carpark-manager

Course Details:

- DAT4001 - Programming Fundamentals
- Autumn 2023
- Group A

Project Team:

- Ross Grant (st20271524)
- Sam Loftus (st20287201)
- Tom Rowan (st20285213)

Project Tutor:

- Dr Barry Bentley

Project Brief

> "You have been asked to develop a car park management system to track the entry and exit times of vehicles, and allow searches on historic data."

The following is written to provide *our customer* with several options to ensure that they can run the prototype on their system.

Prototype

The supplied version of the project (1.0) is a **prototype**. A sample database is included, providing data for 50 visits per day, from 1-30 November 2023.

Limitations

We only support legitimately formatted UK VRNs of the post 2001 format. For example: OP51JKL, AS10ASD, QW63RTY. We do not support 'old style' or personalised number plates in this version, such as B16LUV, A3ACE, BUB87E, POP33Y

Sample Database

We have provided 'rough-and-ready' utility software to generate additional sample data as required. (Further information in extra/generate-test-database/README.md).

Building and Running this Project

There are multiple way to build this Java project.

Using Docker:

- Automated Build of Local Docker Image, uses Maven.
- Pull Docker Image from Docker Hub.

- Manually Building the Docker Image.

Build and Run:

- Use an IDE (No detail below - IDE dependant).
- Automated Build of .jar file, uses Maven.
- Manual Build from Source.

Using Docker

Automated Build of Local Docker Image (Maven)

The preferred and easiest way to build is to use Maven to drive Docker. This guarantees that the build will result in a consistent version of the project that is independent of any local libraries or file paths.

The Dockerfile provides a temporary 'build' image and a runnable container image without needing to compile or build the project code within an IDE. The build process is carried out automatically by the container.

We have provided an option to automatically build docker images locally. This can be carried out using Apache Maven. (If this is not installed in your environment, it can be obtained from https://maven.apache.org/download.cgi?).

We have provided a simple set of copy/paste-able commands for each section below the main text description and example commands.

```
c:\your_path\carpark-manager> mvn install -f .\prototype\pom.xml
c:\your_path\carpark-manager> docker images
REPOSITORY                              TAG       IMAGE ID       CREATED
SIZE
carparkmanager/generate-test-database   latest    46a7b87d9c33   34 seconds ago
210MB
carparkmanager/prototype                1.0       46a7b87d9c33   34 seconds ago
210MB
```

Then to run this image:

```
c:\your_path\carpark-manager> docker run -ti --name cpm-prototype
carparkmanager/prototype:1.0
[...Carpark Manager app opens here...]
```

Commands for simplified Copy/Paste (run from c:\your_path\carpark-manager>) :

```
mvn install -f .\prototype\pom.xml
docker run -ti --name cpm-prototype carparkmanager/prototype:1.0
```

Restarting a Stopped Container

The container created above is not deleted when you exit the Carpark Manager application, and changes to the database are stored. To re-enter the container, and continue, please use:

```
c:\your_path\carpark-manager> docker start -a cpm-prototype
```

If you do want to use a disposable container, add --rm to the docker run command above. This will destroy any database changes made during the run of the container. The image has the same 'starter' database, as will any containers run from the image.

Commands for Copy/Paste (from c:\your_path\carpark-manager>) :

```
docker start -a cpm-prototype
```

Rebuilding the Image

After the first run, to remove and rebuild the images from source, you can use the following. Note that a new test database will be created and installed only when you create a new image.:

```
c:\your_path\carpark-manager> mvn clean install -f .\prototype\pom.xml
```

Commands for Copy/Paste (from c:\your_path\carpark-manager>) :

```
mvn clean install -f .\prototype\pom.xml
```

Pull Docker Image from Docker Hub

We have provided an image on the Docker Hub repository should there be any issues with creating a local Docker image. This can be pulled from the repository and run in a single command:

```
c:\your_path\carpark-manager> docker run -ti --name cpm-prototype
daemonchild/carparkmanager-prototype:1.0
```

Commands for Copy/Paste (from c:\your_path\carpark-manager>) :

```
docker run -ti --name cpm-prototype daemonchild/carparkmanager-prototype:1.0
```

(Please note that the image *name* is different from used in the previous commands; the image was created from the latest source code prior to submission and cannot be uploaded to Docker Hub with name used

elsewhere in this document.)

Manually Building the Docker Image

If you don't have maven, you will need to compile the sources manually (see below). You can then create the Docker image manually by using:

```
c:\your_path\carpark-manager> cd prototype
c:\your_path\carpark-manager> docker build -t mycarparkmanager:1.0 .
```

To run this manually built image, use:

```
c:\your_path\carpark-manager> docker run -ti --name cpm-prototype
mycarparkmanager:1.0
```

Commands for Copy/Paste (from c:\your_path\carpark-manager>) :

```
cd prototype
docker build -t mycarparkmanager:1.0 .
docker run -ti --name cpm-prototype daemonchild/carparkmanager-prototype:1.0
```

Build and Run

Automated Build from Source (Maven)

An automated local build can be carried out using Apache Maven. If not installed in your environment, it can be obtained from https://maven.apache.org/download.cgi?.

The following can be used to build and run the application:

```
c:\your_path\carpark-manager> mvn package -f .\prototype\pom.xml
c:\your_path\carpark-manager> java -jar .\prototype\target\prototype-1.0.jar
```

Commands for Copy/Paste (from c:\your_path\carpark-manager>) :

```
mvn package -f .\prototype\pom.xml
java -jar .\prototype\target\prototype-1.0.jar
```

Manual Build from Source

If all other tools are unavailable, it is possible to build the project as follows:

```
c:\your_path\carpark-manager> cd prototype\src\main\java\
c:\your_path\carpark-manager\prototype\src\main\java\> javac -d
..\..\..\target\classes\ carparkmanager\*.java
c:\your_path\carpark-manager\prototype\src\main\java\> cd ..\..\..\target\classes\
c:\your_path\carpark-manager\prototype\src\main\java\> cp
..\..\src\main\resources\* .
c:\your_path\carpark-manager\carpark-manager\prototype\target\classes>jar cvmf
..\..\MANIFEST.MF ..\prototype-1.0.jar  .\carparkmanager\*.class
```

Commands for Copy/Paste (from c:\your_path\carpark-manager>) :

```
cd prototype\src\main\java\
javac -d ..\..\..\target\classes\ carparkmanager\*.java
cd ..\..\..\target\classes\
cp ..\..\src\main\resources\* .
jar cvmf ..\..\MANIFEST.MF ..\prototype-1.0.jar  .\carparkmanager\*.class
```

You will then need to ensure that there is a file called "database.csv" in the prototype/database-files/ folder. There is a file supplied named "sample-database.csv" in the folder. The simplest option is to rename that file and ensure that you run the application from the root folder of the project.

Application Development File System

For reference, the file system for the app should be as follows:

Running App in development environment on Windows

```
\carpark-manager                                              <-- project root
\carpark-manager\prototype\
\carpark-manager\prototype\src\main\resources\                <-- config files
\carpark-manager\prototype\src\main\java\carparkmanager\
\carpark-manager\prototype\target\                            <-- .jar goes here
\carpark-manager\prototype\target\classes\
\carpark-manager\prototype\database-files\
\carpark-manager\prototype\extra\generate-database\           <-- python based
tool
```

- App searches for database in ($PWD)\prototype\database-files\
- $PWD/Working Directory is [your-path]\carpark-manager
- The .jar and the database folder is all the running app needs.

Running App in Linux Docker Image

```
/app/                                                    <-- .jar is here
/app/database-files/
```

- App searches for database in ($PWD)/database-files/
- $PWD/Working Directory is /app
- The .jar and the database folder is all the running app needs.

Note - it is possible to use a bind mount to provide an external permanent database to the application if required. This would survive a container rebuild. (https://docs.docker.com/storage/bind-mounts/)

(End of Document)