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

docker pull is used to download the image from the docker daemon.

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.2604]
(c) Microsoft Corporation. All rights reserved.

D:\Annapurna>docker pull centos
Using default tag: latest
latest: Pulling from library/centos
a1d0c7532777: Pull complete
Digest: sha256:a27fd8080b517143cbbbab9dfb7c8571c40d67d534bbdee55bd6c473f432b177
Status: Downloaded newer image for centos:latest
docker.io/library/centos:latest

D:\Annapurna>
```

docker run command creates running containers and executes it.

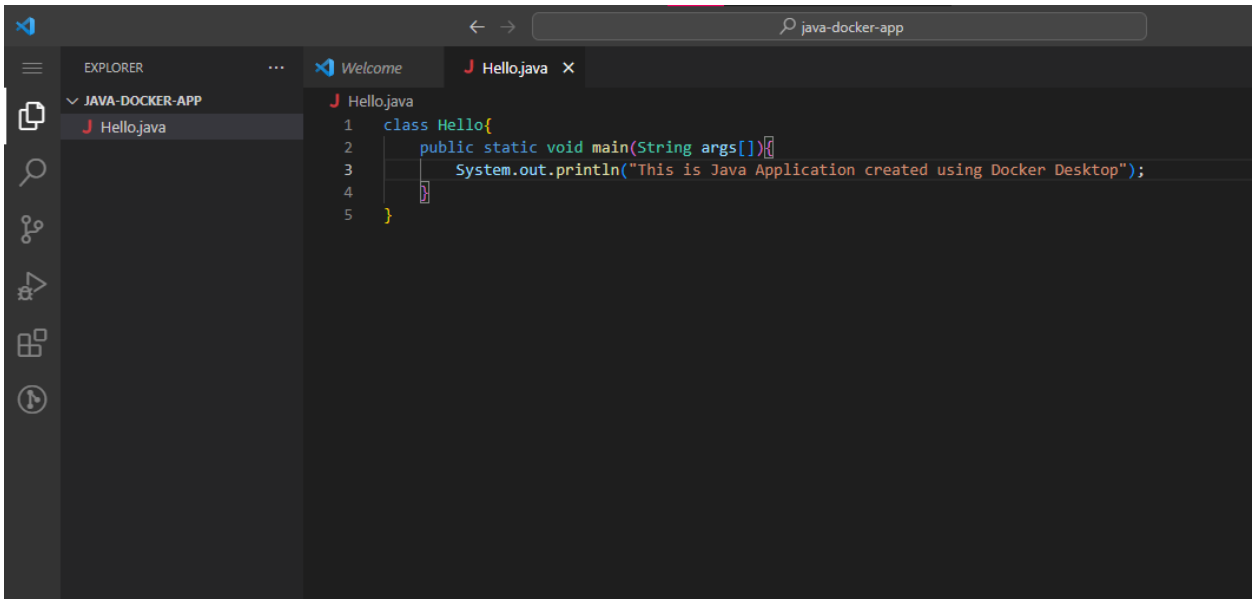
```
@bc4cd9b4747d:/
D:\Annapurna>docker pull centos
Using default tag: latest
latest: Pulling from library/centos
Digest: sha256:a27fd8080b517143cbbbab9dfb7c8571c40d67d534bbdee55bd6c473f432b177
Status: Image is up to date for centos:latest
docker.io/library/centos:latest

D:\Annapurna>docker run -it centos
[root@bc4cd9b4747d /]# ls
bin  dev  etc  home  lib  lib64  lost+found  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
[root@bc4cd9b4747d /]# cat bin
cat: bin: Is a directory
[root@bc4cd9b4747d /]# vi demo
[root@bc4cd9b4747d /]# cat demo
centos was pulled and runned using docker desktop
[root@bc4cd9b4747d /]#
```

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

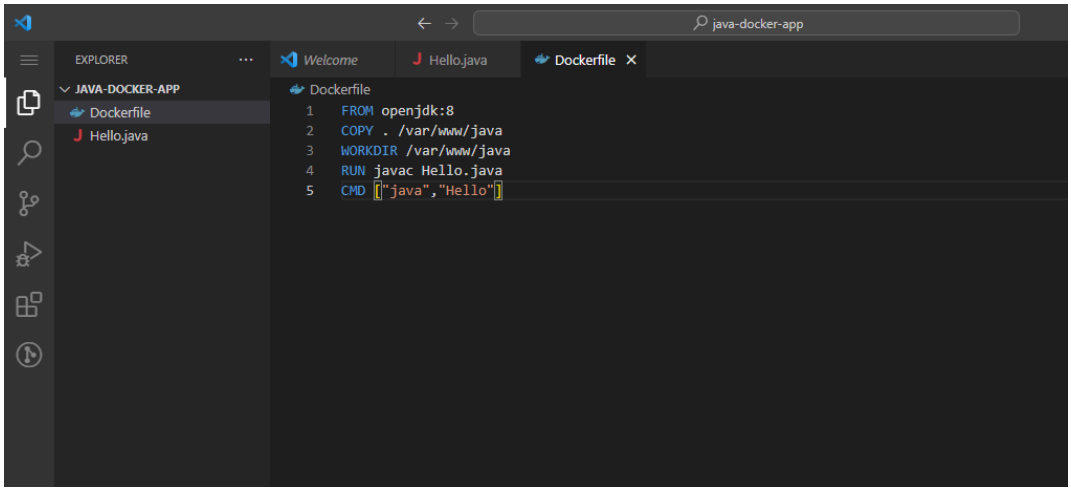
First I have created a directory using mkdir command.

```
D:\Annapurna>mkdir java-docker-app
```

I have created a Hello.java file in the java-docker-app folder and saved it.



Then I have created another file named Dockerfile and added the below content in it.



Now I have changed the directory to java-docker-app.

```
D:\Annapurna>cd java-docker-app

D:\Annapurna\java-docker-app>
```

I have used the build command to create a java-app image

docker build –t java-app .

```
C:\Windows\System32\cmd.exe

D:\Annapurna\java-docker-app>docker build -t java-app .
[+] Building 295.6s (10/10) FINISHED
 => [internal] load build definition from Dockerfile                                                                    6.4s
 => => transferring dockerfile: 140B                                                                                    0.5s
 => [internal] load .dockerignore                                                                                       6.2s
 => => transferring context: 2B                                                                                         0.6s
 => [internal] load metadata for docker.io/library/openjdk:8                                                           13.2s
 => [auth] library/openjdk:pull token for registry-1.docker.io                                                          0.0s
 => [internal] load build context                                                                                       1.9s
 => => transferring context: 334B                                                                                       0.1s
 => [1/4] FROM docker.io/library/openjdk:8@sha256:86e863cc57215cfb181bd319736d0baf625fe8f150577f9eb58bd937f5452cb8     252.4s
 => => resolve docker.io/library/openjdk:8@sha256:86e863cc57215cfb181bd319736d0baf625fe8f150577f9eb58bd937f5452cb8       1.5s
 => => sha256:86e863cc57215cfb181bd319736d0baf625fe8f150577f9eb58bd937f5452cb8 1.04kB / 1.04kB                          0.0s
 => => sha256:3af2ac94130765b73fc8f1b42ffc04f77996ed8210c297fcfa28ca880ff0a217 1.79kB / 1.79kB                          0.0s
 => => sha256:b273004037cc3af245d8e08cfbfa672b93ee7dcb289736c82d0b58936fb71702 7.81kB / 7.81kB                          0.0s
 => => sha256:2068746827ec1b043b571e4788693eab7e9b2a95301176512791f8c317a2816a 10.88MB / 10.88MB                        7.9s
 => => sha256:d9d4b9b6e964657da49910b495173d6c4f0d9bc47b3b44273cf82fd32723d165 5.16MB / 5.16MB                          7.1s
 => => sha256:001c52e26ad57e3b25b439ee0052f6692e5c0f2d5d982a00a8819ace5e521452 55.00MB / 55.00MB                       47.7s
 => => sha256:9daef329d35093868ef75ac8b7c6eb407fa53abbcb3a264c218c2ec7bca716e6 54.58MB / 54.58MB                       52.0s
 => => sha256:d85151f15b6683b98f21c3827ac545188b1849efb14a1049710ebc4692de3dd5 5.42MB / 5.42MB                        15.1s
 => => sha256:52a8c426d30b691c4f7e8c4b438901ddeb82ff80d4540d5bbd49986376d85cc9 210B / 210B                            17.4s
 => => sha256:8754a66e005039a091c5ad0319f055be393c7123717b1f6fee8647c338ff3ceb 105.92MB / 105.92MB                     73.0s
 => => extracting sha256:001c52e26ad57e3b25b439ee0052f6692e5c0f2d5d982a00a8819ace5e521452                              60.9s
 => => extracting sha256:d9d4b9b6e964657da49910b495173d6c4f0d9bc47b3b44273cf82fd32723d165                               4.1s
 => => extracting sha256:2068746827ec1b043b571e4788693eab7e9b2a95301176512791f8c317a2816a                               5.4s
 => => extracting sha256:9daef329d35093868ef75ac8b7c6eb407fa53abbcb3a264c218c2ec7bca716e6                              46.4s
 => => extracting sha256:d85151f15b6683b98f21c3827ac545188b1849efb14a1049710ebc4692de3dd5                               5.0s
 => => extracting sha256:52a8c426d30b691c4f7e8c4b438901ddeb82ff80d4540d5bbd49986376d85cc9                               0.0s
 => => extracting sha256:8754a66e005039a091c5ad0319f055be393c7123717b1f6fee8647c338ff3ceb                              61.3s
 => [2/4] COPY . /var/www/java                                                                                          3.2s
 => [3/4] WORKDIR /var/www/java                                                                                         2.8s
 => [4/4] RUN javac Hello.java                                                                                          9.5s
 => exporting to image                                                                                                  3.2s
 => => exporting layers                                                                                                 2.7s
 => => writing image sha256:e2345dc5bc35de47e7ebf9886e160e8fd528afffde60fbb7a38a25ab2497c8e2                            0.1s
 => => naming to docker.io/library/java-app                                                                             0.1s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
```

To run the java-app application used the following command

docker run java-app

```
C:\Windows\System32\cmd.exe
 => => naming to docker.io/library/java-app

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them

D:\Annapurna\java-docker-app>docker run java-app
This is Java Application created using Docker Desktop

D:\Annapurna\java-docker-app>
```