Academia Tehnică Militară "Ferdinand I“
Facultatea de Sisteme Informatice şi Securitate Cibernetică

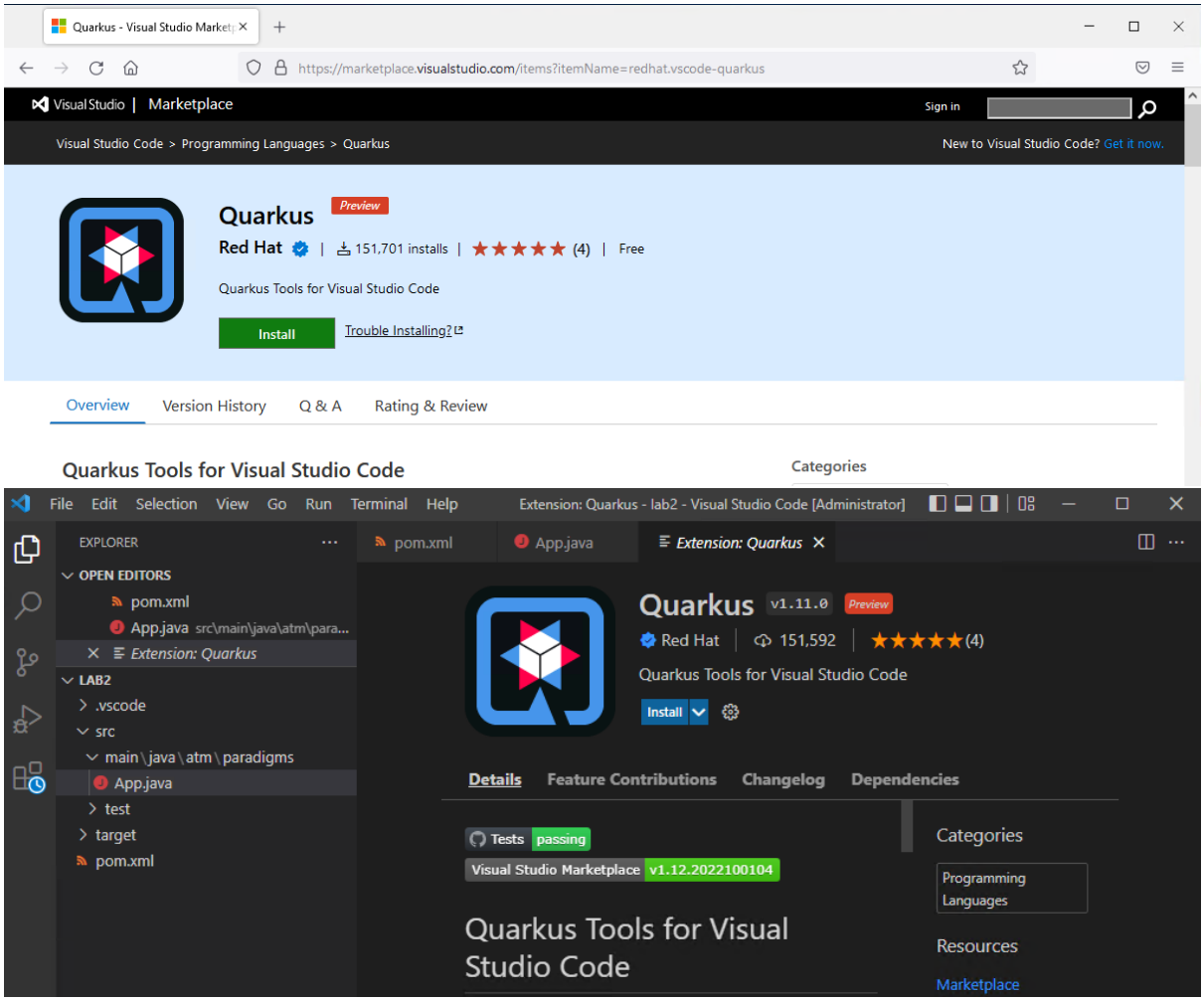
Paradigme de programare (în Java)

Lab 11/Curs 11- Quarkus Framework

Col(r) Traian Nicula

A. Instalare extensie Quarkus pentru VSC

- Se accesează în browser URL https://marketplace.visualstudio.com/items?itemName=redhat.vscode-quarkus și se instalează



- Se deschide Internet Explorer și se acceptă configurările de securitate (dacă nu a fost deschis niciodată) înainte de rularea comenzilor Powershell de mai jos.
- Se rulează în terminalul VSC (Powershell) comenzile pentru instalarea ***quarkus-cli***:

```
iex "& { $(iwr https://ps.jbang.dev) } trust add
https://repo1.maven.org/maven2/io/quarkus/quarkus-cli/"

iex "& { $(iwr https://ps.jbang.dev) } app install --fresh --force
quarkus@quarkusio"
```

- Dacă nu poate descărca (eroare de conectare) se rulează comanda de mai jos, iar apoi cele de deasupra din nou

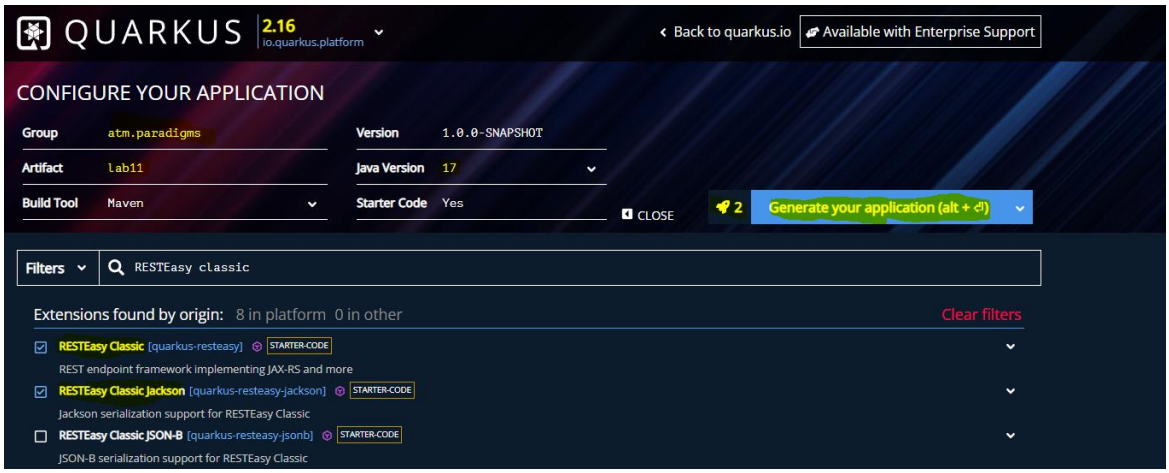
```
[Net.ServicePointManager]::SecurityProtocol =
[Net.SecurityProtocolType]::Tls, [Net.SecurityProtocolType]::Tls11,
[Net.SecurityProtocolType]::Tls12, [Net.SecurityProtocolType]::Ssl3

[Net.ServicePointManager]::SecurityProtocol = "Tls, Tls11, Tls12,
Ssl3"
```

- Este necesară **instalarea Java SE 17** și **actualizarea variabilei de sistem JAVA_HOME** la noua cale

B. Se creează proiectul Quarkus ***lab11*** folosind interfața web ***Quarkus Configure your application*** (https://code.quarkus.io/) astfel:

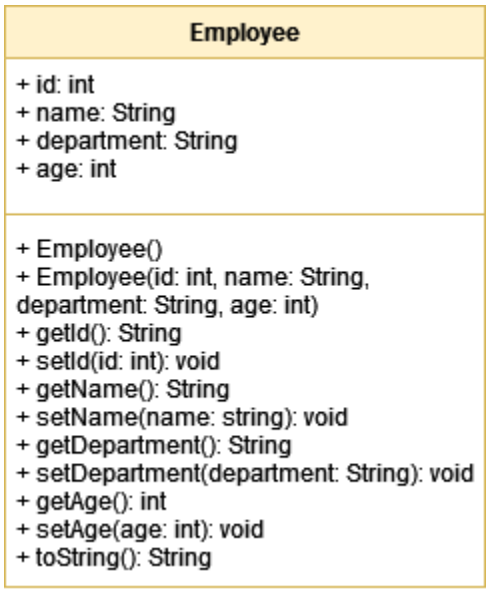
- Se deschide în browser link-ul https://code.quarkus.io/
- Se completează datele ca în figura de mai jos și se aleg extensiile *quarkus-resteasy* și *quarkus-resteasy-jackson*



- Se apasă butonul ***Generate your application*** și apoi ***DOWNLOAD THE ZIP***
- Se extrage folder-ul ***lab11*** din arhivă și se copiază în folder-ul cu numele studentului
- Se deschide proiectul lab11 în VSC

Se rezolvă următoarele exerciții:

1. Scrieți un program Java care: (**2.5p**)
   - creează clasa ***Employee*** conform diagramei UML de mai jos



   - creează clasa ***Tools*** cu metoda statică *List<Employee> getPeople()* care returnează o listă conținând 5 instanțe ale clasei Employee
   - creează clasa ***EmployeesService*** cu scop la nivel de aplicație, având metodele:
     a. *List<Employee> getEmployees()* care returnează o listă conținând 5 instanțe ale clasei *Employee.* Se folosește *Tools.getPeople()*
     b. *Employee getEmployeeById(int id)* care returnează angajatul cu id specificat. Creează local o copie a listei originale folosind Stream API pe care o filtrează după id angajat (se presupune că există).
2. Scrieți un program Java care: (**2.5p**)
   - creează clasa resursă ***EmployeesResource*** cu URI **/employees** având metoda *Response getEmployees ()* care:
     a. poate fi accesată cu metoda **GET** la URI **/employees**
     b. injectează o instanță a clasei ***EmployeesService*** și folosește metoda *getEmployees()* pentru a returna o reprezentarea JSON a listei angajaților (codul de stare 200 OK)
   - pornește Quarkus din terminal cu comanda ***quarkus dev*** și testează rezultatul în browser (http://localhost:8080/employees).
3. Scrieți un program Java care: (**2.5p**)
   - adaugă la clasa resursă ***EmployeesResource*** metoda *Response getEmployeeById(int id)* care:
     a. poate fi accesată cu metoda **GET** la URI **/employees/{id}**
     b. folosind metoda *getEmployeeById(int id)* a clasei injectate ***EmployeesService*** returnează reprezentarea angajatului în format JSON (cod de stare 200 OK)
   - dacă nu este deja pornit, pornește Quarkus din terminal cu comanda ***quarkus dev*** și testează rezultatul în browser (http://localhost:8080/employees/id).
4. Scrieți un program Java care: (**2.5p**)

- adaugă la clasa resursă ***EmployeesService*** un punct de injecție de tip *Event<Employee>* pentru a declanșa un eveniment
- modifică metoda *Employee getEmployeeById(int id)* pentru a declanșa un eveniment ori de câte ori acesta este invocată

Temă pentru acasă:

5. Scrieți un program Java care: (**1p**)
   - creează clasa ***MonitorService,*** de tip Observer, cu scop Singleton care:
     a. injectează o instanță a clasei *org.jboss.logging.Logger*
     b. implementează metoda *void searchEvent(@Observers Employee employee)* care folosește metoda *info()* a clasei ***Logger*** pentru a afișa obiectul accesat
   - dacă nu este deja pornit, pornește Quarkus din terminal cu comanda ***quarkus dev*** și testează rezultatul în browser (http://localhost:8080/employees/id). Observă în terminal mesajul afișat de Logger
6. Scrieți un program Java care: (**1p**)
   - adaugă în fișierul de configurare *application.properties* proprietățile:

```
employee.id = -1
employee.name = Unknown
employee.department = Unknown
employee.age = -1
```

   - creează clasa ***DefaultEmployee*** cu scop aplicație care:
     a. citește proprietățile de mai sus în propriile câmpuri proprii
     b. le folosește în metoda *Employee getInstance()* pentru a crea și returna o instanță a clasei *Employee,* inițializată cu aceste valori
     c. modifică metoda *getEmployeeById(int id)* a clasei ***EmployeesService,*** astfel încât atunci când id nu există în listă să returneze instanța generată de clasa ***DefaultEmployee***
   - dacă nu este deja pornit, pornește Quarkus din terminal cu comanda ***quarkus dev*** și testează rezultatul în browser (http://localhost:8080/employees/id)