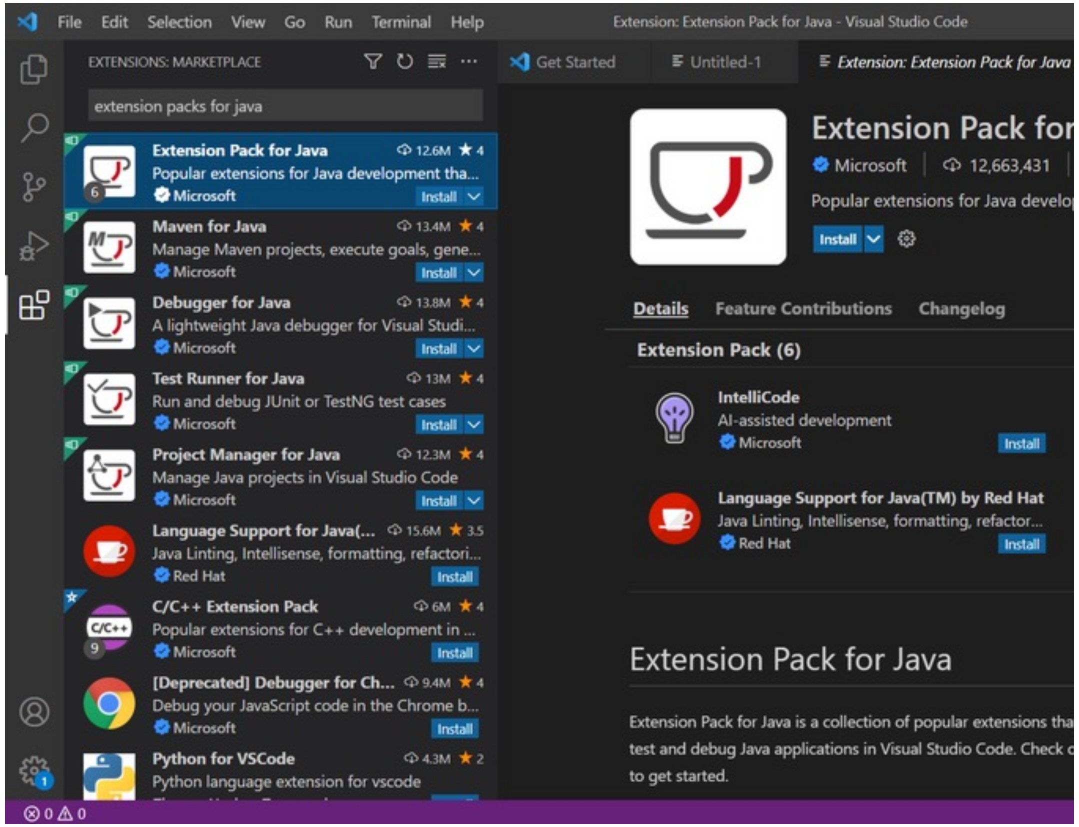
UNIT TESTING LAB

2024/2

JUNIT ON VISUAL STUDIO CODE



Extension Pack for Java 12.6M ★4
Popular extensions for Java development tha...
Microsoft

Maven for Java 13.4M ★4
Manage Maven projects, execute goals, gene...
Microsoft

Debugger for Java 13.8M ★4
A lightweight Java debugger for Visual Studi...
Microsoft

Test Runner for Java 13M ★4
Run and debug JUnit or TestNG test cases
Microsoft

Project Manager for Java 12.3M ★4
Manage Java projects in Visual Studio Code
Microsoft

Language Support for Java(... 15.6M ★3.5
Java Linting, Intellisense, formatting, refactori...
Red Hat

C/C++ Extension Pack 6M ★4
Popular extensions for C++ development in ...
Microsoft

[Deprecated] Debugger for Ch... 9.4M ★4
Debug your JavaScript code in the Chrome b...
Microsoft

Python for VSCode 4.3M ★2
Python language extension for vscode

Extension Pack for
Microsoft | 12,663,431
Popular extensions for Java develo

IntelliCode
AI-assisted development
Microsoft

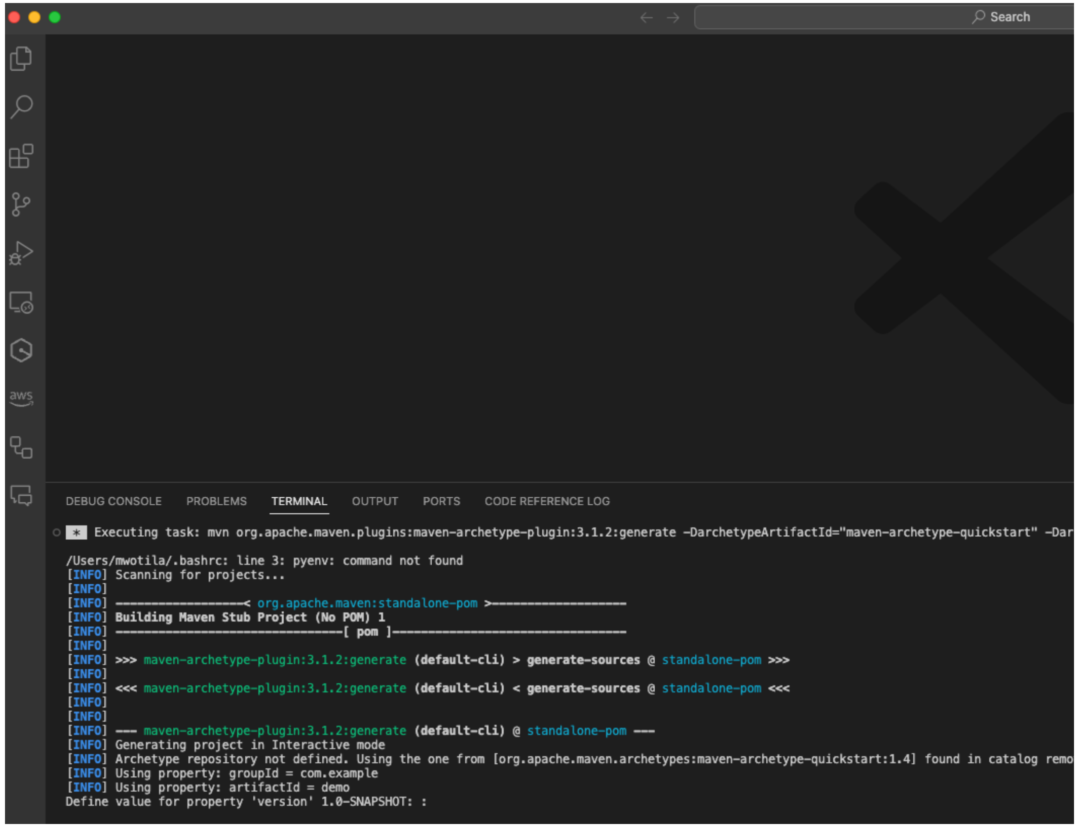
Language Support for Java(TM) by Red Hat
Java Linting, Intellisense, formatting, refactor...
Red Hat

Extension Pack for Java

Extension Pack for Java is a collection of popular extensions tha test and debug Java applications in Visual Studio Code. Check to get started.

- Download & Install Visual Studio (VS) code & Java
  - VS - https://code.visualstudio.com/Download
  - Java - https://www.java.com/en/download/
- Create a project – You can use your student number
  - 21XXXXXXX
- Open the VS code application
  - click the Extensions icon (Ctrl + Shift + X) on the left then type in "Extension Pack of Java" and install it by clicking the install button.

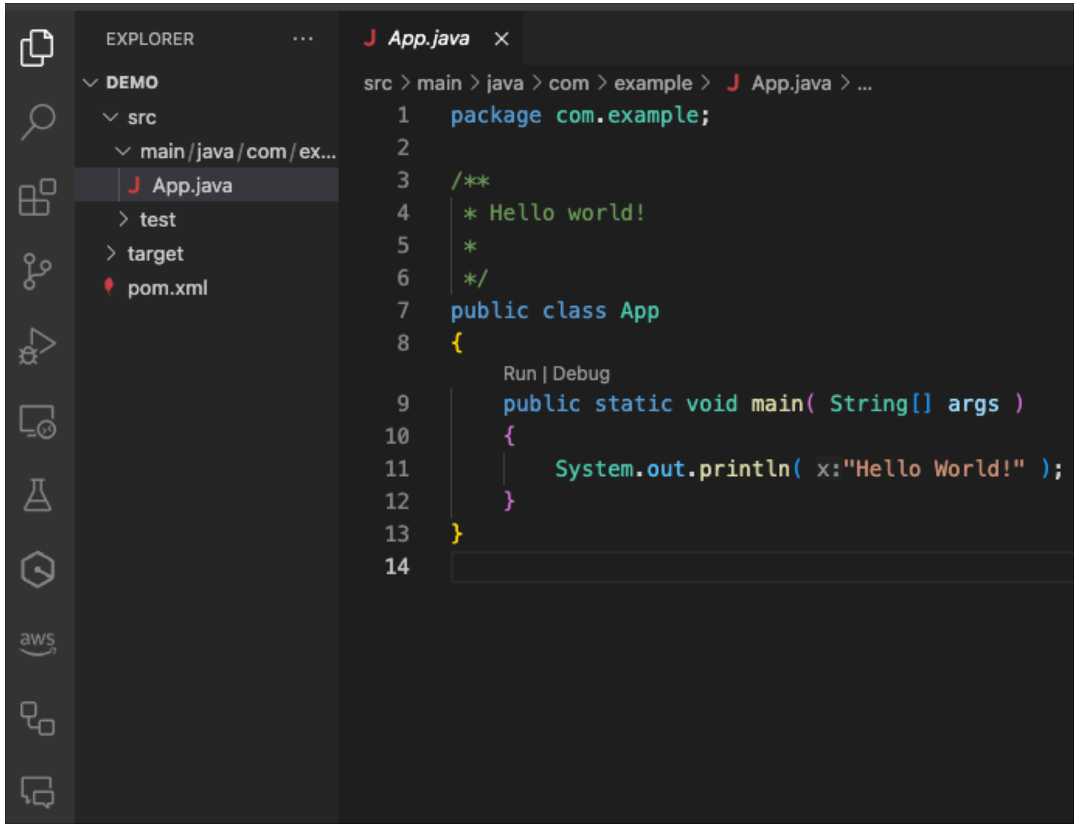
JUNIT ON VISUAL STUDIO CODE



```
* Executing task: mvn org.apache.maven.plugins:maven-archetype-plugin:3.1.2:generate -DarchetypeArtifactId="maven-archetype-quickstart" -Dar

/Users/mwotila/.bashrc: line 3: pyenv: command not found
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------< org.apache.maven:standalone-pom >-------------------
[INFO] Building Maven Stub Project (No POM) 1
[INFO] --------------------------------[ pom ]---------------------------------
[INFO]
[INFO] >>> maven-archetype-plugin:3.1.2:generate (default-cli) > generate-sources @ standalone-pom >>>
[INFO]
[INFO] <<< maven-archetype-plugin:3.1.2:generate (default-cli) < generate-sources @ standalone-pom <<<
[INFO]
[INFO]
[INFO] --- maven-archetype-plugin:3.1.2:generate (default-cli) @ standalone-pom ---
[INFO] Generating project in Interactive mode
[INFO] Archetype repository not defined. Using the one from [org.apache.maven.archetypes:maven-archetype-quickstart:1.4] found in catalog remo
[INFO] Using property: groupId = com.example
[INFO] Using property: artifactId = demo
Define value for property 'version' 1.0-SNAPSHOT: :
```

- Open View > Command Palette (Ctrl + Shift + P) and type in ">Maven: Create Maven Project”, click it then click “maven-archetype-quickstart” and version “1.4."
- Afterward, name the group ID and artifact ID and save the project to a folder.
  - Group ID: ug.ac.mak.se23day
  - Artifact ID: p21xxxxxxxx
- Press Enter to proceed through the Terminal prompts then, open your newly created project.

JUNIT ON VISUAL STUDIO CODE

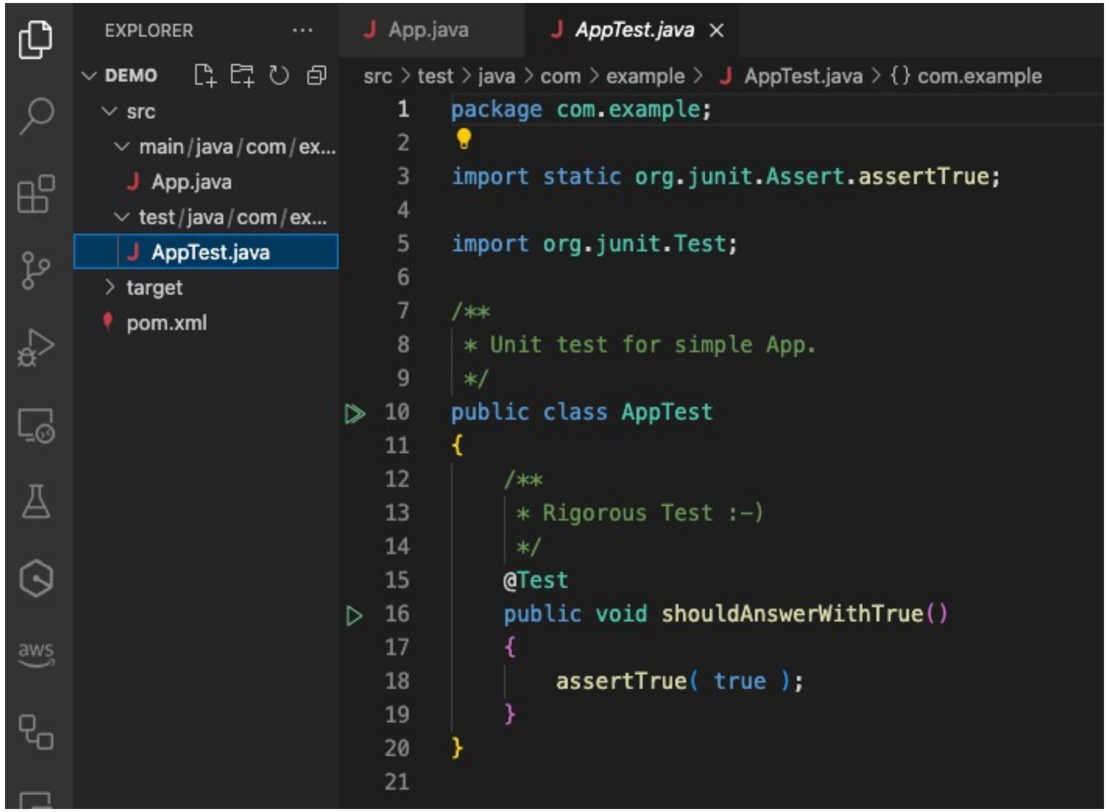

EXPLORER
DEMO
src
main / java / com / ex...
App.java
test
target
pom.xml
App.java
src > main > java > com > example > App.java > ...
Run | Debug

```java
package com.example;

/**
 * Hello world!
 *
 */
public class App
{
    public static void main( String[] args )
    {
        System.out.println( x:"Hello World!" );
    }
}
```

```java
package ug.ac.mak.seday;

public class App {
    public static int evaluate(String expression) {
    int sum = 0;
    for (String summand: expression.split("\\+"))
      sum += Integer.valueOf(summand);
    return sum;
    }
}
```

Replace the code in App.java accordingly

JUNIT ON VISUAL STUDIO CODE


EXPLORER
DEMO
src
main / java / com / ex...
App.java
test / java / com / ex...
AppTest.java
target
pom.xml
App.java
AppTest.java
src > test > java > com > example > AppTest.java > {} com.example
aws

```java
package com.example;

import static org.junit.Assert.assertTrue;

import org.junit.Test;

/**
 * Unit test for simple App.
 */
public class AppTest
{
    /**
     * Rigorous Test :-)
     */
    @Test
    public void shouldAnswerWithTrue()
    {
        assertTrue( true );
    }
}
```

```java
package com.example;

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

public class AppTest {
    @Test
    public void evaluateTest(){
        int sum = App.evaluate("2+3+4");
        assertEquals(9, sum);
    }
}
```

Replace the code in AppTest.java accordingly

ENTER… THE PLANNING SYSTEM

- Everybody likes meetings?.
  - Not true - but we need to book them.
- We don’t want to double-book rooms or employees for meetings.
- System to manage schedules and meetings.

THE PLANNING SYSTEM

Offers the following high-level features:

1. Booking a meeting
2. Booking vacation time
3. Checking availability for a room
4. Checking availability for a person
5. Printing the agenda for a room
6. Printing the agenda for a person

DEVELOP A TEST PLAN

In your groups, come up with a test plan for this system.

- Given the features and the code documentation, plan unit tests to ensure that these features can be performed without error.

FOOD FOR THOUGHT

- Try running the code!
  - Perform exploratory testing to test it at the system level.
- Think about normal and erroneous inputs/actions.
  - How many things can go wrong?
  - You will probably be able to add a normal meeting, but can you add a meeting for February 35th?
  - Try it out - you have the code.

DEVELOP UNIT TESTS

- If a test is supposed to cause an exception to be thrown, make sure you check for that exception.
- Make sure that expected output is detailed enough to ensure that - if something is supposed to fail - that it fails for the correct reasons.
  - Use proper assertions.

CAN YOU EXPOSE THE FAULTS?

1: getMeeting and removeMeeting perform no error checking on dates.

```
public Meeting getMeeting(int month, int day, int index){ return
        occupied.get(month).get(day).get(index);
}

public void removeMeeting(int month, int day, int index){
        occupied.get(month).get(day).remove(index);
}
```

CAN YOU EXPOSE THE FAULTS?

2: Calendar has a 13th month.

```
public Calendar(){
        occupied = new ArrayList<ArrayList<ArrayList<Meeting>>>();

        for(int i=0;i<=13;i++){
            // Initialize month
            occupied.add(new  ArrayList<ArrayList<Meeting>>()); for(int
            j=0;j<32;j++){
                // Initialize days
                occupied.get(i).add(new  ArrayList<Meeting>());
            }
        }
}
```

CAN YOU EXPOSE THE FAULTS?

3: November has 30 days.

- Oh - and we just added a meeting to a day with a date that does not match that date.

- occupied.get(11).get(**30**).add(new Meeting(11,**31**,"Day does not exist"));

CAN YOU EXPOSE THE FAULTS?

4: Used a >= in checking for illegal times. December no longer exists.

```
if(mMonth < 1 || mMonth >=  12){
                    throw new TimeConflictException("Month does not
exist.");
}
```

CAN YOU EXPOSE THE FAULTS?

5:We should be able to start and end a meeting in the same hour.

```
if(mStart >=  mEnd){
      throw new TimeConflictException("Meeting starts before it ends.");
}
```

What Other Faults Can You Find?