Setting Up Spelling Corrector Unit Tests

Downloads

First you will need to visit the Projects section of the current CS 240 website and download the "Files (Zip)" folder as well as the "Test Files (Zip)" folder. Inside this "Test Files (Zip)" you will find 3 dictionary files and two folders labeled "test_files" and "jars".

Create a Project

Open IntelliJ and create a new Project.



This tutorial will be setting up as a normal Java project. If you wish to create a Maven or Gradle project you will be responsible for setting it up on your own and getting dependencies working correctly.

Select "Java" for your project type and in "Project SDK" select whatever version of Java you want (You should select whatever is the latest version of Java installed on your computer). You do not need to select any additional libraries or frameworks. Click "Next". Continue to click "Next" until you reach this window:

2	New Project	×		
Project name:	SpellingCorrector			
Project location:	~/SpellingCorrector			
 More Settings 				
	Previous Finish Cancel Help			

Name your project whatever you want though we suggest using SpellingCorrector. Click "Finish"

Now your project will build and will open:



Open the "Test Files (zip)" file you downloaded earlier and move the dictionary files and "jars" folder into the main folder of your project.



Right click on your src folder and add two new packages (New > Package). One needs to be called passoff, the other needs to be called spell.



Take all the files you downloaded from "Files (Zip)" and move them into the spell folder



Move the files from test_files into passoff



You'll see a lot of red errors, but that is okay for now.

Dependencies

The last step we need to do is add dependencies to your project. Since these tests run on JUnit 5 you need to have the files for JUnit 5 so that IntelliJ can access them and use them. The folder that is labeled "jars" is where all this code is stored as .jar files. So we will add dependencies to your project for these files so that your project knows where to go in order to access the code.

Project Structure Ľ Project name: SpellingCorrector Project Settin Project SDK: Modules This SDK is default for all project modules. Libraries A module specific SDK can be configured for each of the modules as required. Facets Edit 🗮 11 (1) (java version "11.0.4") Artifacts Project language level: **Platform Setti** This language level is default for all project modules. A module specific language level can be configured for each of the modules as required. Global Librar SDK default (11 - Local variable syntax for lambda parameters) Problems Project compiler output: This path is used to store all project compilation results. A directory corresponding to each module is created under this path. This directory contains two subdirectories: Production and Test for production code and test sources, respectively. A module specific compiler output path can be configured for each of the modules as required. /users/guest/w/westenm/SpellingCorrector/out OK Cancel

Select File > Project Structure (or use Ctrl+Alt+Shift+S) and you'll see this screen



Double-click "Modules" and click the tab labeled "Dependencies"

Now click the little "+" icon to the far right of the window and select "JARs or directories"



Navigate to the jars folder in your project and select all of the jar files then click "OK"



These will be added to the dependencies list. Click "OK"

(1)			Project Structure		×
← → Project Sett	+ – 🖻 SpellingCor	Name:	SpellingCorrector		
Project Modules Libraries		Source	s Paths Dependencies FindBugs-IDEA		
		Module	SDK: Project SDK (11 (1)) New Edit		
Facets Artifacts		Export		Scope	+
Platform Set		11 (1) (java version "11.0.4") Module source>		-	
Global Libr			II junit-5.0.jar (/users/guest/w/westenm/SpellingCorrector/jars)	Compile *	
Problems			 junit-jupiter-api-5.2.0.jar (/users/guest/w/westenm/SpellingCorrector junit-jupiter-engine-5.2.0.jar (/users/guest/w/westenm/SpellingCorre junit-platform-commons-1.2.0.jar (/users/guest/w/westenm/Spellingr junit-platform-engine-1.2.0.jar (/users/guest/w/westenm/SpellingCor junit-platform-launcher-1.2.0.jar (/users/guest/w/westenm/SpellingCor 	Compile + Compile + Compile + Compile + Compile +	
		Depende	encies storage format: Intelli JIDEA (iml) 🚽		
?			OK Cancel	Apply	

All of the errors should be gone now except for one related to *new SpellCorrector();* in SpellTest and two related to *new Trie();* in TrieTest. We'll fix those now by creating these classes.

Right click on your spell package and select New > Java Class and name it SpellCorrector.



Now we'll make it implement the interface file that we have. After *public class SpellCorrector* write *implements ISpellCorrector*



You'll see an error, click on the red lightbulb or click Alt+Enter to pull up suggestions. One of the suggestions will be "Implement Methods" click this and select all the suggested methods



IntelliJ will automatically generate the needed code



The error in SpellTest will now be gone

Now lets do the same thing for Trie Test with ITrie.

Create a new Java class and call it Trie. Type in *implements ITrie* and have IntelliJ autogenerate the needed methods.



All of the errors will now be gone in TrieTest and SpellTest. Now right click on SpellTest or TrieTest and select "Run Test" or click on the green arrow next to the class declaration in the file itself.

🔻 🖿 passoff		8 import spell.Sp
🕝 Spell	Pect	9
😅 Trie	New	· · · · · · · · · · · · · · · · · · ·
🔻 🖿 spell 🕺	(Cu <u>t</u>	Ctrl+X
🙂 INo 🖷	I <u>С</u> ору	Ctrl+C
😨 ISp	C <u>o</u> py Path	Ctrl+Shift+C
inTI 😳	Copy Reference	Ctrl+Alt+Shift+C
🥶 Mai 🗖	<u>P</u> aste	Ctrl+V
Spe	Find <u>U</u> sages	Alt+F7
	Analy <u>z</u> e	•
	FindBugs	•
a word.txt	<u>R</u> efactor	•
🖞 words.tx	Add to F <u>a</u> vorites	•
IIII External Lit	Browse Type Hierarchy	Ctrl+H
🌄 Scratches a	<u>R</u> eformat Code	Ctrl+Alt+L
	Optimi <u>z</u> e Imports	Ctrl+Alt+O
	<u>D</u> elete	Delete
	Build <u>M</u> odule 'SpellingCorre	ector'
	R <u>e</u> compile 'SpellTest.java'	Ctrl+Shift+F9
	R <u>u</u> n 'SpellTest'	Ctrl+Shift+F10
Ŭ	<u>D</u> ebug 'SpellTest'	
G	Run 'SpellTest' with Co <u>v</u> erag	je .
ଙ୍କ	Run 'SpellTest' with 'CPU Pro	ofiler'
	Run 'SpellTest' with 'Allocat	ion Profiler'
ଙ୍କ	Run 'SpellTest' with 'Java Fli	ight Recorder'
•	Save 'SpellTest'	
	Show in Caja	
Run: 🐠 SpellT	File <u>P</u> ath	Ctrl+Alt+Shift+2
▶ ✓ Ø ↓ª 🖥	Open in Terminal	
🥵 🔻 😣 Test Re	Local <u>H</u> istory	
👸 🔻 😣 Spel 🛇	Synchronize 'SpellTest.java'	
💛 E 🚽	Compare With	Ctrl+D
8 E	Run PMD	•
🔍 🕺 E 🖬	Diagrams	•
	Create Gist	
N E	Convert Java File to Kotlin F	ile Ctrl+Alt+Shift+K
12 Jampor	t static are junit i	unitor ani Accortions *
12 etilipot	t statte org.junit.j	upiter.api.Assertions.
11 0 alare	CoollToot (
📜 🕨 Run 'S	SpellTest'	Ctrl+Shift+F10
15 📥 Dobu		ME.
17 5 Due 17		17/E =
18 Kun S	spell rest with Covera	age IE = "
19 📽 Run 'S	SpellTest' with 'CPU P	rofiler'
20 🖡 Run 'S	SpellTest' with 'Alloca	ition Profiler'
21 🚱 Run 'S	SpellTest' with 'Java F	Flight Recorder'
22 🤅	BetoreEach	
23 👳 🕴	<pre>/oid setup(){</pre>	
24 👳	try{	
25	studentCorrec	tor = new SpellCorrecto
26 卓		
27 👳	catch(Throwable t){
28	fail(t.getCla	ss() + ". Make sure cla
29 🍦		
30 0 3		

This will run the tests. The tests will obviously fail because you have not written your code yet, but now you have the tests all ready to go so you can test as you code.