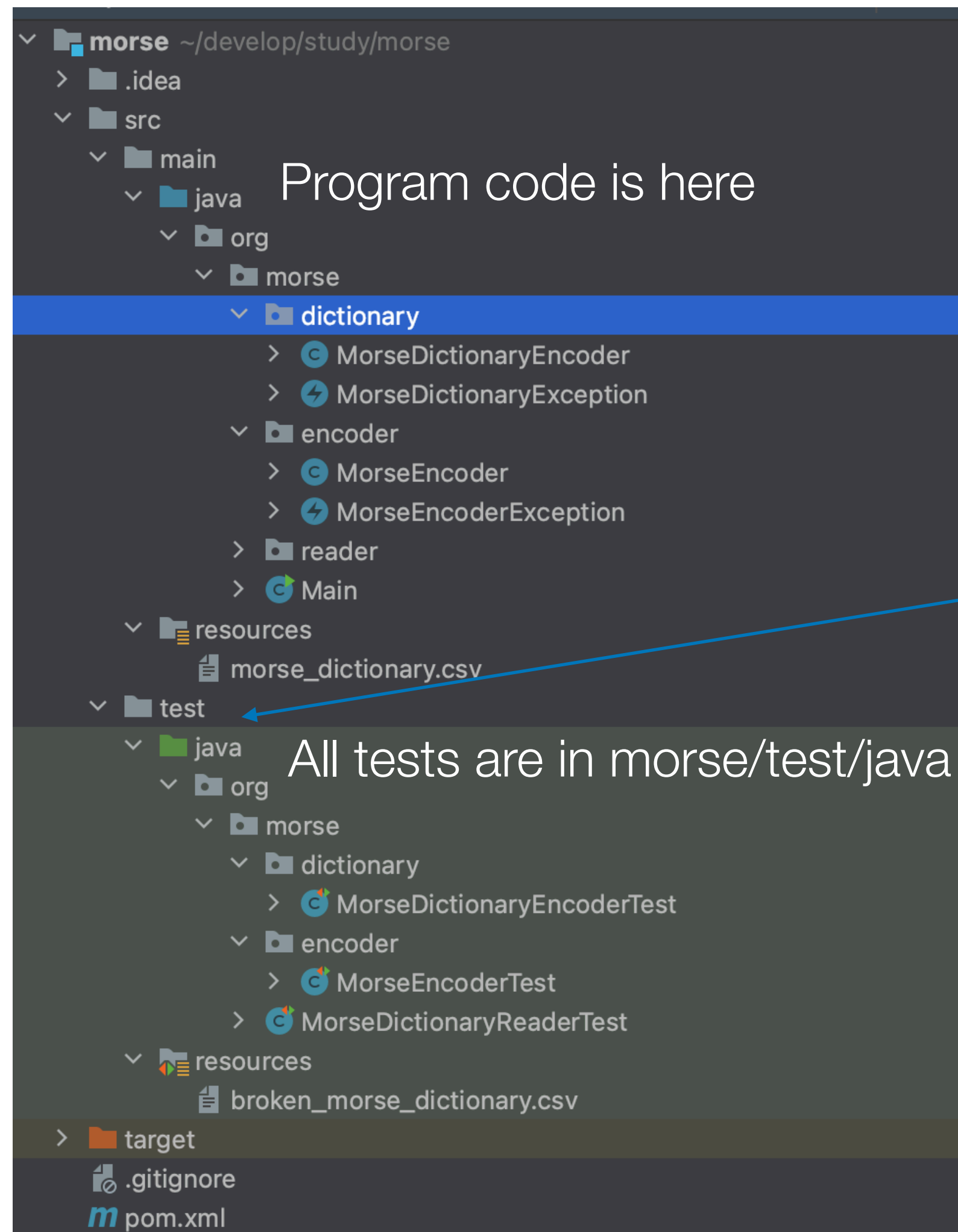
  <!-- https://mvnrepository.com/artifact/org.hamcrest/hamcrest -->
  <dependency>
    <groupId>org.hamcrest</groupId>
    <artifactId>hamcrest</artifactId>
    <version>2.2</version>
    <scope>test</scope>
  </dependency>

  <!-- https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-api -->
  <dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter-api</artifactId>
    <version>5.10.2</version>
    <scope>test</scope>
  </dependency>
</dependencies>
```

Add dependencies

# Follow code structure enforced by build system

## Separate tests from program code



```
<dependency>
  <groupId>commons-io</groupId>
  <artifactId>commons-io</artifactId>
  <version>2.15.1</version>
</dependency>
```

scope is not set, library available everywhere

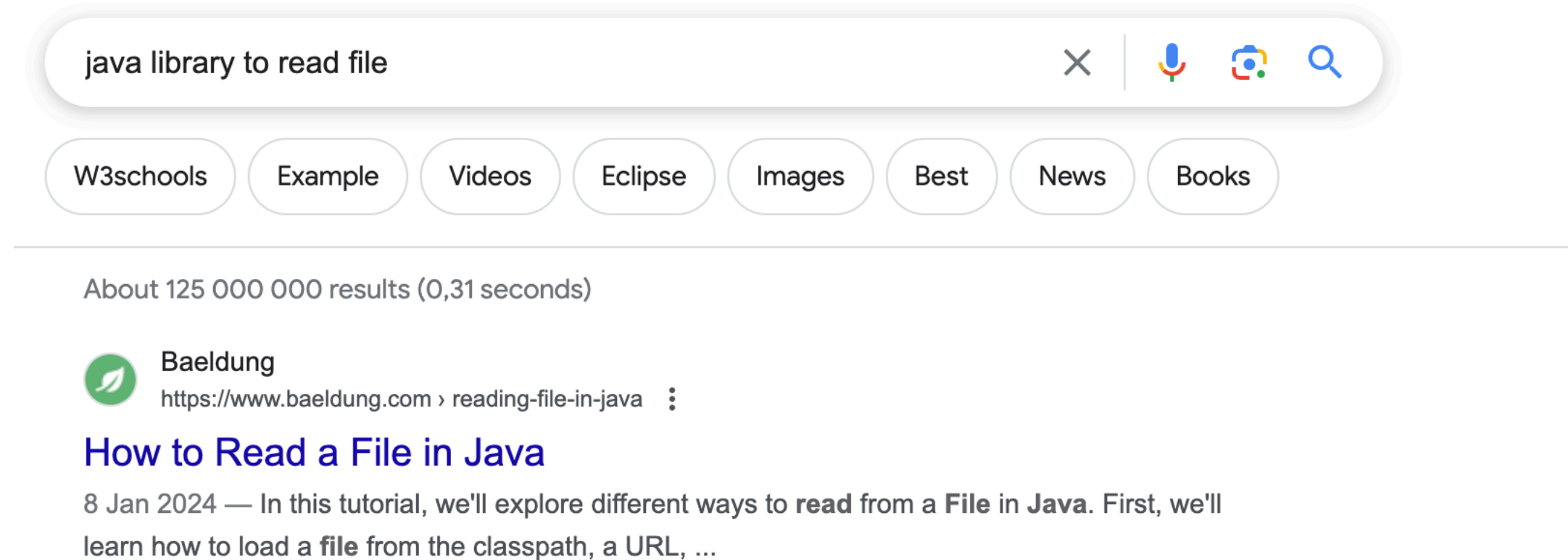
```
<dependency>
  <groupId>org.junit.jupiter</groupId>
  <artifactId>junit-jupiter-api</artifactId>
  <version>5.10.2</version>
  <scope>test</scope>
</dependency>
```

scope is 'test', library available only in tests



# How do I find library to solve my problem?

## google + SO (https://stackoverflow.com/)



Google, copy-paste don't be shy.  
it's fine!

### 3.2. Using the *commons-io* Library

Another common option is using the *FileUtils* class of the *commons-io* package:

```
@Test
public void givenFileName_whenUsingFileUtils_thenFileData() {
    String expectedData = "Hello, world!";

    ClassLoader classLoader = getClass().getClassLoader();
    File file = new File(classLoader.getResource("fileTest.txt").getFile());
    String data = FileUtils.readFileToString(file, "UTF-8");

    assertEquals(expectedData, data.trim());
}
```

Also try ChatGPT )  
It's fine too.

# Divide and conquer

**Split one complex problem into set of simple tasks**

- Complex problem: encode Morse into English and back
- Set of simple problems
  - ask user to input Morse or English word
  - encode Morse or English word
  - encode single English letter or Morse symbol
  - have a dictionary of English letters and Morse symbols somewhere

# Create program skeleton

## Define classes, methods, arguments, return types

A class that provides Dictionary of English letter to Morse Symbol mapping

```
class MorseDictionaryReader {  
    no usages  
    public Map<String, String> readEnglishToMorseDictionary() {  
        return null;  
    }  
}
```

Encode single English Letter to Morse symbol and back

```
class MorseDictionaryEncoder {  
    no usages  
    public MorseDictionaryEncoder(MorseDictionaryReader morseDictionaryReader) {  
    }  
  
    no usages  
    public String encodeEnglishLetterToMorse(String englishLetter) throws MorseDictionaryException {  
        return null;  
    }  
  
    no usages  
    public String encodeMorseSymbolToEnglish(String morseCode) throws MorseDictionaryException {  
        return null;  
    }  
}
```

It needs dictionary to do it

Encodes words

```
class MorseEncoder {  
    It needs class that encodes single letter  
    or symbol  
    no usages  
    public MorseEncoder(MorseDictionaryEncoder morseDictionaryEncoder) {  
    }  
  
    no usages  
    public String encodeEnglishWordToMorse(String englishWord) {  
        return null;  
    }  
  
    no usages  
    public String encodeMorseWordToEnglish(String morseWord) {  
        return null;  
    }  
}
```




# Software design patterns: Composition

Transform real life problems into structured program

- <https://refactoring.guru/design-patterns/composite>
- MorseDictionaryReader reads file with English letter to Morse mapping
- MorseDictionaryEncoder needs MorseDictionaryReader to get
  - A to .- mapping
  - and .- to A mapping
- MorseEncoder needs MorseDictionaryEncoder to encode letters to Morse and back
- `new MorseEncoder(new MorseDictionaryEncoder(new MorseDictionaryReader()));`
  - thing to explore: Dependency Injection

# Morse composition (induction thinking)

From reading file to encoding words, bottom to top

- 
- MorseEncoder, converts English <-> Morse words  
One more level up, split words into letters or symbols, encode entire word
  - MorseDictionaryEncoder, converts Letters <-> morse symbols  
One level up, use dictionary from file to encode single letters and symbols
  - MorseDictionaryReader, reads Morse alphabet from file  
The lowest level, read file, return Morse dictionary as Map which is convenient to use