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: **Време за работа: 120 min**

Инструкции: Изпълнете следното задание за обектно ориентирано програмиране 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.

Оценки:

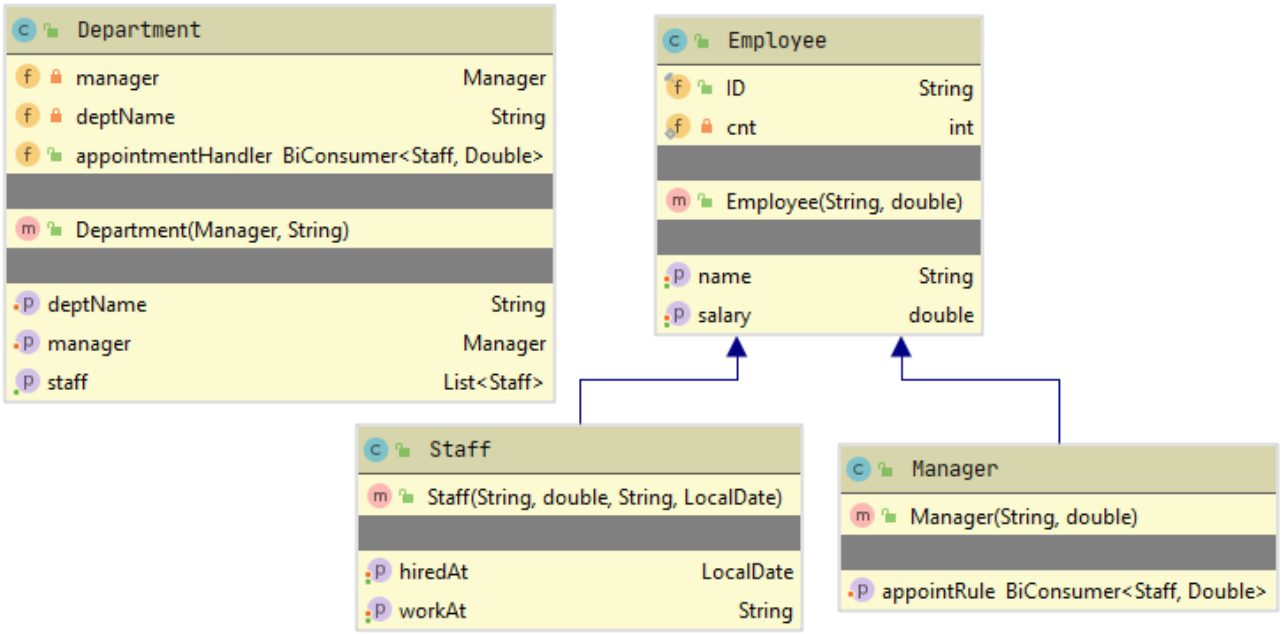
2	**от 0 до 54 точки**
3	**от 55 до 64 точки**
4	**от 65 до 74 точки**
5	**от 75 до 84 точки**
6	**от 85 до 100 точки**

Задача 1 (100 точки)

Приложете следните принципи на Обектно ориентираното програмиране на **Java**:

- **hiding of information**
- **software reuse**
- **inheritance**
- **polymorphism**

при намиране решението на следната **задача. Напишете *модулен проект*** на **Java,** посредством което **менажерът** на даден отдел **назначава личния състав** за своя отдел и **визуализира данни за назначените служители** за промяна на заплатите им. При това се приема, че менажерът (**`Manager`**) и членовете (**`Staff`**) на личният състав в отдела му са служители (**`Employee`**) **според дадената UML диаграма** по- долу.



Всеки отдел(**Department**) има наименование(**manageDeptName**), управител(**Manager**) и списък от служители(**staff**), назначени от **Manager** в този отдел. **При реализацията** на **събитието по назначаването** на член от състава **Manager** използва следното **бизнес правило**:

- задава се принадлежност на служителя към **Department,** управляван от **Manager**,
- задава се текущата дата на назначаване
- задава се работна заплата на служителя **Staff** и
- съответния служител **Staff** се добавя в списъка **staff** на личния състав на отдела.

За **по- голяма гъвкавост при реализацията на бизнес правилата** използвайте **функционален интерфейс** и **Ламбда израз** за **обработка на това събитие** като **извършете следните действия**

А. **Създайте модул** именуван като **model**, съответен Java **package** и файл **module-info.java** с описание на модула. (**4 точки**)

1. **Дефинирайте class Employee**, чиито инстанции **имат** *уникален идентификатор* ***ID,*** *име* ***name*** **и** *заплата* **salary. Създайте в class Employee**
   - **ID** като публично достъпна **константа** от тип **String** от вида **E-###**, където незначещите цифри в трицифреното число **###** са заменени с нули
   - **GET и SET методи за name и salary**. .По подразбиране ***name*** и ***salary*** имат стойности "**No Name**" и **1000**.
   - **Конструктор за общо ползване**
   - метод
     public String toString()
     за този клас – връща **String,** съдържащ текущите стойности на **ID, name и salary, форматирана с два знака** след десетичната запетая

     Точки:8

2. **Дефинирайте class Staff**, **който наследява Employee,** който **има** данна **workAt** от тип **String** и данна **hiredAt** от тип **LocalDate. Създайте в class Staff:**
   - **GET и SET методи за данните workAt** и **hiredAt на този клас**. Когато **workAt** е ***null***, то **workAt** приема подразбираща се стойност „**Candidate**", По подразбиране, **hiredAt** е текущата дата
   - **Конструктор за общо ползване**
   - метод
     public String toString()
     за този клас– връща **String**, съдържащ текущите стойности на всичките *данни* на инстанцията от **class Staff** .

     Точки: 8

3. **Дефинирайте class Manager**, **който наследява Employee.** Нека този клас **има данна appointRule от тип BiConsumer<Staff,Double> и конструктор за общо ползване. . Създайте в class Manager** методи**:**

   SET метод за данната appointRule

   и

   public void onStaffAppointment(Staff member, Double newStaffMemberSalary)
   който да изпълнява Ламбда израза, представян с **appointRule** с параметри **member** ***и*** **newStaffMemberSalary**

   Точки: 8

4. **Дефинирайте** `class Department`. Този клас **има** `manager` (`Manager` тип), наименование `deptName` (`String` тип) на `Department` и списък `staff` от слу**жителите** `Staff` **на** `Department`. Нека този клас **също има:**

   - **SET методи за данните** `manager` и `deptName`. Хвърля се изключение `java.security.InvalidParameterException`, когато `manager` или `deptName` *са* `null`
   - **GET метод за данната** `staff`
   - **Конструктор за общо ползване с аргументи от тип** `Manager` *и* `String` *за* инициализиране съответно на `manager` и `deptName` *(при създаване на нов* `Department` ***да се отчете****, че списъкът* `staff` *е празен)*
   - метод
     `public String toString()`
     за този клас – връща `String` , съдържащ текущите стойности на всичките *данни* на инстанцията от `class Department`

Точки: 8

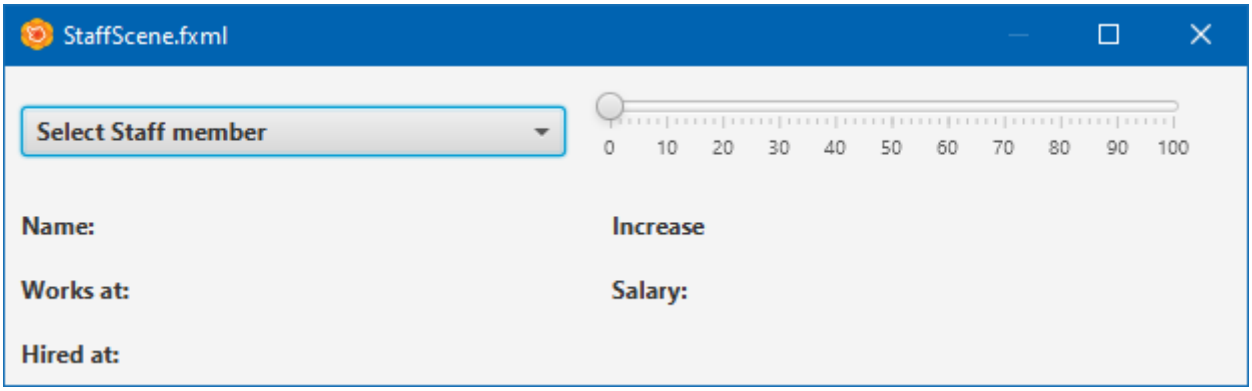
5. Нека `class Department` да задава **бизнес правилата** при **назначаване на нови членове** на личния състав от своя менажер.

   a) За целта добавете в `class Department` **публична** данна `appointmentHandler` от тип `BiConsumer<Staff,Double>` **(2 точки)**
   b) Инициализирайте `appointmentHandler` на Ламбда израз с параметри `Staff worker, Double workerSalary` , който Ламбда израз **(10 точки)**
      - Инициализира `workAt` свойството на `worker` на наименованието `deptName` *на* `Department`,
      - Инициализира `hiredAt` свойството на `worker` на текущата дата на назначаване
      - Инициализира `salary` свойството на `worker` на работна заплата на назначавания служител
      - Добавя служителя `worker` към списъка `staff` служители на отдела.

Точки: 12

Б. **Създайте JavaFX модул** именуван като `gui`, съответен Java `package` и файл `module-info.java` с описание на модула. (**6 точки**)

1. **Създайте FXML описание на 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)



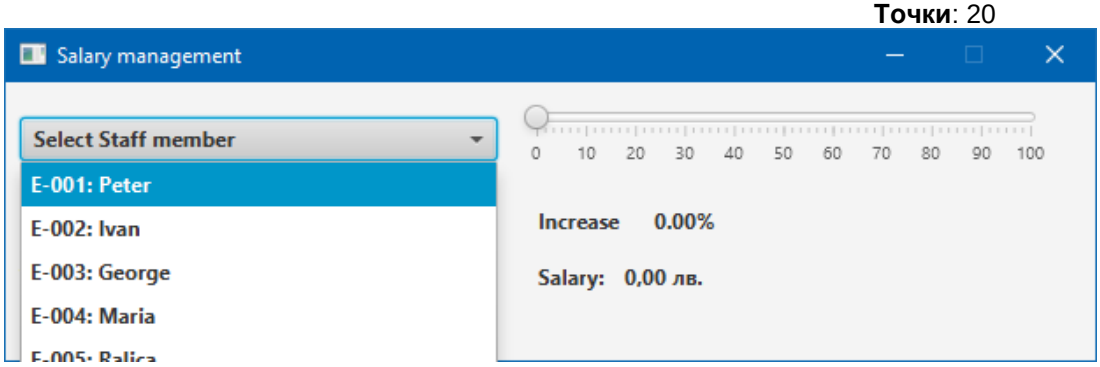
Точки: 22

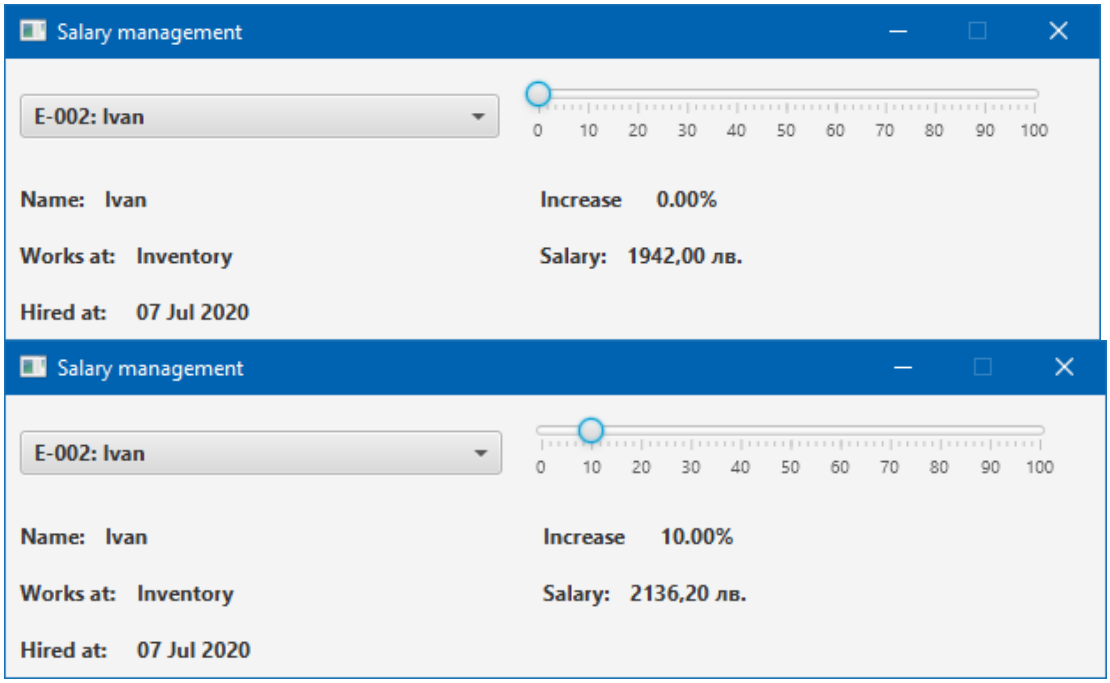
2. **Създайте** Контролер, съответен на **FXML сцената, и клас на Java за стартиране на FXML приложението.**

Точки: 4

3. Изпълнете следните действия в метода **`initialize()`** на Контролера (за справка ползвайте очакваното примерно изпълнение в края на текста):
   - **Създайте** масив **`candidates`** от шест- седем **`Staff`** обекти, **където *имената на служителите са по избор на студента***, а останалите данни в конструктора са подразбиращи се (0 или null). Създайте по аналогичен начин обекти **`manager`** и **`department`** съответно от тип **`Manager`** и **`Department`**
   - Инициализирайте данната **`appointRule на manager`** да реферира **`department.appointmentHandler`** Създайте обект **`rand`** от тип **`Random`** и в цикъл изпълнете метода **`onStaffAppointment`** на **`manager`**, за да назначите в **`department`** всеки един от **`candidates`**. Използвайте **`rand`**, за да генерирате стойност на **`salary`** (втория параметър на метода) на всеки един от кандидатите в интервала [**`1000`**, **`2000`**]
   - Създайте **`ObservableList<Staff >,`** чиито елементи да са елементите на списъка **`staff`** с назначени служители в **`department`** Заредете елементите на **`ObservableList<Staff`** в падащия списък (**`Select Staff element`**) като посредством **`StringConverter`** извеждете **`ID`** и **`name`** свойствата на обектите в този списък според примерното изпълнение в края на текста.
   - При **промяна** на JavaFX свойството **`selectedItemProperty()`** на падащия списък (избиране на **`Staff`** обект), нека етикетите за **`Name`**, **`Works At`**, **`Hired at`** да показват съответните стойности на избрания **`Staff`** обект
   - Свържете JavaFX свойството **`text`** на етикета отдясно на **`Increase`** със JavaFX свойството **`value`** на плъзгача, така че този етикет да показва във вид на процент текущата стойност на плъзчага
   - Създайте **`StringBinding`**, който да е форматиран като парична сума текст на заплатата (**`salary`**) на текущо избрания **`Staff`** обект от падащия списък, увеличена с процента, избран с плъзгача. Свържете JavaFX свойството **`text`** на етикета отдясно на **`Salary`** с така създадения **`StringBinding`** (извеждайте 0 лева, когато в началото няма избран обект от падащия списък)

Точки: 20





Примерна структура на JavaFX възлите