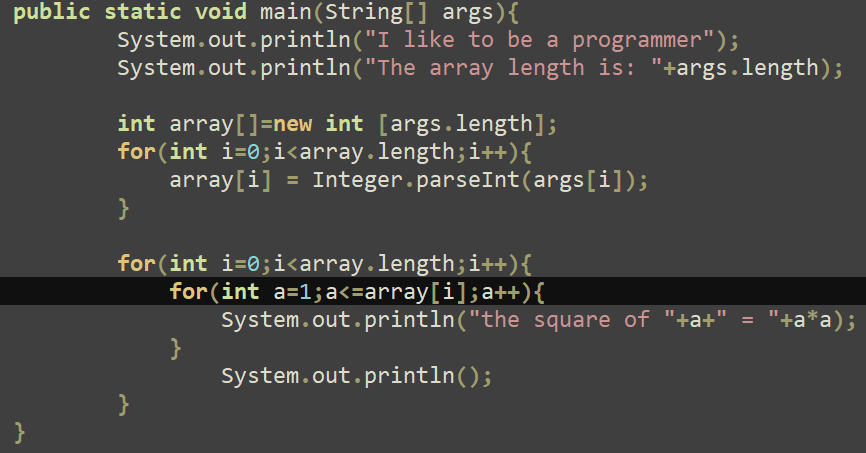
So assume we create an image and part of that image we want to be a java program.

We can go in a different ways to achieve this, but first of all we need to understand the desired behavior of the container. It could be:

1) We run the container and then being in interactive mode with command line we call the java programm and pass the arguments in it when we initiate the run proces, after the programm is finished we are still in interactive mode with command line (cli), and if we want we call program again using different set of values.

2) We run the image and pass the arguments in exactly the same moment as we execute the docker run command, docker runs the container and executes the java program with arguments that was given to him when we initiated docker images. So we pass the arguments for the java program through the run command and set the variable in a Dockerfile, so when the docker runs he gets the values (from run command) and passes them in a variable that was inserted in a dockerfile. In that scenario we need to set the desired behavior in a Dockerfile and provide appropriate arguments when we run docker image.

The java program will look like this:

It accepts an array

Dockerfile:

1)-FROM alpine

2)-RUN apk update && apk add nano && apk add openjdk17

3)-ADD Hello.java Hello.java

4)-ENTRYPOINT javac Hello.java && java Hello $**VARIABLE**

$VARIABLE ->means that docker will expect some values (arguments) in that place to pass to the Java Hello program from the docker run command

For example:

**docker run --rm -e VARIABLE="4 9" passing\_arguments\_to\_javaprogram\_using\_dickerfile**

--rm means that the container will be removed after it's stopped.

-e VARIABLE=”4 9” -> we assign values for java program using -e key so docker will know what it is and what to do with it.

passing\_arguments\_to\_javaprogram\_using\_dickerfile ->it is just the name of the image being run.