

# Контейнеризация (семинары)

## Урок 4. Dockerfile и слои

### Задание:

Необходимо создать Dockerfile, основанный на любом образе (вы в праве выбрать самостоятельно).

В него необходимо поместить приложение, написанное на любом известном вам языке программирования (Python, Java, C, C#, C++).

При запуске контейнера должно запускаться самостоятельно написанное приложение.

### Работа:

Создание папки для работы:

```
cd ~
```

```
mkdir MyContainer
```

```
cd MyContainer
```

В качестве запускаемой программы выбрана простая программа на Java для вывода всех простых чисел до 1000.

Создание файла программы и его заполнение:

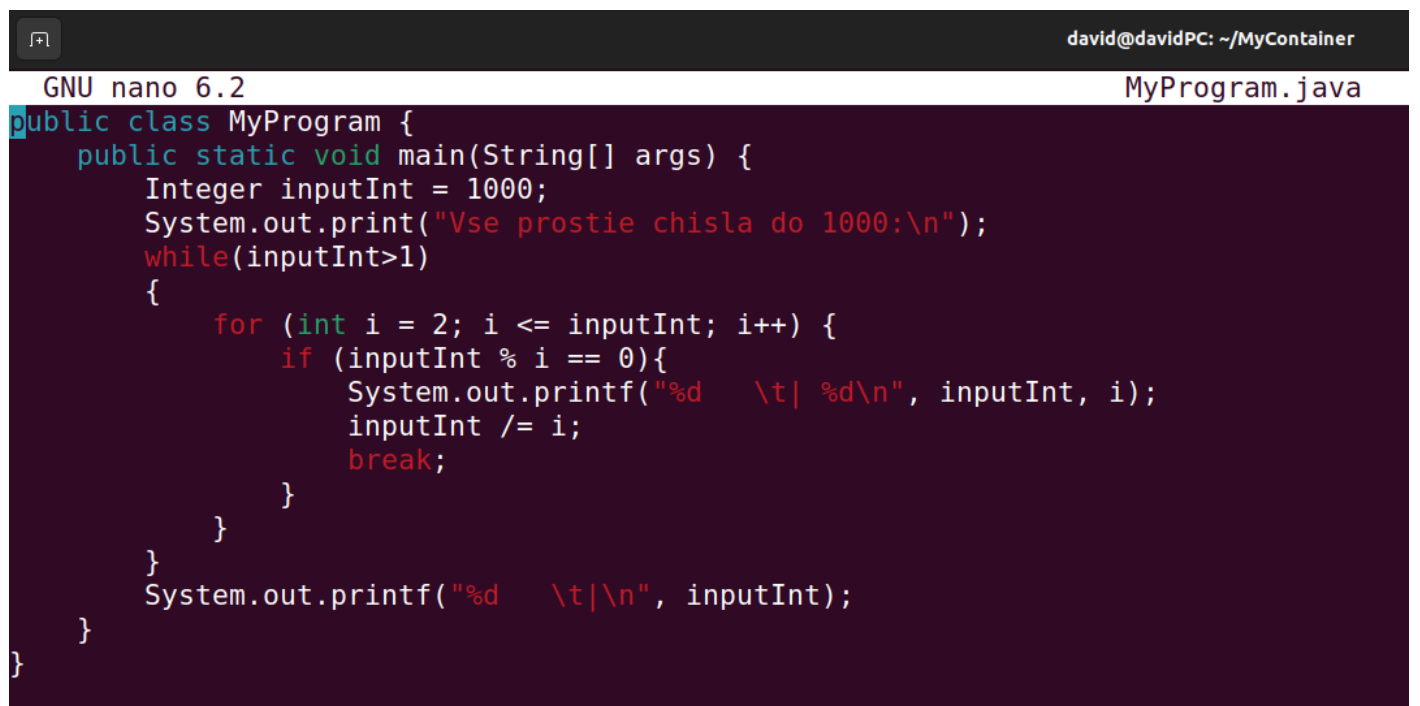
```
touch MyProgram.java
```

```
nano MyProgram.java
```

// ---ниже код на Java---

```
public class MyProgram {
    public static void main(String[] args) {
        Integer inputInt = 1000;
        System.out.print("Vse prostie chisla do 1000:\n");
        while(inputInt>1)
        {
            for (int i = 2; i <= inputInt; i++) {
                if (inputInt % i == 0){
                    System.out.printf("%d \t| %d\n", inputInt, i);
                    inputInt /= i;
                    break;
                }
            }
        }
        System.out.printf("%d \t|\n", inputInt);
    }
}
```

*Скриншот описанных выше действий*



```
GNU nano 6.2 MyProgram.java
public class MyProgram {
    public static void main(String[] args) {
        Integer inputInt = 1000;
        System.out.print("Vse prostie chisla do 1000:\n");
        while(inputInt>1)
        {
            for (int i = 2; i <= inputInt; i++) {
                if (inputInt % i == 0){
                    System.out.printf("%d \t| %d\n", inputInt, i);
                    inputInt /= i;
                    break;
                }
            }
        }
        System.out.printf("%d \t|\n", inputInt);
    }
}
```

Создание Docker файла рядом:

**touch Docker**

**nano Docker**

Код Docker файла:

```
# Используем образ Debian как базовый образ
FROM debian:latest

# Обновляем список пакетов и устанавливаем OpenJDK (default-jdk) и
очищаем кэш
RUN apt-get update -y && apt-get install default-jdk -y && apt-get
clean

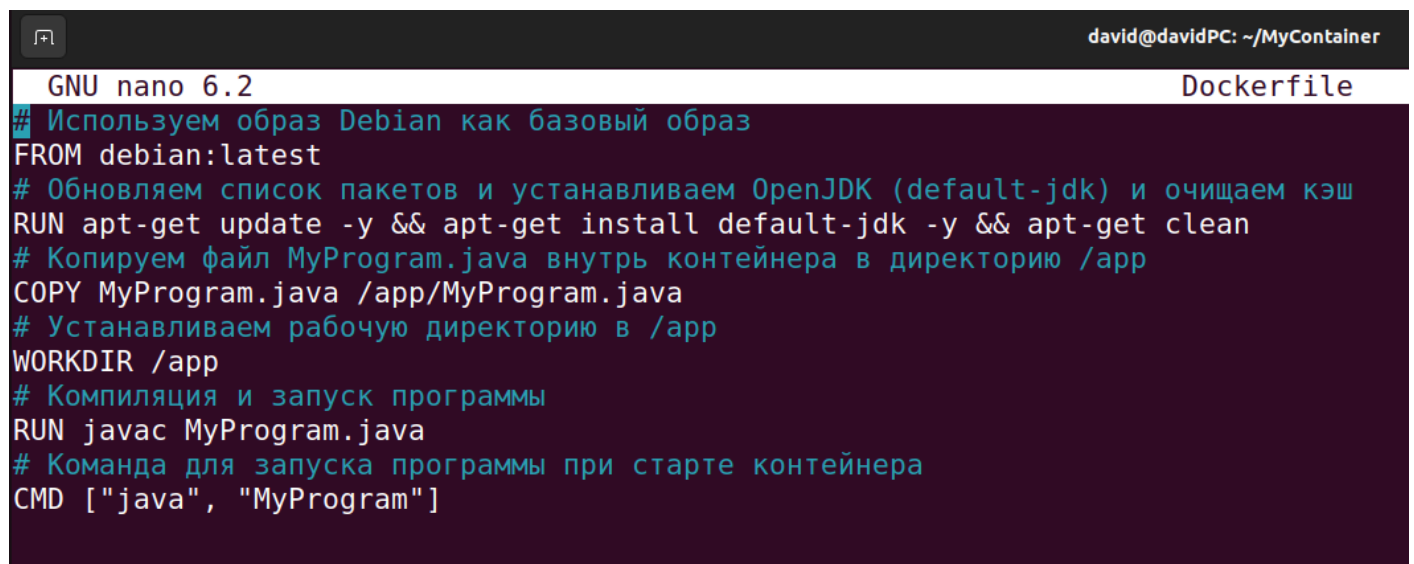
# Копируем файл MyProgram.java внутрь контейнера в директорию /app
COPY MyProgram.java /app/MyProgram.java

# Устанавливаем рабочую директорию в /app
WORKDIR /app

# Компиляция и запуск программы
RUN javac MyProgram.java

# Команда для запуска программы при старте контейнера
CMD ["java", "MyProgram"]
```

*Скриншот описанных выше действий*



```
david@davidPC: ~/MyContainer
GNU nano 6.2 Dockerfile
# Используем образ Debian как базовый образ
FROM debian:latest
# Обновляем список пакетов и устанавливаем OpenJDK (default-jdk) и очищаем кэш
RUN apt-get update -y && apt-get install default-jdk -y && apt-get clean
# Копируем файл MyProgram.java внутрь контейнера в директорию /app
COPY MyProgram.java /app/MyProgram.java
# Устанавливаем рабочую директорию в /app
WORKDIR /app
# Компиляция и запуск программы
RUN javac MyProgram.java
# Команда для запуска программы при старте контейнера
CMD ["java", "MyProgram"]
```

Сборка:

```
sudo docker build -t my-java-app .
```

Скриншот описанных выше действий

```
david@davidPC:~/MyContainer$ sudo docker build -t my-java-app .
[+] Building 19.9s (10/10) FINISHED
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 771B
=> [internal] load metadata for docker.io/library/debian:latest
=> [1/5] FROM docker.io/library/debian:latest@sha256:b4042f895d5d1f8df415caebe7c416f9dbcf0dc8867abb225955006de50b21f3
=> [internal] load build context
=> => transferring context: 893B
=> CACHED [2/5] RUN apt-get update -y && apt-get install default-jdk -y && apt-get clean
=> [3/5] COPY MyProgram.java /app/MyProgram.java
=> [4/5] WORKDIR /app
=> [5/5] RUN javac MyProgram.java
=> exporting to image
=> => exporting layers
=> => writing image sha256:1calee01aff99257f286efd1c8c6b80775b71ff66b9997281f3d5870ddf2f1b1
=> => naming to docker.io/library/my-java-app
```

```
david@davidPC:~/MyContainer$ sudo docker ps -a
[sudo] пароль для david:
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS
9a8b86de85d4   my-java-app    "java MyProgram"        About an hour ago   Exited (0) 59 minutes ago
my-container
c/a109e9c015   phpmyadmin     "/docker-entrypoint..." 25 hours ago     Up 25 hours   0.0.0.0:8080->80/tcp, :::8080->80/tcp
my_admin
c44e48e3a905   mysql:8.1.0    "docker-entrypoint.s..." 47 hours ago     Up 47 hours   3306/tcp, 33060/tcp
my_db
```

Запуск:

```
sudo docker run --name my-container my-java-app
```

Скриншот описанных выше действий

```
david@davidPC:~/MyContainer$ sudo docker run --name my-container my-java-app
Vse prostie chisla do 1000:
1000 | 2
500  | 2
250  | 2
125  | 5
25   | 5
5    | 5
1    |
```

В результате запустился контейнер и запустилась программа для вывода всех простых чисел до 1000.