Dependency Manager

Release Notes

Author	Adam Trcka
Reviewer	N/A
SWQA	N/A
Approver	N/A

Document Version control

Version	Date	Authors	Notes
1.0	4.5.2021	Adam Trcka	Initial Issue

Table of Contents

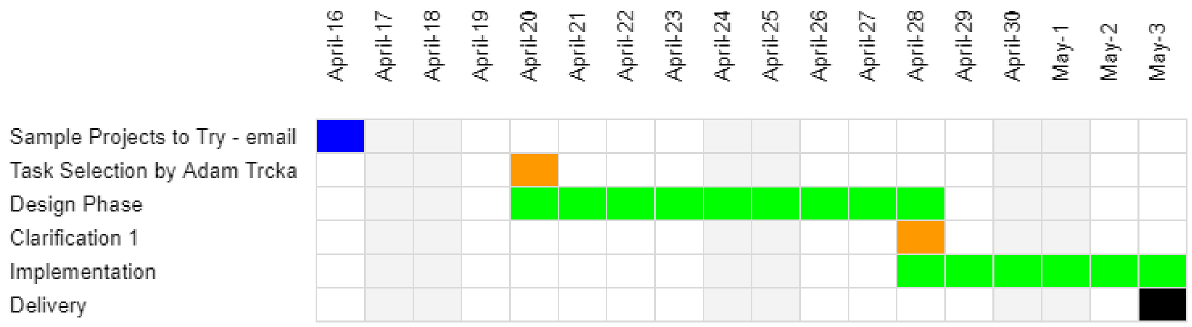
1. Introduction

This SW was generated for evaluation purposes only. It was a commissioned by:

"Fach, Brad" <bfach@netsuite.com>
"Devitt, Stan" <sdevitt@netsuite.com>

For the purposes of evaluating the general coding status and skills of a potential candidate to join their development team.

1.1. Project Summary



1.2. Delivery Package Identification

Name: newdm.zip

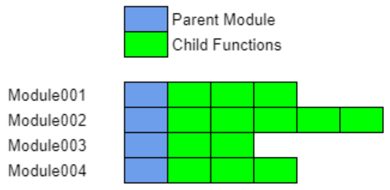
Availability: https://github.com/sadamski/netsuiteDemo

2. Product Overview

2.1. Design Requirements

There is 1 Customer requirement spec, and 1 clarification existing. These are available in the ANNEXES. The following is a table of derived requirements, based on those specs:

Requirement ID	Descriptions	Remarks/justification	Verification Strategy
REQ-001	The SW shall used input as a JSON		UnitTest
REQ-002	The JSON shall be specified as an argument passed into the application	Derived - either it could a) Hardcoded b) be specified at runtime c) Passed as a parameter c) was chosen as default option	UnitTest

REQ-003	The JSON shall have the following structure:
	{
	Element key (string) : [array of strings],
	Element key (string) : [array of strings],
	...
	}
	Graphical concept (FYI only):
	
	 Parent Module
	Child Functions
	Module001
	Module002
	Module003
	Module004
		Per input spec ANNEX 1	Validation Test (see ANNEX 3)

REQ-004	There shall be a simple forward discovery readout of the JSON data	Per input spec ANNEX 2 Part 1	UnitTest
REQ-005	There shall be a reverse discovery readout of the JSON data, such that each child data point shall be listed, and its effective higher level parent(s) shall be displayed	Per input spec ANNEX 2 Part 2	UnitTest
REQ-006	The build system shall be GRADLE	Per input spec ANNEX 1, suggested but not hard requirement. Gradle Accepted for ease of use	Review
REQ-007	The main outputs off the SW shall be displayed in the console	Assumed to be powershell in the IDE (VS Code)	Review

2.2. SW Architecture

Not required at this time by the customer.

3. Installation and Deployment

3.1. Dependencies and Prerequisites for deployment

The following installations are required prior to running the program:

Tool / Package	Purpose	Mandatory / Optional	Version
Java	Programming language and runtime environment	Mandatory	16 or higher
gradle	Gradle is a build automation tool for multi-language software development.	Mandatory	7.0 or higher
Visual Studio Code	General purpose IDE, alternates such as Eclipse also possible	Optional	Any
Winzip	General delivery, but also used to inspect *.war* files	Optional	Or any equivalent

Its notes that the gradle build system pulls in all defined plugins and prerequisite packages. During each build (execution), gradle shall check in the background that all required packagers are available prior to compilation

3.2. SW Release Package

The following the is contents of the delivered SW package:

Name	Purpose
.gradle	gradle tool folder
.idea	gradle tool folder
app	**main directory with working code** + **Contains the *gradle.build* script**
gradle	gradle build folder
lib	gradle build folder
.gitattributes	GIT repo hidden files
.gitignore	GIT repo hidden files
gradlew	gradle tool file
gradlew.bat	Windows batch file (not used)
settings.gradle	project wide dependency declarations (not used)

3.3. Sample JSONs

The following JSONs are additionally delivered, to be used as a part of the evaluation of the SW:

Name	Purpose	Gradle Run Command
originalSpec.json	A structured JSON, based on the original customer specification, per requirement REQ-001	`gradle run --args originalSpec.json`
testSmall.json	A tiny JSON user for simple human testing to see if program is working correctly	`gradle run --args testSmall.json`
testLarge.json	An arbitrary large and complicated JSON to see stability, even if multiple looped dependencies exist, causes a crash because of duplicate keys	`gradle run --args testLarge.json`
testLarge2.json	An arbitrary large and complicated JSON to see stability, even if multiple looped dependencies exist.	`gradle run --args testLarge2.json`

They are available in `\app\src\main\resources`

3.4. Deployment

The SW app can be build in 1 main step. In the root folder, run the following universal gradle command:

```
> gradle clean build
```

Where `clean` is a optional command to clear/delete existing build folders (this is a best practice, but not necessary), and `build` executes the groovy script, as defined in the *app/gradle.build* file

Since both *jar* and *war* plug-ins are imported in the *app/gradle.build* file, executing the above build scrip will generate both .jar and .war files.
They are available in `\app\build\libs`
Where their filenames shall be:

app-<project version number in */gradle.build*>.jar
app-<project version number in */gradle.build*>.war

3.5. Execution

Once the main build process is complete per the section above, the user can now use the SW. Its important to note, that the SW requires and input JSON file to passed as an argument (requirement REQ-001)

3.5.1. Execution from gradle

The following gradle command can be used to execute the application

```
gradle run --args <JSON filename.json>
```

Where `--args` is the native gradle indicator that the user will pass an argument. If this is not used, gradle will assume the user is instead calling a custom task in the *gradle.build* script

Example:

```
gradle run --args originalSpec.json
```

3.5.2. Execution from java / command prompt

The following gradle command can be used to execute the application

```
java -cp app-1.0.jar com.adamtrcka.work.App <JSON filename>.json
```

Where `--args` is the native gradle indicator that the user will pass an argument. If this is not used, gradle will assume the user is instead calling a custom task in the *gradle.build* script

Example: To run, the following command in your console:

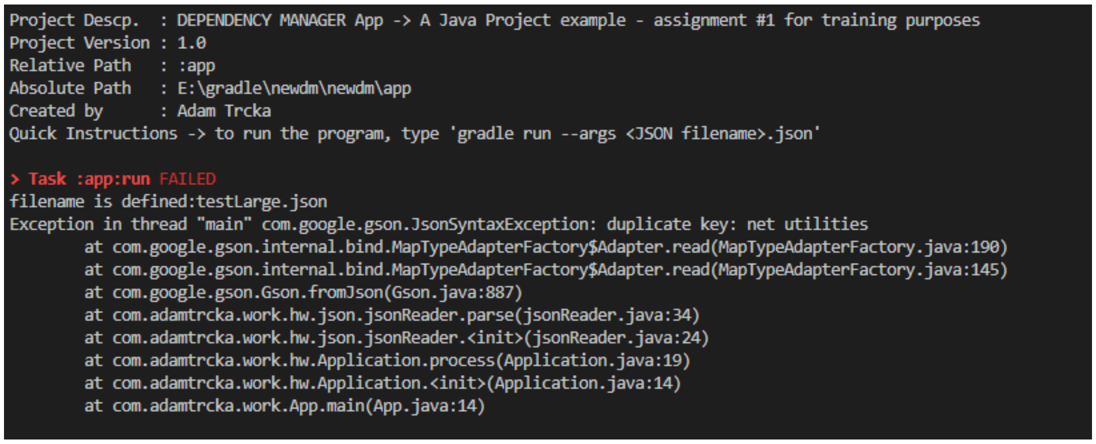
```
java -cp app-1.0.jar com.adamtrcka.work.App E:\originalSpec.json
```

3.6. Known Defects[1]

Ticket : artfactXXXX - duplicate keys cause system to crash:

Inputs: testLarge.json

Actual Outputs:



```
Project Descp.  : DEPENDENCY MANAGER App -> A Java Project example - assignment #1 for training purposes
Project Version : 1.0
Relative Path   : :app
Absolute Path   : E:\gradle\newdm\newdm\app
Created by      : Adam Trcka
Quick Instructions -> to run the program, type 'gradle run --args <JSON filename>.json'

> Task :app:run FAILED
filename is defined:testLarge.json
Exception in thread "main" com.google.gson.JsonSyntaxException: duplicate key: net utilities
        at com.google.gson.internal.bind.MapTypeAdapterFactory$Adapter.read(MapTypeAdapterFactory.java:190)
        at com.google.gson.internal.bind.MapTypeAdapterFactory$Adapter.read(MapTypeAdapterFactory.java:145)
        at com.google.gson.Gson.fromJson(Gson.java:887)
        at com.adamtrcka.work.hw.json.jsonReader.parse(jsonReader.java:34)
        at com.adamtrcka.work.hw.json.jsonReader.<init>(jsonReader.java:24)
        at com.adamtrcka.work.hw.Application.process(Application.java:19)
        at com.adamtrcka.work.hw.Application.<init>(Application.java:14)
        at com.adamtrcka.work.App.main(App.java:14)
```

Expected Outputs: SW does not crash.

[1] *N/A This section is typically included on ongoing SW releases. However is N/A for this release.*

4. Testing

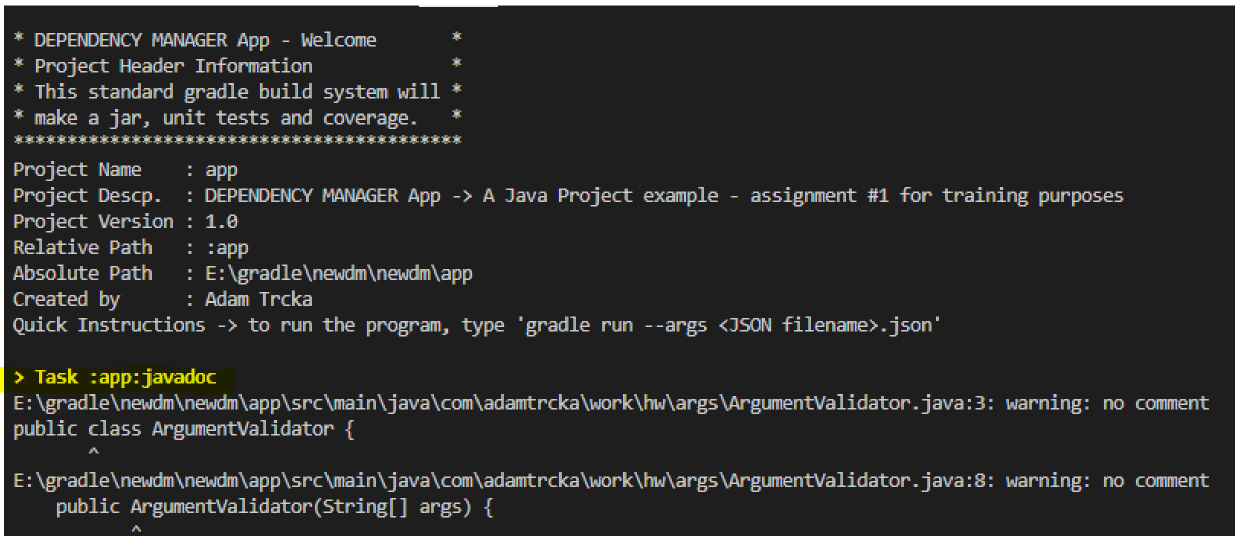
The SW package comes with a pre-loaded basic test coverage. Each test level is automatically triggered, unless otherwise indicated.

4.1. Automatic Documentation

The build in command, *javadoc*, is coded into the *build.gradle* script, such that each build triggers this automatic documentation. It can however be manually requested by performing using the following gradle command line:

```
> gradle javadoc
```

The resulting HTLM docs are stored in: `app\build\docs\javadoc`
If invoked, it will list in the terminal common issues such as non-commented functions:



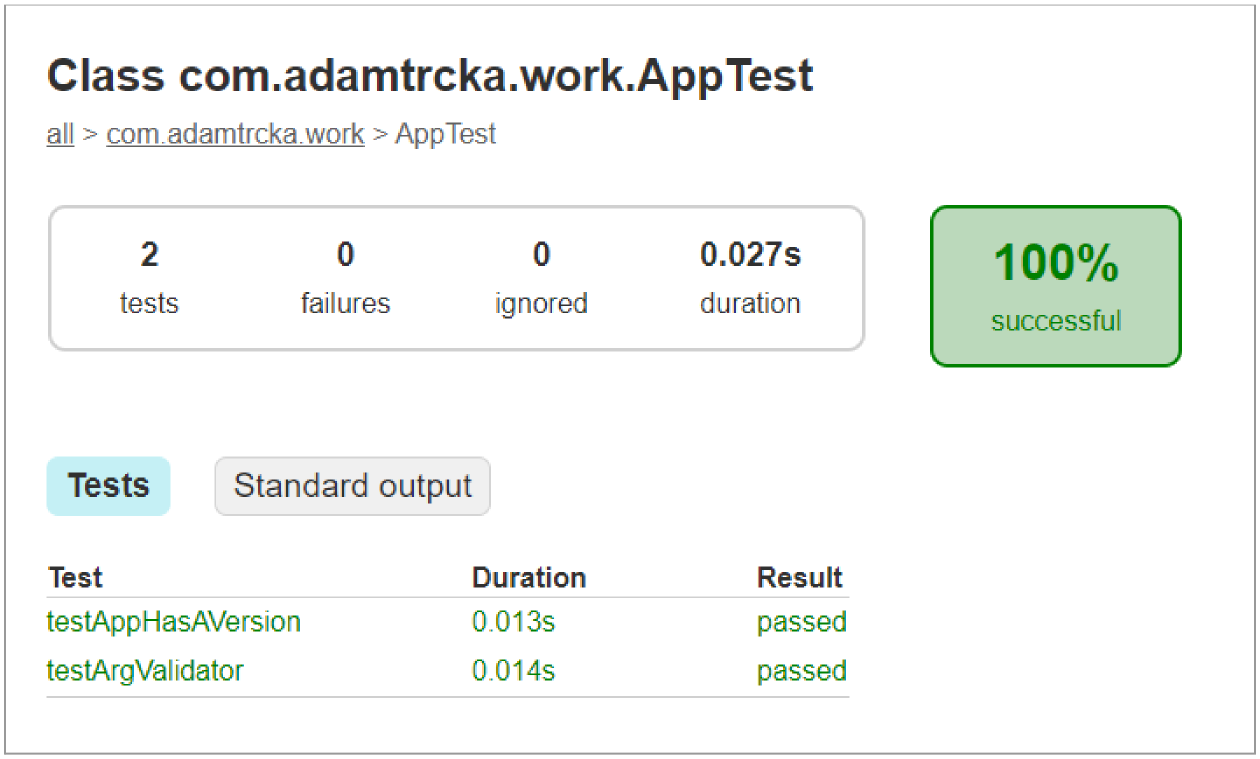
```
* DEPENDENCY MANAGER App - Welcome      *
* Project Header Information            *
* This standard gradle build system will *
* make a jar, unit tests and coverage.   *
******************************************
Project Name    : app
Project Descp.  : DEPENDENCY MANAGER App -> A Java Project example - assignment #1 for training purposes
Project Version : 1.0
Relative Path   : :app
Absolute Path   : E:\gradle\newdm\newdm\app
Created by      : Adam Trcka
Quick Instructions -> to run the program, type 'gradle run --args <JSON filename>.json'

> Task :app:javadoc
E:\gradle\newdm\newdm\app\src\main\java\com\adamtrcka\work\hw\args\ArgumentValidator.java:3: warning: no comment
public class ArgumentValidator {
       ^
E:\gradle\newdm\newdm\app\src\main\java\com\adamtrcka\work\hw\args\ArgumentValidator.java:8: warning: no comment
    public ArgumentValidator(String[] args) {
           ^
```

4.2. Unit Tests

Basic unit tests are generated when you build the SW package using gradle.
The resulting HTLM docs are stored in: `\app\build\reports\tests\test`

The resulting reports show the basic pass/fail Junit4 unit tests, written in the AppTest testing:



Class com.adamtrcka.work.AppTest

all > com.adamtrcka.work > AppTest

2 tests	0 failures	0 ignored	0.027s duration	100% successful

Test	Duration	Result
testAppHasAVersion	0.013s	passed
testArgValidator	0.014s	passed

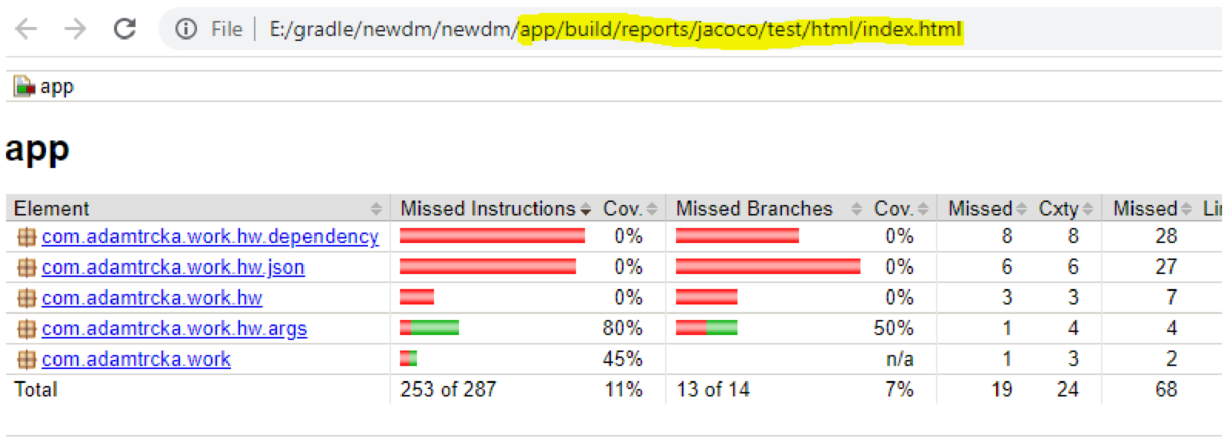
Typical Junit4 unit test report

4.3. Test Coverage

The Jacoco plug-in is triggered with each build to show the complete code coverage. To run units tests, perform the following command line interface

```
> gradle mTR
```
[2]

The resulting HTLM docs are stored in: `app\build\reports\jacoco\test\html`



File | E:/gradle/newdm/newdm/app/build/reports/jacoco/test/html/index.html

app

app

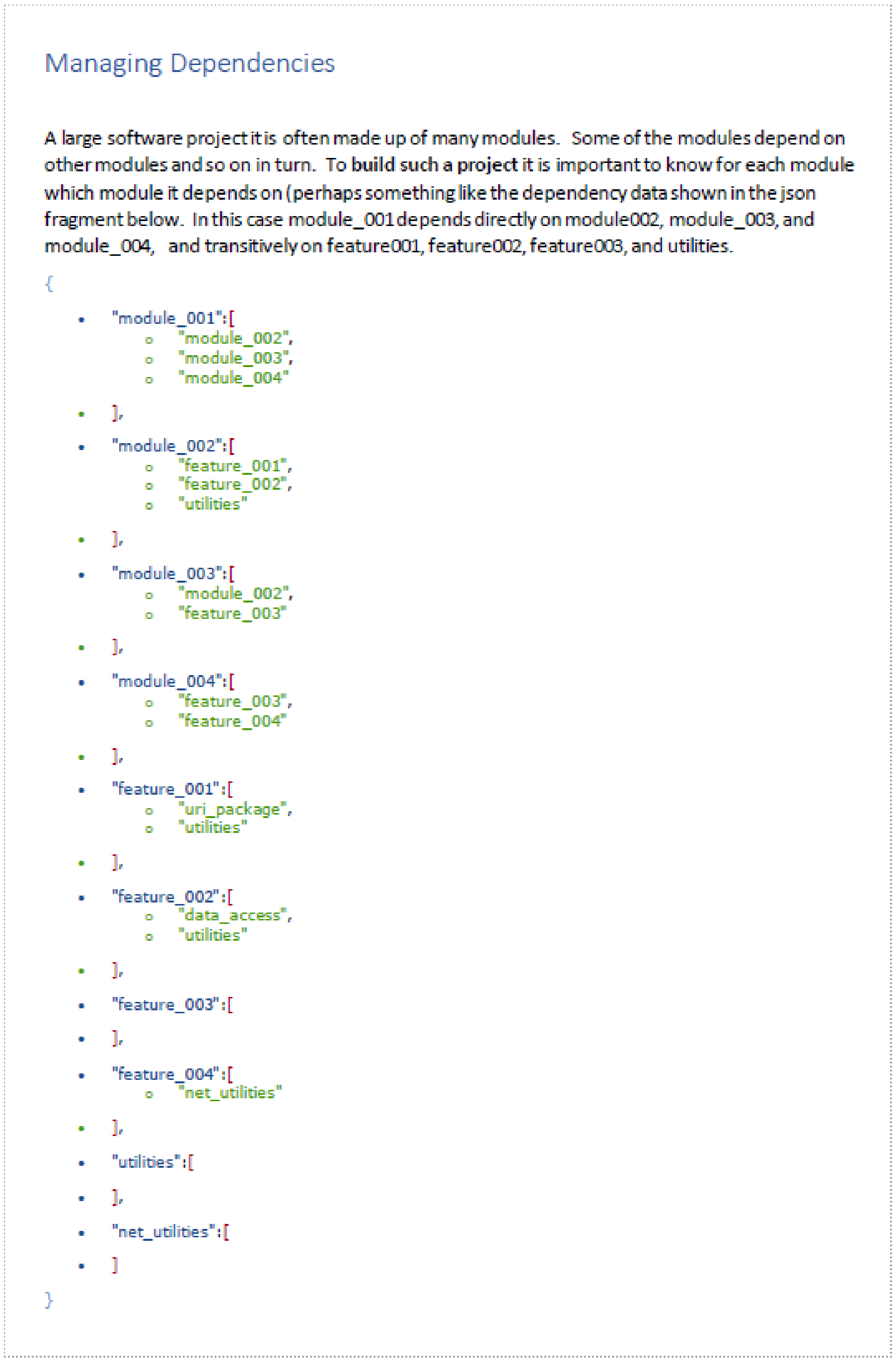
Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Li
com.adamtrcka.work.hw.dependency		0%		0%	8	8	28	
com.adamtrcka.work.hw.json		0%		0%	6	6	27	
com.adamtrcka.work.hw		0%		0%	3	3	7	
com.adamtrcka.work.hw.args		80%		50%	1	4	4	
com.adamtrcka.work		45%		n/a	1	3	2	
Total	253 of 287	11%	13 of 14	7%	19	24	68	

Typical Jacoco coverage report

[2] mTR is a camel case call for the *masterTestReport()* custom task in the *build.gradle* script

ANNEX 1 - Customer inputs

For reference purposes only. These customer inputs are included for traceability reasons.



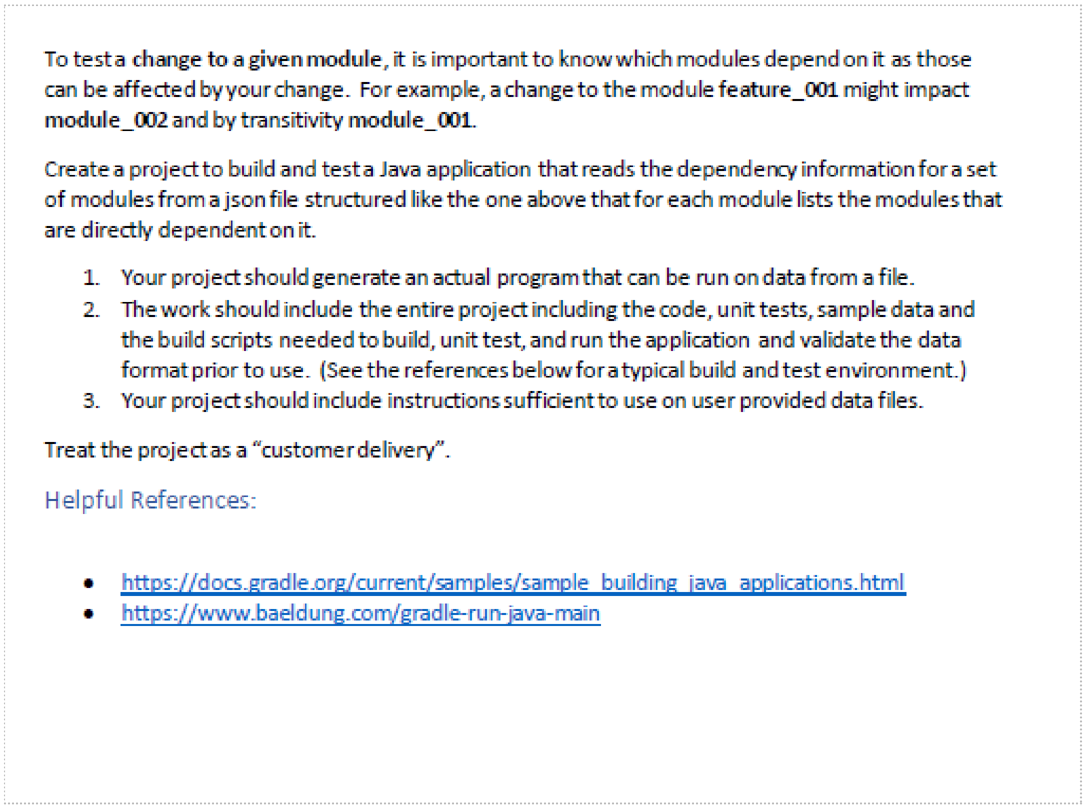
Managing Dependencies

A large software project it is often made up of many modules. Some of the modules depend on other modules and so on in turn. To **build such a project** it is important to know for each module which module it depends on (perhaps something like the dependency data shown in the json fragment below. In this case module_001 depends directly on module002, module_003, and module_004, and transitively on feature001, feature002, feature003, and utilities.

{

- "module_001":[
    - "module_002",
    - "module_003",
    - "module_004"
-],
- "module_002":[
    - "feature_001",
    - "feature_002",
    - "utilities"
-],
- "module_003":[
    - "module_002",
    - "feature_003"
-],
- "module_004":[
    - "feature_003",
    - "feature_004"
-],
- "feature_001":[
    - "uri_package",
    - "utilities"
-],
- "feature_002":[
    - "data_access",
    - "utilities"
-],
- "feature_003":[
-],
- "feature_004":[
    - "net_utilities"
-],
- "utilities":[
-],
- "net_utilities":[
-]

}



To test a **change to a given module**, it is important to know which modules depend on it as those can be affected by your change. For example, a change to the module **feature_001** might impact **module_002** and by transitivity **module_001**.

Create a project to build and test a Java application that reads the dependency information for a set of modules from a json file structured like the one above that for each module lists the modules that are directly dependent on it.

1. Your project should generate an actual program that can be run on data from a file.
2. The work should include the entire project including the code, unit tests, sample data and the build scripts needed to build, unit test, and run the application and validate the data format prior to use. (See the references below for a typical build and test environment.)
3. Your project should include instructions sufficient to use on user provided data files.

Treat the project as a "customer delivery".

Helpful References:

- https://docs.gradle.org/current/samples/sample_building_java_applications.html
- https://www.baeldung.com/gradle-run-java-main

• https://docs.gradle.org/current/samples/sample_building_java_applications.html
• https://www.baeldung.com/gradle-run-java-main

ANNEX 2 - Customer inputs - clarifications

For item 2, forget the dependencies call that gradle as because the point is you have json in a file that just says this

A -> B, C
B -> D
D-> A
E-> F, G, A

Now you want a file outputed that give the DEPENDENTS report

A: [D,E]
B: [A]
C: [A]
D: [B]
E: []
F: [E]
G: [E]

I know the above isn't proper json. So assume you have a file that has a map with a set of node names. If you reverse that map, you have your report.

ANNEX 3 - Useful resources JSON Generator

Internal Note:
https://www.json-generator.com/

```
[
  '{{repeat(4, 7)}}',
  {
    utilizesFunctions: [
      '{{repeat(7)}}',
      '{{lorem(1, "words")}}'
    ]
  }
]
```

Validation Test:

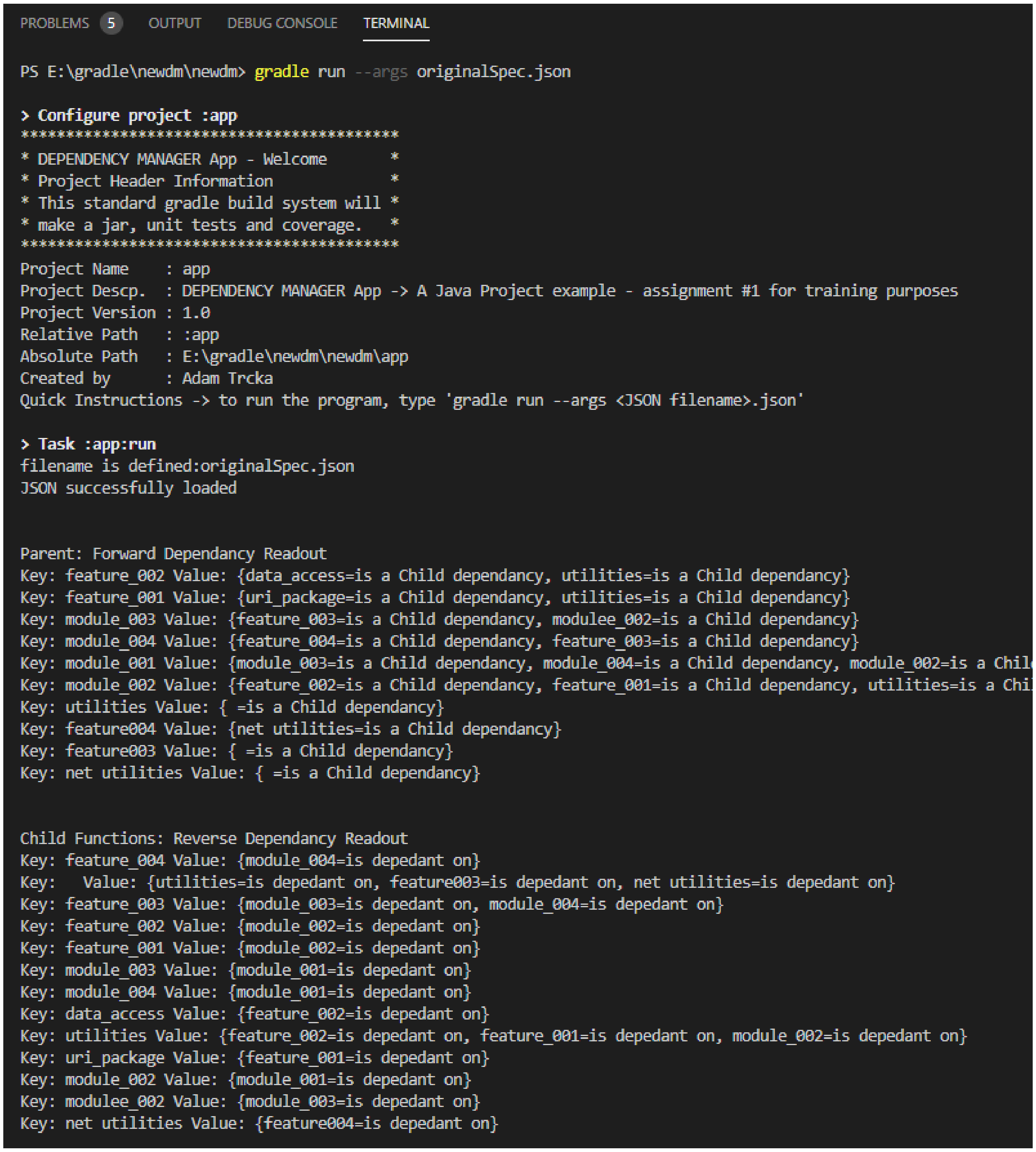
The following tool shall be used to validate JSON sample data:

`https://jsonlint.com/?code=`

Alternative

`https://jsonformatter.org/`

ANNEX 4 - Nominal output with originalSpec.json



```
PS E:\gradle\newdm\newdm> gradle run --args originalSpec.json

> Configure project :app
******************************************
* DEPENDENCY MANAGER App - Welcome       *
* Project Header Information             *
* This standard gradle build system will *
* make a jar, unit tests and coverage.   *
******************************************
Project Name    : app
Project Descp.  : DEPENDENCY MANAGER App -> A Java Project example - assignment #1 for training purposes
Project Version : 1.0
Relative Path   : :app
Absolute Path   : E:\gradle\newdm\newdm\app
Created by      : Adam Trcka
Quick Instructions -> to run the program, type 'gradle run --args <JSON filename>.json'

> Task :app:run
filename is defined:originalSpec.json
JSON successfully loaded


Parent: Forward Dependancy Readout
Key: feature_002 Value: {data_access=is a Child dependancy, utilities=is a Child dependancy}
Key: feature_001 Value: {uri_package=is a Child dependancy, utilities=is a Child dependancy}
Key: module_003 Value: {feature_003=is a Child dependancy, modulee_002=is a Child dependancy}
Key: module_004 Value: {feature_004=is a Child dependancy, feature_003=is a Child dependancy}
Key: module_001 Value: {module_003=is a Child dependancy, module_004=is a Child dependancy, module_002=is a Chil
Key: module_002 Value: {feature_002=is a Child dependancy, feature_001=is a Child dependancy, utilities=is a Chi
Key: utilities Value: { =is a Child dependancy}
Key: feature004 Value: {net utilities=is a Child dependancy}
Key: feature003 Value: { =is a Child dependancy}
Key: net utilities Value: { =is a Child dependancy}


Child Functions: Reverse Dependancy Readout
Key: feature_004 Value: {module_004=is depedant on}
Key:   Value: {utilities=is depedant on, feature003=is depedant on, net utilities=is depedant on}
Key: feature_003 Value: {module_003=is depedant on, module_004=is depedant on}
Key: feature_002 Value: {module_002=is depedant on}
Key: feature_001 Value: {module_002=is depedant on}
Key: module_003 Value: {module_001=is depedant on}
Key: module_004 Value: {module_001=is depedant on}
Key: data_access Value: {feature_002=is depedant on}
Key: utilities Value: {feature_002=is depedant on, feature_001=is depedant on, module_002=is depedant on}
Key: uri_package Value: {feature_001=is depedant on}
Key: module_002 Value: {module_001=is depedant on}
Key: modulee_002 Value: {module_003=is depedant on}
Key: net utilities Value: {feature004=is depedant on}
```