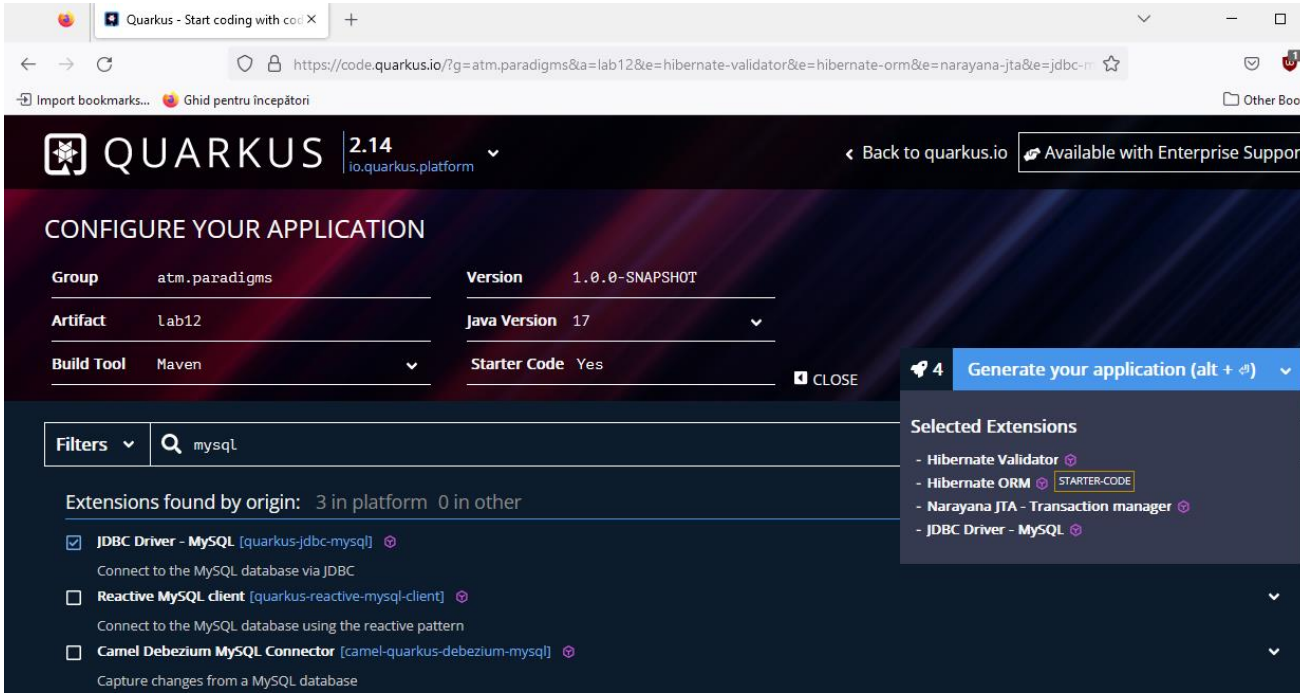
Academia Tehnică Militară "Ferdinand I“
Facultatea de Sisteme Informatice şi Securitate Cibernetică

Paradigme de programare (în Java)

Lab 12/Curs 12- Quarkus Framework

Col(r) Traian Nicula

A. Se creează proiectul Quarkus ***lab12*** 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-hibernate-validator, quarkus-hibernate-orm, quarkus-narayana-jta, quarkus-jdbc-mysql*



   - Se apasă butonul ***Generate your application*** și apoi ***DOWNLOAD THE ZIP***
   - Se extrage folder-ul ***lab12*** din arhivă și se copiază în folder-ul cu numele studentului
   - Se deschide proiectul ***lab12*** în VSC
   - Se creează în proiect folder-ul ***test/java/atm/paradigms*** sub ***src***
   - Se adaugă în fișierul ***application.properties*** proprietățile de vai jos. Parola este cea stabilită la instalare.

```
# datasource configuration
quarkus.datasource.db-kind = mysql
quarkus.datasource.username = user
```

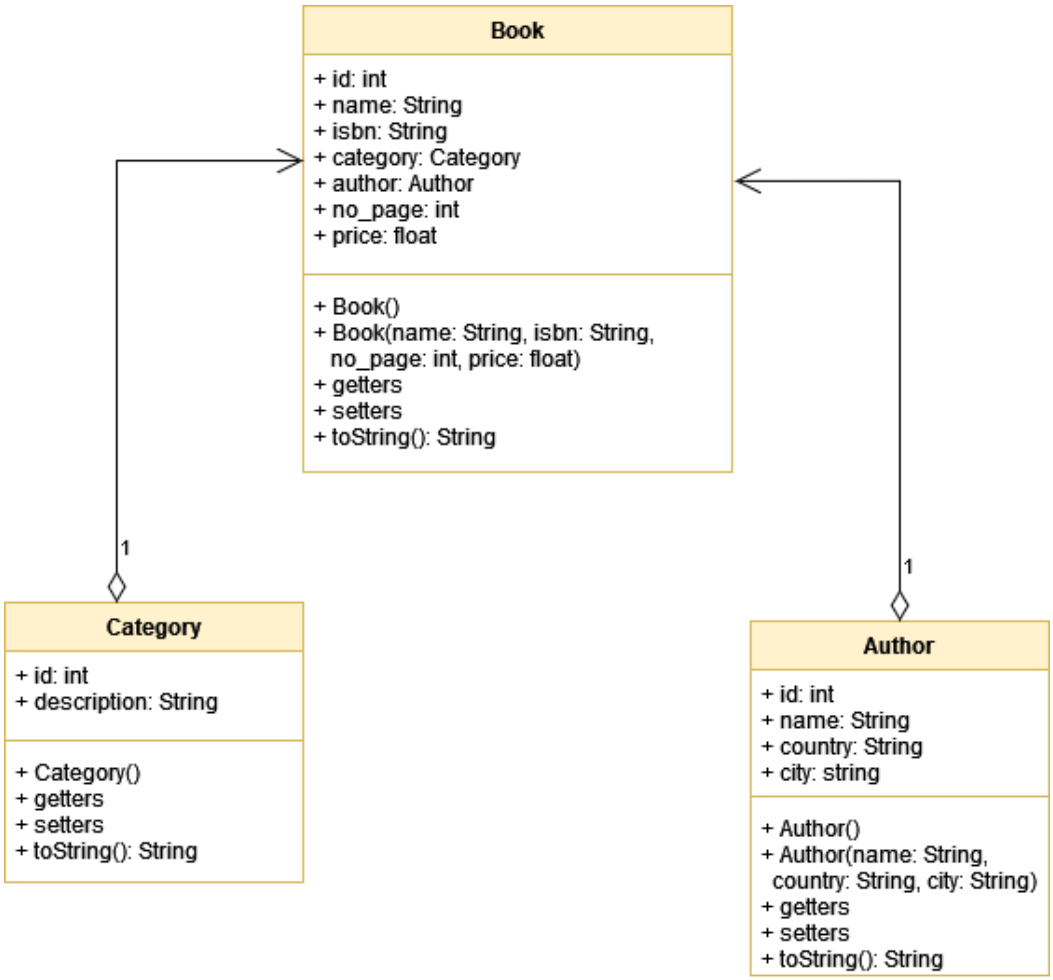
```
quarkus.datasource.password = qaz123QAZ!@#
quarkus.datasource.jdbc.url = jdbc:mysql://localhost:3306/testdb

# drop and create the database at startup
quarkus.hibernate-orm.database.generation=drop-and-create
```

- Se copiază în fișierul ***import.sql*** conținutul fișierului ***lab12.sql*** și se salvează.

Se rezolvă următoarele exerciții:

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



   - convertește clasa ***Category*** în entitate și adaugă constrângerile:

Câmp	Constrângere

id	Cheie primară autogenerată
	`@Id`
	`@SequenceGenerator(name = "categorySequence", sequenceName = "hibernate_sequence", allocationSize = 1, initialValue = 50)`

	`@GeneratedValue(generator = "categorySequence")`
description	Nenul cu dimensiunea maximă de 25 caractere

- convertește clasa ***Author*** în entitate și adaugă constrângerile:

Câmp	Constrângere

id	Cheie primară autogenerată
	`@Id`
	`@SequenceGenerator(name = "authorSequence",`
	`sequenceName = "hibernate_sequence",`
	`allocationSize = 1, initialValue = 50)`
	`@GeneratedValue(generator = "authorSequence")`

name	Nenul
country	Nenul
city	Nenul

- convertește clasa ***Book*** în entitate și adaugă constrângerile:

Câmp	Constrângere

id	Cheie primară autogenerată
	`@Id`
	`@SequenceGenerator(name = "bookSequence",`
	`sequenceName = "hibernate_sequence",`
	`allocationSize = 1, initialValue = 50)`
	`@GeneratedValue(generator = "bookSequence")`

name	Nenul
isbn	Nenul cu dimensiunea maximă de 15 caractere
category	Nenul
author	Nenul
no_page	Nenul, pozitiv
price	Partea întreagă are 4 cifre, partea fracțională are 2 cifre

2. Scrieți un program Java care: (**2.5p**)
   - creează clasa de test ***AuthorTest*** care injectează o instanță a ***EntityManager***
   - implementează metoda de test *void insertAuthor()* care:
     a. creează o instanță a clasei ***Author*** (ex. new Author("Marin Preda", "Romania", "Bucharest"))
     b. persistă obiectul în baza de date
     c. verifică cu metoda *assertNotNull* dacă obiectul are atribuită valoare din baza de date pentru câmpul id
     d. șterge obiectul din baza de date
   - verifică testul cu comanda ***.\mvnw test***
3. Scrieți un program Java care: (**2.5p**)
   - creează clasa de test ***CategoryTest*** care injectează o instanță a ***EntityManager***
   - implementează metoda de test *void insertCategory()* care:
     a. creează o instanță a clasei ***Category***
     b. persistă obiectul în baza de date

c. verifică cu metoda *assertNotNull* dacă obiectul are atribuită valoare din baza de date pentru câmpul id
d. șterge obiectul din baza de date

- verifică testul cu comanda ***.\mvnw test***

4. Scrieți un program Java care: (**2.5p**)
   - adaugă la clasa ***BookTest*** metoda *void insertBook()*, executată prima, care:
     a. creează o instanță a clasei ***Book*** (ex. new Book("Morometzii", "9786069098103", 1024, 70.23F))
     b. creează o instanță a clasei ***Category*** (ex. category.setDescription("Literature"))
     c. creează o instanță a clasei ***Author*** (ex. new Author("Marin Preda", "Romania", "Bucharest"))
     d. atribuie obiectului ***Book*** cele 2 obiecte create: ***Category*** și ***Author*** persistă obiectele în baza de date
     e. tipărește **id**-urile pentru obiectele ***Category*** și ***Author***
     f. verifică cu metoda *assertNotNull* dacă obiectul ***Book*** are atribuită valoare din baza de date pentru câmpul id
   - verifică testele cu comanda ***.\mvnw test***

Temă pentru acasă:

5. Scrieți un program Java care: (**1p**)
   - creează clasa ***BookTest*** care injectează o instanță a ***EntityManager*** și una a ***Validator***
   - implementează metoda de test *void validateBook()* care:
     a. creează o instanță a clasei ***Book*** (ex. new Book("Morometzii", "9786069098103", 1024, 70.235F))
     b. verifică cu metoda *assertEquals* numărul de constrângeri care sunt încălcate
   - verifică testele cu comanda ***.\mvnw test***
6. Scrieți un program Java care: (**1p**)
   - adaugă la clasa ***BookTest*** metoda *void insertOtherBook()*, executată a doua, care:
     a. se execută în ordine după metoda *void insertBook()*
     b. creează o instanță a clasei ***Book*** (ex. new Book("Viata ca o prada", "9789737883988", 304, 33.98F))
     c. folosește metoda ***find*** a ***EntityManager*** pentru a regăsi instanțele ***Category*** și ***Author*** (folosește id-urile din valorile tipărite la punctul 4 sau din baza de date)
     d. atribuie obiectului ***Book*** cele 2 obiecte create: ***Category*** și ***Author***
     e. persistă obiectul Book în baza de date
     f. verifică cu metoda *assertNotNull* dacă obiectul ***Book*** are atribuită valoare din baza de date pentru câmpul id
   - verifică testele cu comanda ***.\mvnw test***