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

Paradigme de programare (în Java)

Lab 7/Curs 7- Programare funcțională

Col(r) Traian Nicula

Se creează un proiect Maven cu numele ***lab7***, folosind ca artefact **archetype-quickstart-jdk8** și se salvează în folder-ul cu numele studentului. Pentru fiecare exercițiu se va crea câte o clasă de test (Exercise1 etc.) cu metoda statică *main*, precum și alte clase cerute de exerciții.

Se rezolvă următoarele exerciții:

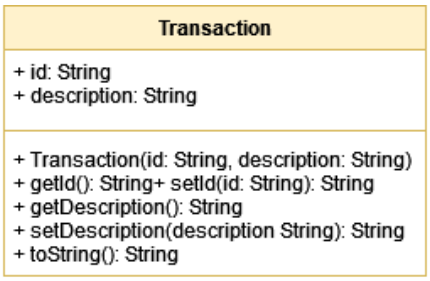
1. Scrieți un program Java care: (**1.5p**)
   - implementează o metodă statică *void sleep(int millis)* care oprește thread-ul curent pentru un număr de milisecunde trecute ca argument
   - implementează metoda statică *String getMessage()* care returnează un mesaj transmis întârziat (cu *sleep)*
   - implementează metoda statică *int getNumber()* care returnează un număr întreg aleator (generat cu Random.nextInt()) transmis cu întârziere
   - folosește *CompletableFuture.supplyAsync* pentru a invoca asincron metodele *getMessage* și *getNumber*
   - așteaptă finalizarea operațiilor cu *CompletableFuture.get* și afișează rezultatul.
2. Scrieți un program Java care: (**1p**)
   - folosește *CompletableFuture.supplyAsync* pentru a invoca asincron metoda *getMessage* de la exercițiul 1
   - aplică metoda *thenApply* pentru a concatena String-ul returnat cu un alt String
   - așteaptă finalizarea cu *CompletableFuture.get* și afișează rezultatul.
3. Scrieți un program Java care: (**1p**)
   - folosește metodele dezvoltate la punctul 1 pentru a înlănțui cele 2 *CompletableFuture* cu metoda *thenCombine* și a concatena într-un singur String
   - așteaptă finalizarea cu *CompletableFuture.get* și afișează rezultatul.
4. Scrieți un program Java care: (**1.5p**)
   - implementează metoda statică *int getNumber(int n)* care returnează un număr întreg aleator, generat cu *Random.nextInt(100) * n*, transmis cu întârziere (vezi metoda *sleep* de la punctul 1)
   - creează o listă de 10 numere întregi
   - folosește Stream API pentru a invoca asincron metoda *getNumber*
   - folosește Stream API și *CompletableFuture.thenApply* pentru a calcula în mod sincron pătratul numărului generat anterior
   - așteaptă rezultatul tuturor operațiilor cu *CompletableFuture.join* și folosește operația terminală *forEach* pentru tipărirea rezultatelor.
5. Scrieți un program Java, folosind noul API pentru dată și timp, care: (**1p**)
   - creează o variabilă ce stochează data curentă

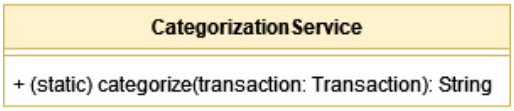
- creează o variabilă ce stochează timpul curent
- creează o variabilă care stochează data și timpul curent
- afișează valorile variabilelor de mai sus.

6. Scrieți un program Java, folosind noul API pentru dată și timp, care: (**1p**)
   - creează o variabilă ce stochează data și timpul curent
   - folosește metodele corespunzătoare pentru a afișa ziua din lună, luna, anul, ora și minutul.
7. Scrieți un program Java, folosind noul API pentru dată și timp, care: (**1p**)
   - creează o variabilă ce stochează data și timpul curent
   - afișează data și timpul curent folosind șablonul “dd.MM.yyyy HH:mm:ss”.
8. Scrieți un program Java, folosind noul API pentru dată și timp, care: (**1p**)
   - pornind de la data și timpul curent modifică ziua, luna și anul
   - afișează rezultatul folosind șablonul “dd.MM.yyyy HH:mm:ss”.
9. Scrieți un program Java, folosind noul API pentru dată și timp, care: (**1p**)
   - creează o variabilă ce stochează timpul local
   - creează o durată de 5 ore
   - adaugă durata la timpul curent
   - afișează rezultatul

Temă pentru acasă:

10. Scrieți un program Java, folosind noul API pentru dată și timp, care: (**0.5p**)
    - creează o variabilă ce stochează data locală
    - adaugă la dată cu metoda *plus* și enumerația *ChronoUnit* următoarele:
      a. 2 ani
      b. 1 lună
      c. 1 săptămână
    - afișează rezultatul folosind șablonul “dd.MM.yyyy”.
11. Scrieți un program Java care: (**1p**)
    - creează obiectele din diagrama UML de mai jos astfel:
      a. implementează metoda *categorize*, care returnează String-ul “Category_” concatenat cu ID-ul tranzacției
      b. răspunsul metodei *categorize* este transmis cu întârziere *(vezi metoda sleep de la punctul 1)*





- creează un Executor cu *Executors.newFixedThreadPool(10)*
- creează un stream de 8 tranzacții cu ID litere mari (*A, B,* etc.) iar descrierea de tipul: *description A, description B,* etc.
- folosește Stream API pentru a invoca asincron metoda *CategorizationService.categorize*
- așteaptă rezultatul tuturor operațiilor cu *CompletableFuture.join* și folosește operația terminală *toList* pentru colectarea rezultatelor
- tipărește lista rezultată.

12. Scrieți un program Java, folosind noul API pentru dată și timp, care: (**0.5p**)
    - stochează și afișează zona implicită
    - creează o variabilă care stochează data și timpul curent
    - creează o variabilă de tip *ZonedDateTime* pe baza variabilelor anterioare
    - creează zona “Asia/Tokyo”
    - calculează *ZonedDateTime* pentru noua zonă
    - afișează rezultatul folosind șablonul “dd.MM.yyyy HH:mm:ss”.