Софийски Университет “Климент Охридски”
Факултет по Математика и Информатика

Финален изпит No. 1b

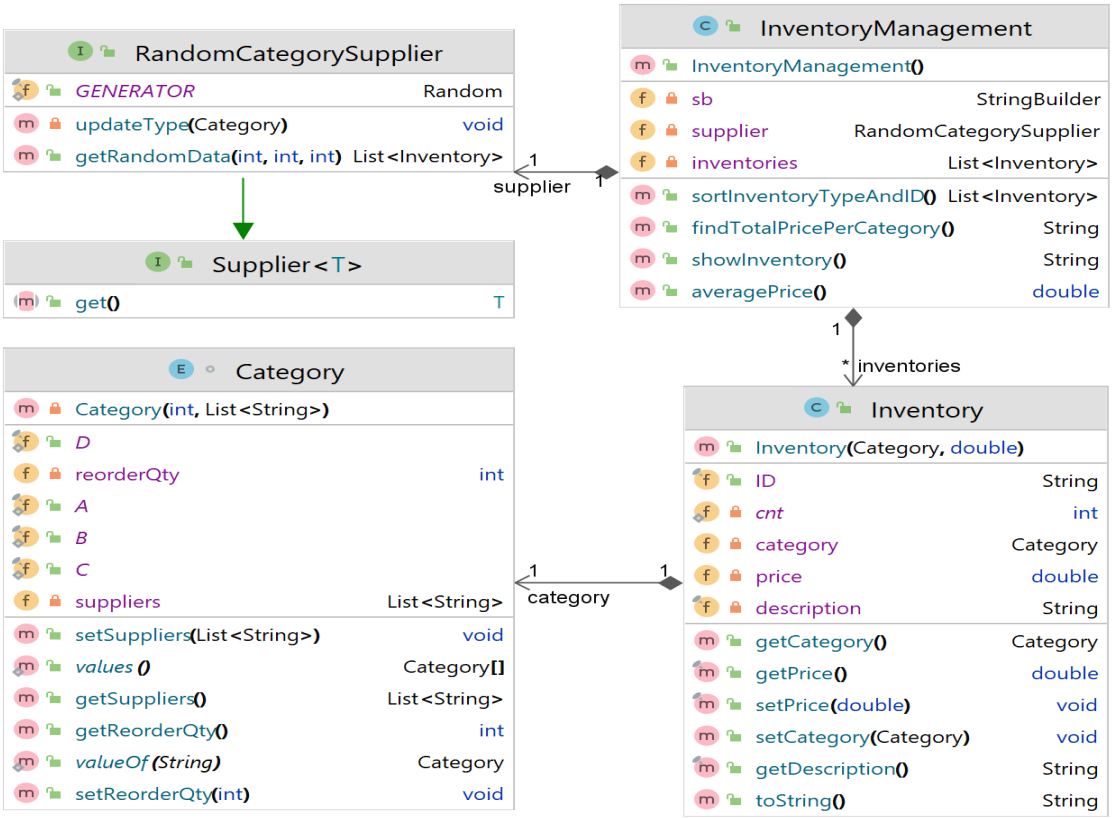
Курс: **Приложно Обектно Ориентирано Програмиране с Java, част 1**
Преподавател: проф. д-р. Е. Кръстев **Студент** :
Дата: юни 2022 **Време за работа: 120 min**

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 **`exam.logic`**, съответен на модула **`package exam.logic`** и файл **`module-info.java`** с описание на модула.. **Създайте в пакета `exam.logic` следните типове данни (показани в UMLклас диаграми по-горе за яснота)**

Точки: 6

1. **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

2. **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

3. **Напишете функционален интерфейс RandomCategorySupplier, който наследява функционалния интерфейс Supplier<List<Inventory>>. Добавете в този интерфейс:**
   - **Константа RANDOM от тип Random, инициализирана по подразбиране**

   и

   a) **Метод**

   `private void updateType(Category type)`
   - **изтрива текущите елементи на suppliers референцирани с параметъра type**
   - **използва RANDOM за добавяне на произволно избран брой наименования (от 1 до 5 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 RANDOM 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 get() метода на Supplier<List<Inventory>> 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 RANDOM се избира по произволен начин category от Category и price в интервала [a, b] за създавания обект Inventory. Преди добавянето на обекта Inventory в списъка се изпълнява updateType() за актуализиране на съдържанието на произволно избраната category.**
   - **Методът връща актуализирания List<Inventory>**

Точки: 20

4. **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 реферирание 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

5. **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`

   (4 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.**

   (4 точки)

   c) **Метод**

   `public List<Inventory> sortInventoryTypeAndID()`

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

   (5 точки)

   d) **Метод**

   `public String findTotalPricePerCategory()`

   Използва задължително Stream API за групиране елементите на `inventories` **по свойството** `category` **и за всяка група извежда общата сума на цените (**`price`**) на** `Inventory` `обектите в списъка inventories` **за съответната група. Методът връща текст с типа** `category` **и пресметнатата общата сума на цена (**`price`**) за всяка група, сортирани възходящо по групата** `category`.

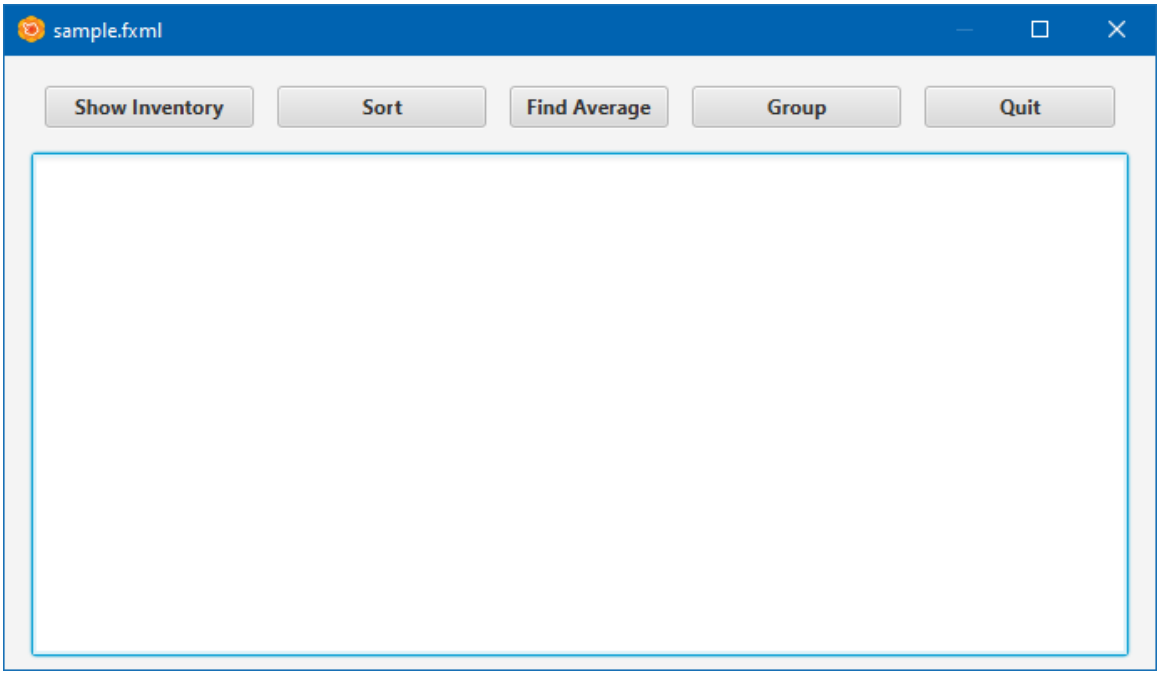
   (7 точки)

Общо точки: 20

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

Точки: 7

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)



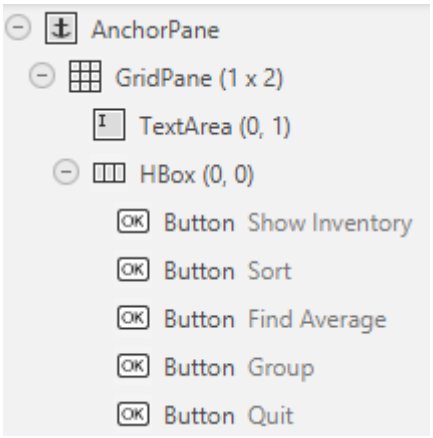
Точки: 10

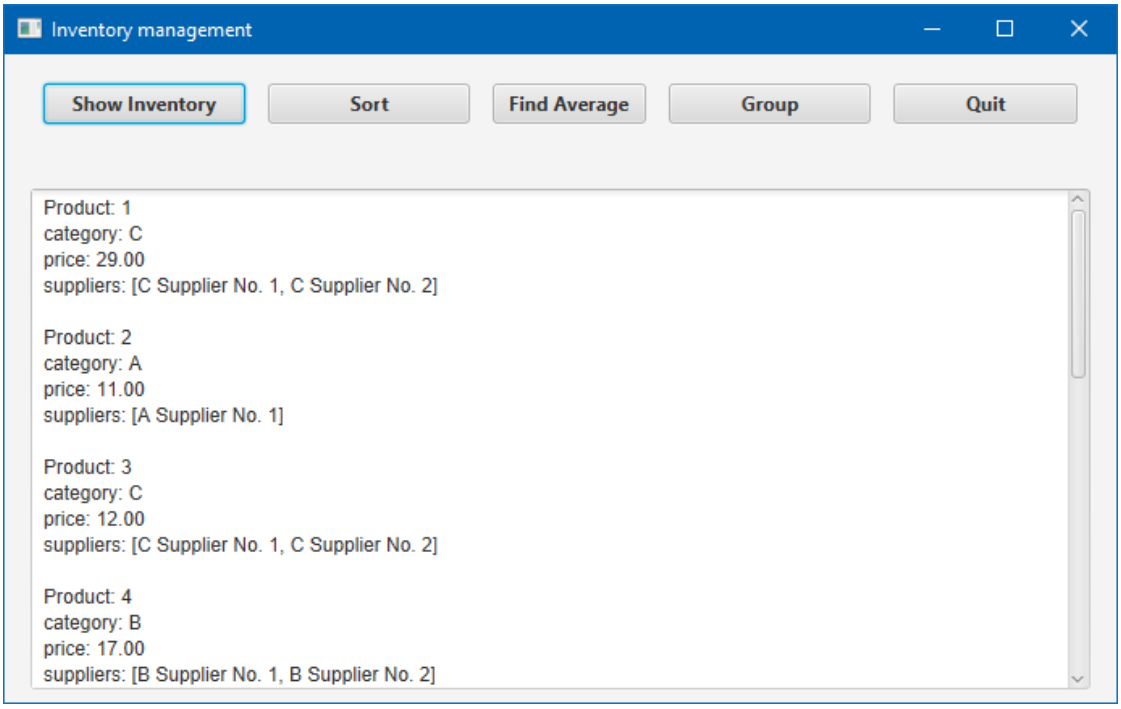
6. **Генерирайте съдържание на Контролера, съответно на тази Сцена. Добавете данна** `inventoryManagement` **от тип** `InventoryManagement` **в Контролера на FXML приложението. Актуализирайте описанието на модулите** `exam.view` и `exam.logic`,, **което позволява импортирането на клас** `InventoryManagement`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

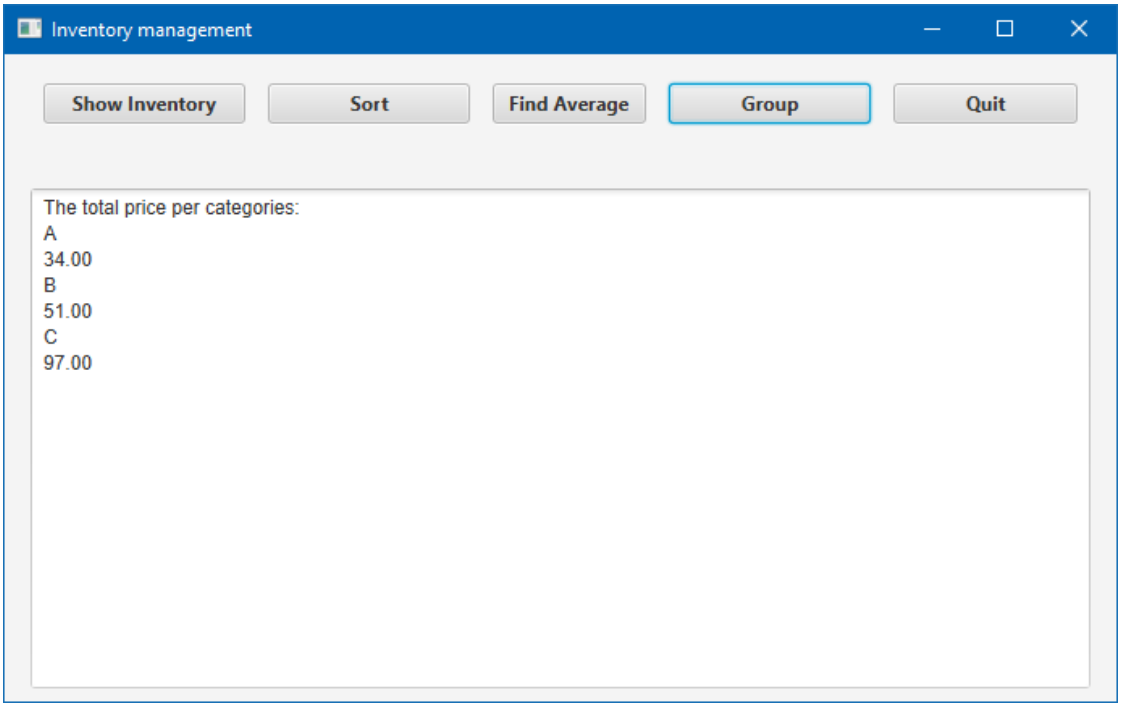
Точки: 5

7. **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** `findTotalPricePerCategory()`
   - **при натискане на бутона** `Quit` **да се прекрати изпълнението на програмата**

Точки: 10


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


Inventory management
Show Inventory
Sort
Find Average
Group
Quit
Product: 1
category: C
price: 29.00
suppliers: [C Supplier No. 1, C Supplier No. 2]
Product: 2
category: A
price: 11.00
suppliers: [A Supplier No. 1]
Product: 3
category: C
price: 12.00
suppliers: [C Supplier No. 1, C Supplier No. 2]
Product: 4
category: B
price: 17.00
suppliers: [B Supplier No. 1, B Supplier No. 2]


Inventory management
Show Inventory
Sort
Find Average
Group
Quit
The total price per categories:
A
34.00
B
51.00
C
97.00