Софийски Университет "Климент Охридски"
Факултет по Математика и Информатика

Финален изпит No. 1

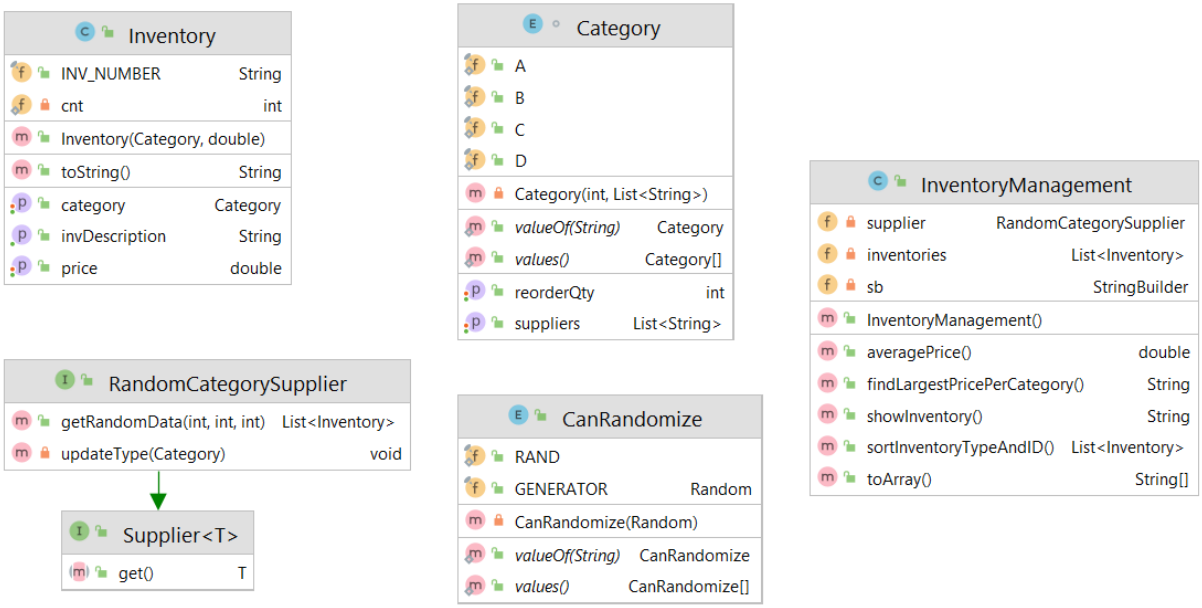
Курс: **Приложно Обектно Ориентирано Програмиране с Java, част 1**
Преподавател: проф. д-р. Е. Кръстев **Студент** :
Дата: юни 2021 **Време за работа: 120 min**

Инструкции: **Изпълнете следното задание и предайте в своя акаунт в Мудъл пълния набор от файлове на IntelliJ проекта, създаден за решаване на програмата. Пълен набор от точки се присъжда за пълно и коректно решение на всички подзадачи.**
Всеки студент носи отговорност за пълнотата на предаденото решение. Каквото е предадено, само това се оценява
Оценки:

- **2 от 0 до 54 точки**
- **3 от 55 до 64 точки**
- **4 от 65 до 74 точки**
- **5 от 75 до 84 точки**
- **6 от 85 до 100 точки**

Задача 1 (100 точки)

Задание за програмиране.



A. Създайте проект на IntelliJ с **Java модул** именуван като **`data.model`**, съответен на модула **`package data.model`** и файл **`module-info.java`** с описание на модула.. **Създайте в пакета `data.model` следните типове данни (показани в UMLклас диаграми по-горе за яснота)**

Точки: 5

1. **`Enum` тип `CanRandomize` съдържащ поле `RAND` със стойност от тип `java.util.Random`, инициализирана по подразбиране. Добавете към `CanRandomize` публично достъпна константа `GENERATOR` за извличане стойността на `RAND`**

Точки: 3

2. **Enum тип Category съдържащ поле A, B, C, D със стойности от тип int и List<String>, достъпни с данни reorderQty и suppliers. Добавете Getter Setter и конструктор за тези данни.**

Точки: 6

3. **Клас Inventory, който има уникален идентификатор ID, нестатична константа от тип String, а също category, price и description съответно от тип Category, double и нестатична константа от тип String. Инициализирайте тези данни в конструктор за общо ползване, където стойността на description се образува от префикса "Product-" към текущата стойност на ID. Добавете коректна дефиниция на Getter и Setter, където е необходимо. Предефинирайте метода toString() да извежда във форматиран вид данните на Inventory, както е показано в примерното изпълнение в края на текста.**

Точки: 12

4. **Напишете функционален интерфейс RandomCategorySupplier, който наследява функционалния интерфейс Supplier<List<Inventory>>. Добавете в този интерфейс следните методи:**

   a) **Метод**

   ```
   private void updateType(Category type)
   ```

   - **изтрива текущите елементи на suppliers референрани с параметъра type**
   - **използва GENERATOR на CanRandomize за добавяне на произволно избран брой наименования (от 1 до 4 вкл.) в списъка suppliers. Всяко наименование да започва с типа на категорията type(A, B, C или D), следвано от „Supplier " и пореден номер в списъка.**
   - **използва GENERATOR за задаване на произволно избрана стойност за reorderQty на type в интервала [1, 10 * броя на елементите на suppliers]**

   b) **Метод**

   ```
   default List<Inventory> getRandomData(int howMany, int a, int b)
   ```

   - **Изпълнява Ламбда израза, реферирaн с този интерфейс и актуализира свойствата на елементите на получения List<Inventory> по следния начин: Добавя howMany на брой обекти Inventory към този списък, където с помощта на GENERATOR се избира по произволен начин category от Category и price в интервала [a, b] за създавания обект Inventory.**
   - **Методът връща актуализирания List<Inventory>**

Точки: 20

5. **Напишете class InventoryManagement, чиито обекти имат данни supplier и inventories съответно от тип RandomCategorySupplier и List<Inventory>. Инициализирайте тези данни в конструктора по подразбиране. Данната supplier инициализирайте с Ламбда израз (или с реферирane на метод) за създаване на**

празен `List<Inventory>`**.Данната** `inventories` **инициализирайте с подразбиращия метод на интерфейса** `RandomCategorySupplier`, **със стойности на параметрите по Ваш избор (за идея вижте примерното решение)**

Точки: 4

6. **Напишете** следните методи в `class InventoryManagement`

   a) **Метод**

   `public String showInventory()`

   Използва задължително Stream API за създаване на `String`, **където елементите на** `inventories` **се изписват един след друг на отделни редове като се използва метода** `toString()` **на** `Inventory`

   (5 точки)

   b) **Метод**

   `public double averagePrice()`

   Използва задължително Stream API за пресмятане средната стойност на `price` **за елементите в списъка** `inventories`. **Методът връща пресметнатата стойност.**

   (5 точки)

   c) **Метод**

   `public List<Inventory> sortInventoryTypeAndID()`

   Използва задължително Stream API за сортиране на `inventories` **в низходящ ред на** `category` **и възходящ ред на** `ID` **на** `Inventory`. **Методът връща резултата от сортирането като** `List<Inventory>`.

   (5 точки)

   d) **Метод**

   `public String findLargestPricePerCategory()`

   Използва задължително Stream API за групиране елементите на `inventories` **по свойството** `category` **и за всяка група извежда най- голямата цена (**`price`**) измежду** `Inventory` **обектите в списъка** `inventories`, **които са от същата група. Методът връща текст с типовете** `category` **и пресметнатата най- голямата цена (**`price`**) за всяка група.**

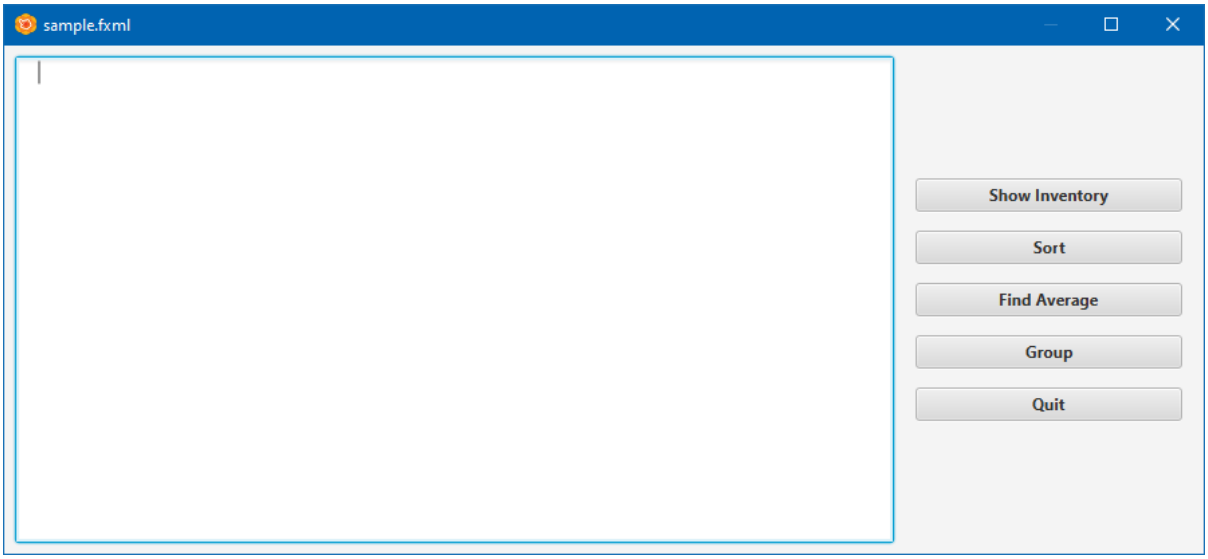
   (5 точки)

Общо точки: 20

В. **Добавете нов Java модул** именуван като `com.view` **към същия проект**, съответен на модула `package com.view` и файл `module-info.java` с описание на JavaFX модул. **Създайте в пакета** `com.view` **следните** FXML артефакти **(FXML сцена, съответни класна контролер и приложение)**

Точки: 5

1. **Създайте FXML сцена** която да възпроизвежда следния графичен модел, като използвате смислени имена за идентификатори по стила на т. нар. Модифицирана Унгарската нотация (за улеснение в края на текста е дадена примерна структура на JavaFX възлите)



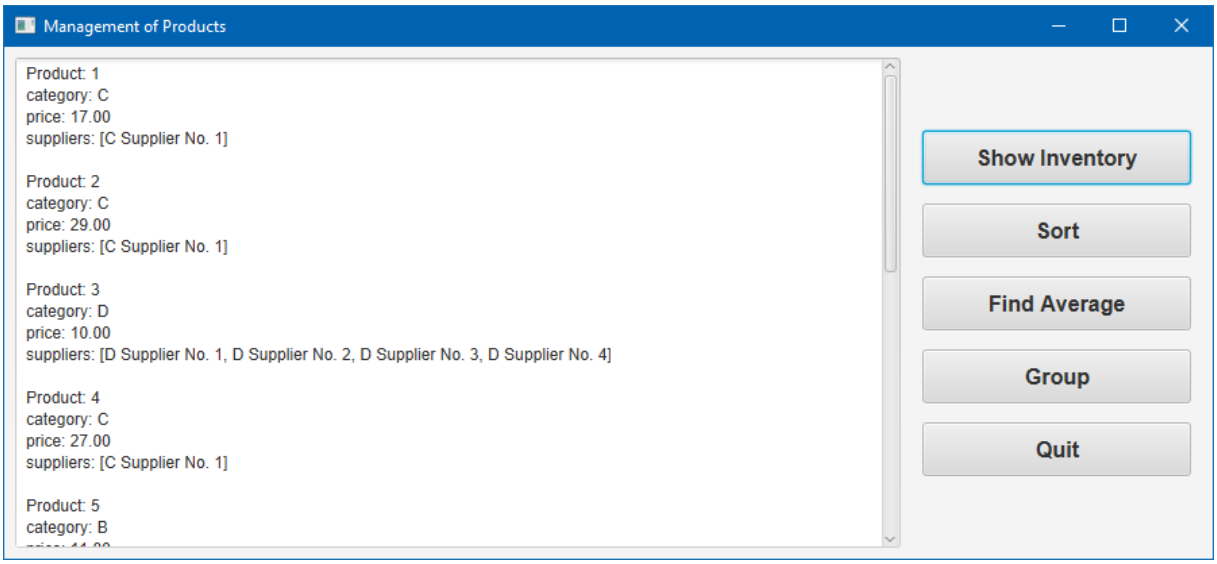
Точки: 10

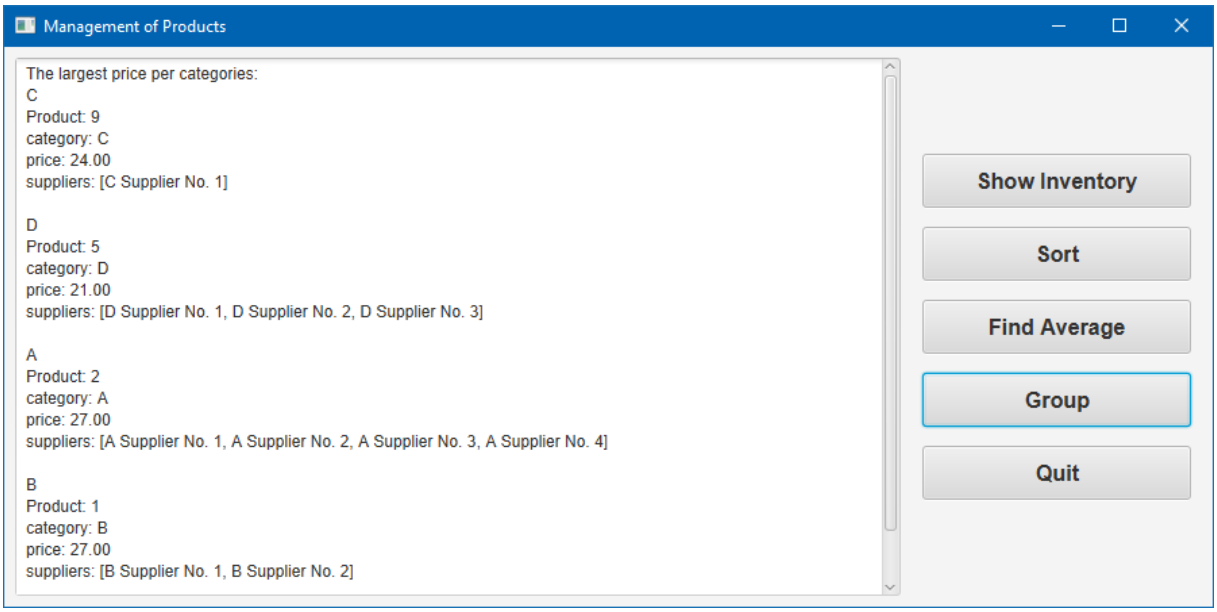
7. **Генерирайте съдържание на Контролера, съответно на тази Сцена. Добавете данна** `inventoryManagement` **от тип** `InventoryManagement` **в Контролера на FXML приложението. Актуализирайте описанието на модулите** `com.view` и `data.model`, **което позволява импортирането на клас** `InventoryManagement` **от модул** `data.model` **Инициализирайте по подразбиране** `inventoryManagement` **в метода** `initialize()` **на Контролера.**

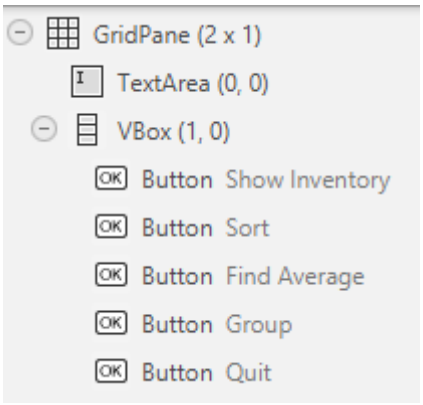
Точки: 5

8. **Напишете следната обработка на събитията, създавани при натискане на бутоните на графичния интерфейс (***да се използват методите на* `class InventoryManagement`**):**
   - **при натискане на бутона** `Show Inventory` **да се изведе в текстовата област резултата от изпълнението на метода** `showInventory()`
   - **при натискане на бутона** `Sort` **да се изведе в текстовата област резултата от изпълнението на метода** `sortInventoryTypeAndID()`
   - **при натискане на бутона** `Find Average` **да се изведе в текстовата област резултата от изпълнението на метода** `averagePrice()`
   - **при натискане на бутона** `Group` **да се изведе в текстовата област резултата от изпълнението на метода** `findLargestPricePerCategory()`
   - **при натискане на бутона** `Quit` **да се прекрати изпълнението на програмата**

Точки: 10


Management of Products
Product: 1
category: C
price: 17.00
suppliers: [C Supplier No. 1]
Product: 2
category: C
price: 29.00
suppliers: [C Supplier No. 1]
Product: 3
category: D
price: 10.00
suppliers: [D Supplier No. 1, D Supplier No. 2, D Supplier No. 3, D Supplier No. 4]
Product: 4
category: C
price: 27.00
suppliers: [C Supplier No. 1]
Product: 5
category: B
Show Inventory
Sort
Find Average
Group
Quit


Management of Products
The largest price per categories:
C
Product: 9
category: C
price: 24.00
suppliers: [C Supplier No. 1]
D
Product: 5
category: D
price: 21.00
suppliers: [D Supplier No. 1, D Supplier No. 2, D Supplier No. 3]
A
Product: 2
category: A
price: 27.00
suppliers: [A Supplier No. 1, A Supplier No. 2, A Supplier No. 3, A Supplier No. 4]
B
Product: 1
category: B
price: 27.00
suppliers: [B Supplier No. 1, B Supplier No. 2]
Show Inventory
Sort
Find Average
Group
Quit


GridPane (2 x 1)
TextArea (0, 0)
VBox (1, 0)
Button Show Inventory
Button Sort
Button Find Average
Button Group
Button Quit