Q1) Pull any image from the docker hub, create its container, and execute it showing the output.

Image: An Image is a read only template with instructions for creating a container. Image is described in the text file called Docker File.

Docker hub: Docker Hub is a Docker Registry, a cloud-hosted version, open-source, scalable server-side application, and stateless. It can manage the sharing and storage of docker images.

Container: Container is like creating a running instance for the particular image. The basic Purpose of docker is to run the containers using docker Engine.

How to Run a Container.?

Running of containers can be managed by the command called Docker run command.

Syntax : Docker run <Image Name>.

Pulling an image from the docker hub:

We use docker pull command to pull an image from the docker hub.

Syntax: docker pull <image name>

```
C:\Users\ravin>docker pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world
Digest: sha256:6e8b6f026e0b9c419ea0fd02d3905dd0952ad1feea67543f525c73a0a790fefb
Status: Image is up to date for hello-world:latest
docker.io/library/hello-world:latest
```

Here the image i.e hello world is already exits in the docker hub.

```
C:\Users\ravin>docker images
REPOSITORY    TAG       IMAGE ID       CREATED         SIZE
java-app      latest    ed27da69ccce   34 hours ago    526MB
<none>        <none>    6e1076b1fd55   34 hours ago    526MB
<none>        <none>    f3d775d217ae   34 hours ago    526MB
ubuntu        latest    58db3edaf2be   3 weeks ago     77.8MB
resin/docs    latest    592de848a9b7   4 months ago    1.1GB
hello-world   latest    feb5d9fea6a5   17 months ago   13.3kB
```

Here image hello world is present in the docker hub.

Creating a container:

To create a container we use the below command

Syntax: docker create -it associated the image.

```
C:\Users\ravin>docker create -it centos
Unable to find image 'centos:latest' locally
latest: Pulling from library/centos
a1d0c7532777: Pull complete
Digest: sha256:a27fd8080b517143cbbbab9dfb7c8571c40d67d534bbdee55bd6c473f432b177
Status: Downloaded newer image for centos:latest
6d4b3481889ca38565588bec08fa7cde47c59432ce9cac21a8ef822045b8f89e

C:\Users\ravin>
```

Here I have created a container for the centos image but the centos image is not available in the ready state.so it is giving me like **"Unable to find the image locally".** It means locally image is not in the local machine, So now docker daemon will pulls the centos image from the docker hub and creates container for that centos image.

```
C:\Users\ravin>docker images
REPOSITORY    TAG       IMAGE ID       CREATED         SIZE
java-app      latest    ed27da69ccce   34 hours ago    526MB
<none>        <none>    6e1076b1fd55   34 hours ago    526MB
<none>        <none>    f3d775d217ae   35 hours ago    526MB
ubuntu        latest    58db3edaf2be   3 weeks ago     77.8MB
resin/docs    latest    592de848a9b7   4 months ago    1.1GB
hello-world   latest    feb5d9fea6a5   17 months ago   13.3kB
centos        latest    5d0da3dc9764   17 months ago   231MB
```

Here centos image is not created by using the docker build command although docker daemon pulls the image from the docker hub's registry.

Display all the containers with their respective container ID's:

To Display all the containers we use the docker ps –all command

Syntax : docker ps –all

```
C:\Users\ravin\dockerfiles>docker ps --all
CONTAINER ID   IMAGE          COMMAND         CREATED          STATUS                      PORTS     NAMES
4c1fc5c99425   centos         "/bin/bash"     9 minutes ago    Exited (0) 9 minutes ago              strange_chatelet
6d4b3481889c   centos         "/bin/bash"     19 minutes ago   Created                               sharp_saha
567bfdf99727   ubuntu         "/bin/bash"     24 hours ago     Exited (255) 20 hours ago             exciting_torvalds
7cb6d75b93b1   java-app       "java Method"   34 hours ago     Exited (255) 27 hours ago             upbeat_hofstadter
c74c367476ec   6e1076b1fd55   "java Method"   35 hours ago     Exited (0) 35 hours ago               zen_raman
c5a6d40ead6c   6e1076b1fd55   "java Method"   35 hours ago     Exited (0) 35 hours ago               busy_wilson
a060c709b5bf   f3d775d217ae   "java Sample"   35 hours ago     Exited (0) 35 hours ago               gracious_banzai
5503fb4685f3   f3d775d217ae   "."             35 hours ago     Created                               musing_dubinsky
d5d455bc10b3   f3d775d217ae   "."             35 hours ago     Created                               compassionate_shaw
87964001aaeb   hello-world    "/hello"        2 days ago       Exited (0) 2 days ago                 vigorous_edison
6abb22b81023   hello-world    "/hello"        2 days ago       Exited (0) 2 days ago                 competent_saha
61e6c5478a44   hello-world    "/hello"        3 days ago       Exited (0) 3 days ago                 determined_satoshi
55022b0de522   hello-world    "/hello"        3 days ago       Exited (0) 3 days ago                 focused_ishizaka
b972788a7386   hello-world    "/hello"        4 days ago       Exited (0) 4 days ago                 adoring_kepler

C:\Users\ravin\dockerfiles>S
```

Following the containers with their container ID's.

Q2) Create the basic java application, generate its image with necessary files, and execute it with docker.

Creating the basic Java Application.

Step 1 : Create a Directory to store java File and Docker File.To Create a Directory we use the below command.

Syntax : mkdir <directory name>

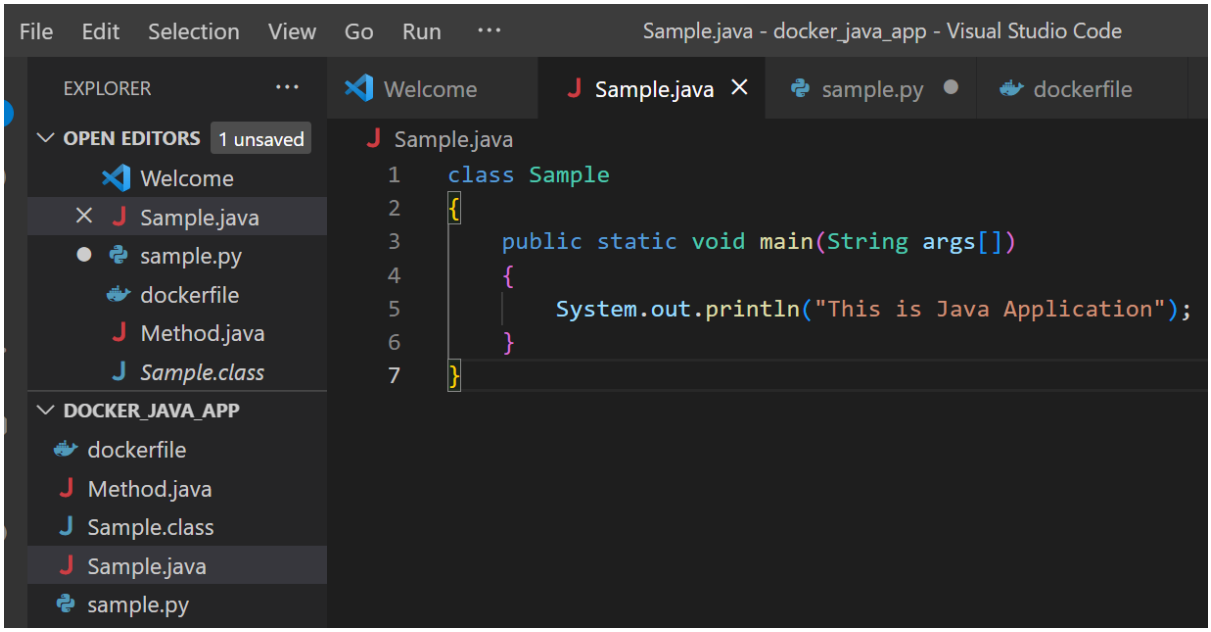
```
C:\Users\ravin>mkdir docker_java_app
```

Step2: swift to the directory called docker_java_app

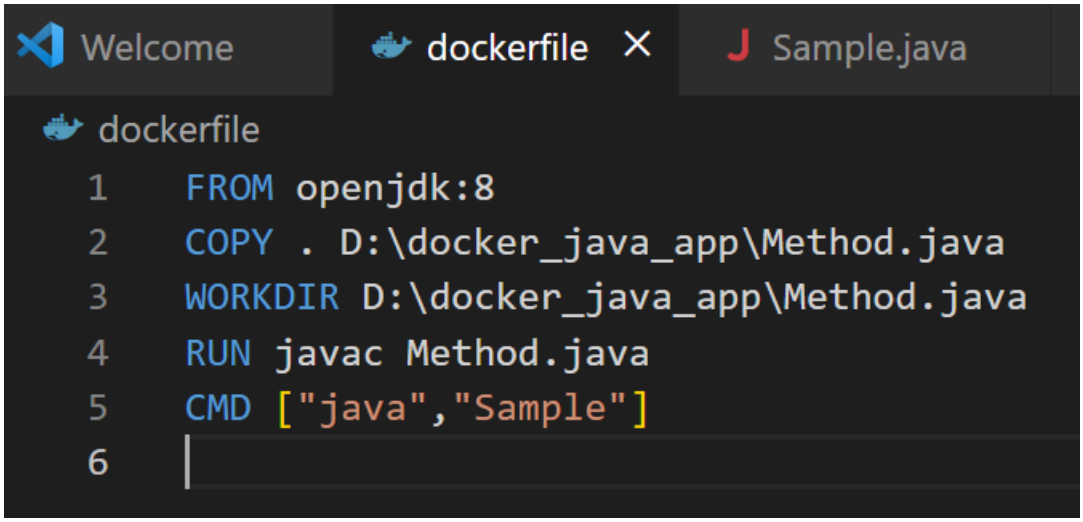
```
D:\>cd docker_java_app

D:\docker_java_app>code .
```

Step3: Create a java file,save it as Sample.java



Step4: Create a docker file.



Step 5: Now create an Image by following below command. We must login as root in order to create a image. In the following command, java-app is name of the image .We can have any name for our docker image.

```
d:\docker_java_app>docker build -t java-app .
[+] Building 502.9s (10/10) FINISHED
 => [internal] load build definition from Dockerfile                                                    0.1s
 => => transferring dockerfile: 179B                                                                    0.0s
 => [internal] load .dockerignore                                                                       0.0s
 => => transferring context: 2B                                                                         0.0s
 => [internal] load metadata for docker.io/library/openjdk:17.0.2                                       4.4s
 => [auth] library/openjdk:pull token for registry-1.docker.io                                          0.0s
 => [internal] load build context                                                                       0.0s
 => => transferring context: 486B                                                                       0.0s
 => [1/4] FROM docker.io/library/openjdk:17.0.2@sha256:528707081fdb9562eb819128a9f85ae7fe000e2fbaeaf9f87662e7b3  497.0s
 => => resolve docker.io/library/openjdk:17.0.2@sha256:528707081fdb9562eb819128a9f85ae7fe000e2fbaeaf9f87662e7b3f3  0.0s
 => => sha256:5e28ba2b4cdb3a7c3bd0ee2e635a5f6481682b77eabf8b51a17ea8bfe1c05697 4.45kB / 4.45kB           0.0s
 => => sha256:38a980f2cc8accf69c23deae6743d42a87eb34a54f02396f3fcfd7c2d06e2c5b 42.11MB / 42.11MB      122.9s
 => => sha256:de849f1cfbe60b1c06a1db83a3129ab0ea397c4852b98e3e4300b12ee57ba111 13.53MB / 13.53MB       57.2s
 => => sha256:a7203ca35e75e068651c9907d659adc721dba823441b78639fde66fc988f042f 187.53MB / 187.53MB    493.1s
 => => sha256:528707081fdb9562eb819128a9f85ae7fe000e2fbaeaf9f87662e7b3f38cb7d8 1.04kB / 1.04kB         0.0s
 => => sha256:98f0304b3a3b7c12ce641177a99d1f3be56f532473a528fda38d53d519cafb13 954B / 954B             0.0s
 => => extracting sha256:38a980f2cc8accf69c23deae6743d42a87eb34a54f02396f3fcfd7c2d06e2c5b               1.5s
 => => extracting sha256:de849f1cfbe60b1c06a1db83a3129ab0ea397c4852b98e3e4300b12ee57ba111               0.5s
 => => extracting sha256:a7203ca35e75e068651c9907d659adc721dba823441b78639fde66fc988f042f               3.6s
 => [2/4] COPY . D:docker_java_appSample.java                                                           0.4s
 => [3/4] WORKDIR D:docker_java_appSample.java                                                          0.0s
 => [4/4] RUN java Sample                                                                               0.6s
 => exporting to image                                                                                  0.2s
 => => exporting layers                                                                                 0.2s
 => => writing image sha256:cc9f496e459efeb0efaca4c30f21a30f45f06798ee4a865ac7ab258c472c66be            0.0s
 => => naming to docker.io/library/java-app                                                             0.0s
```

Step 6: After successfully building the image, now we can run docker by using run command.

```
d:\docker_java_app>docker run java-app
This is Java Application
```