Христослав Захариев, фак. No: 1909011457, спец. „Софтуерно инженерство“, курс 2

Задача по Java за текуща оценка – Документация

Структура на класовете:

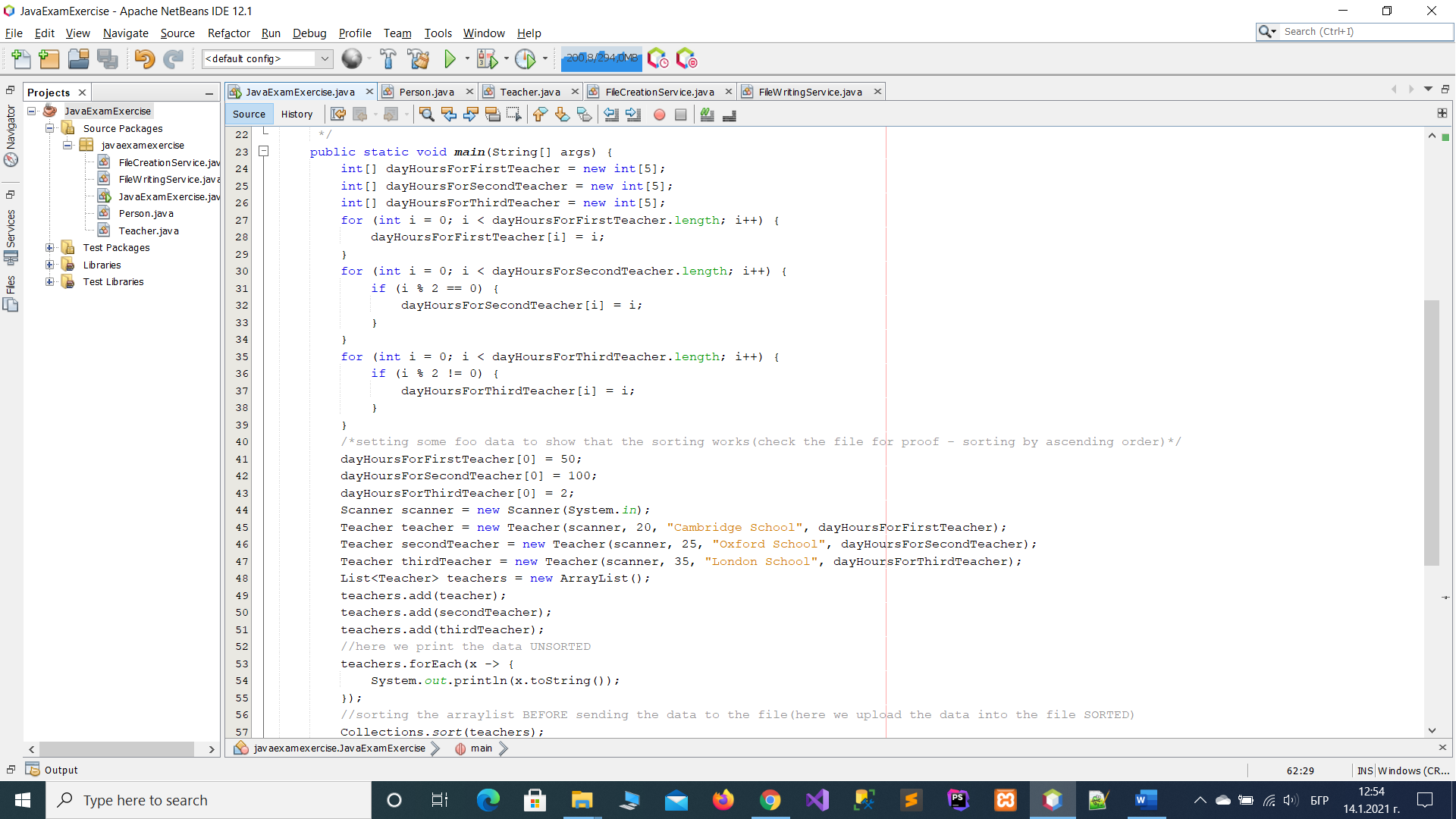
Имаме четири класа(два служещи за модели и два под формата на сървиси, изпълняващи бизнес логика).

Клас Person: Класът Person съдържа две полета – name – от тип String, в което се съдържа името на дадения човек и age(от тип int), в което се съдържа възрастта на дадения човек. Имаме по два гетъра и сетъра (свойства) за класа, извършващи базовата инициализираща + връщата стойността логика. Също така имаме два оувърлоуда на конструктора, единия, от които е празен, а другия оувърлоуд – задава името на дадения човек, използвайки вход от конзолата + свойството за „сет-ване“ на име. Освен това, се съдържа и „override-нат“ метод toString, който връща като резултат стойностите на полетата за текущата инстанция.

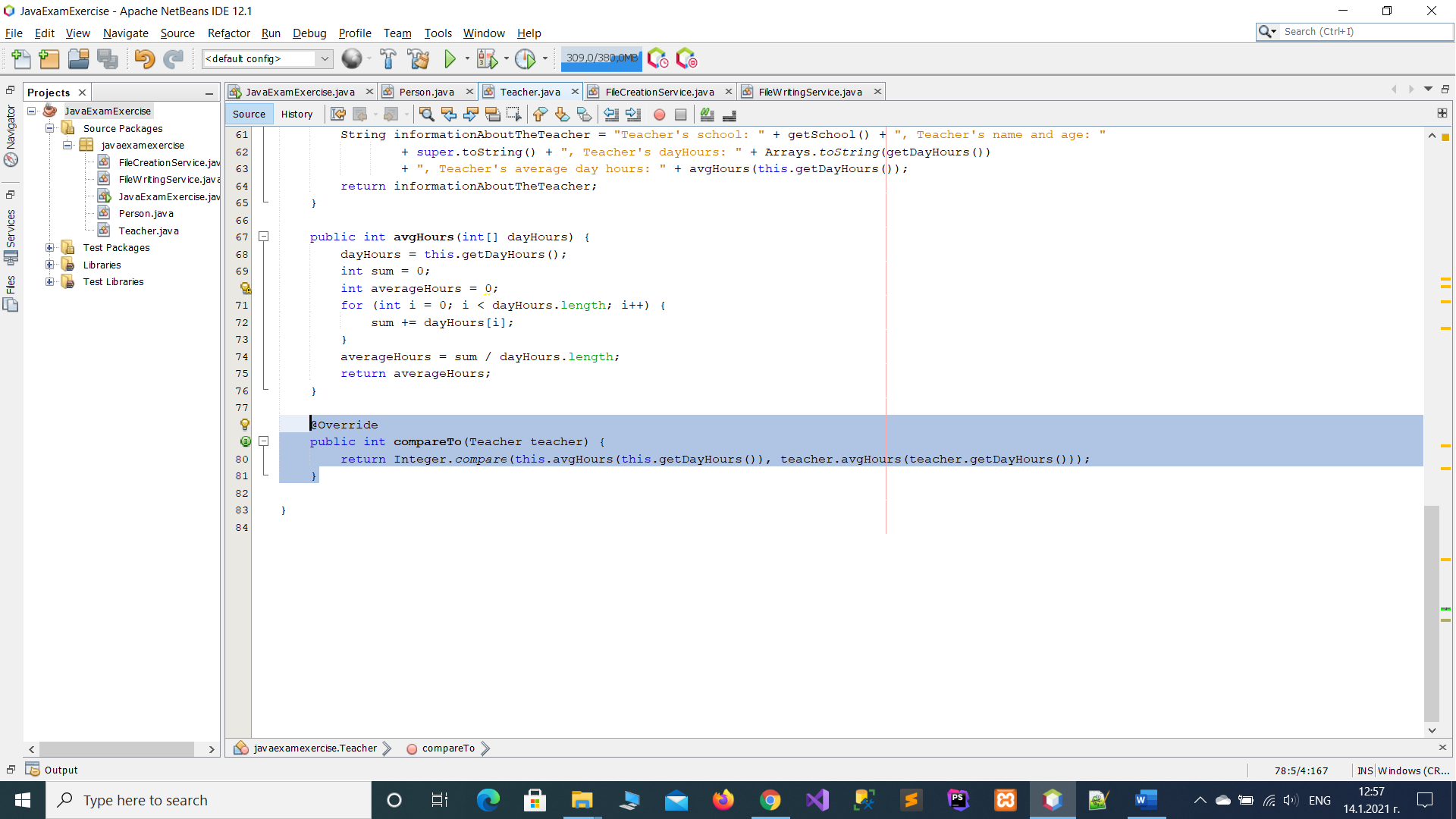
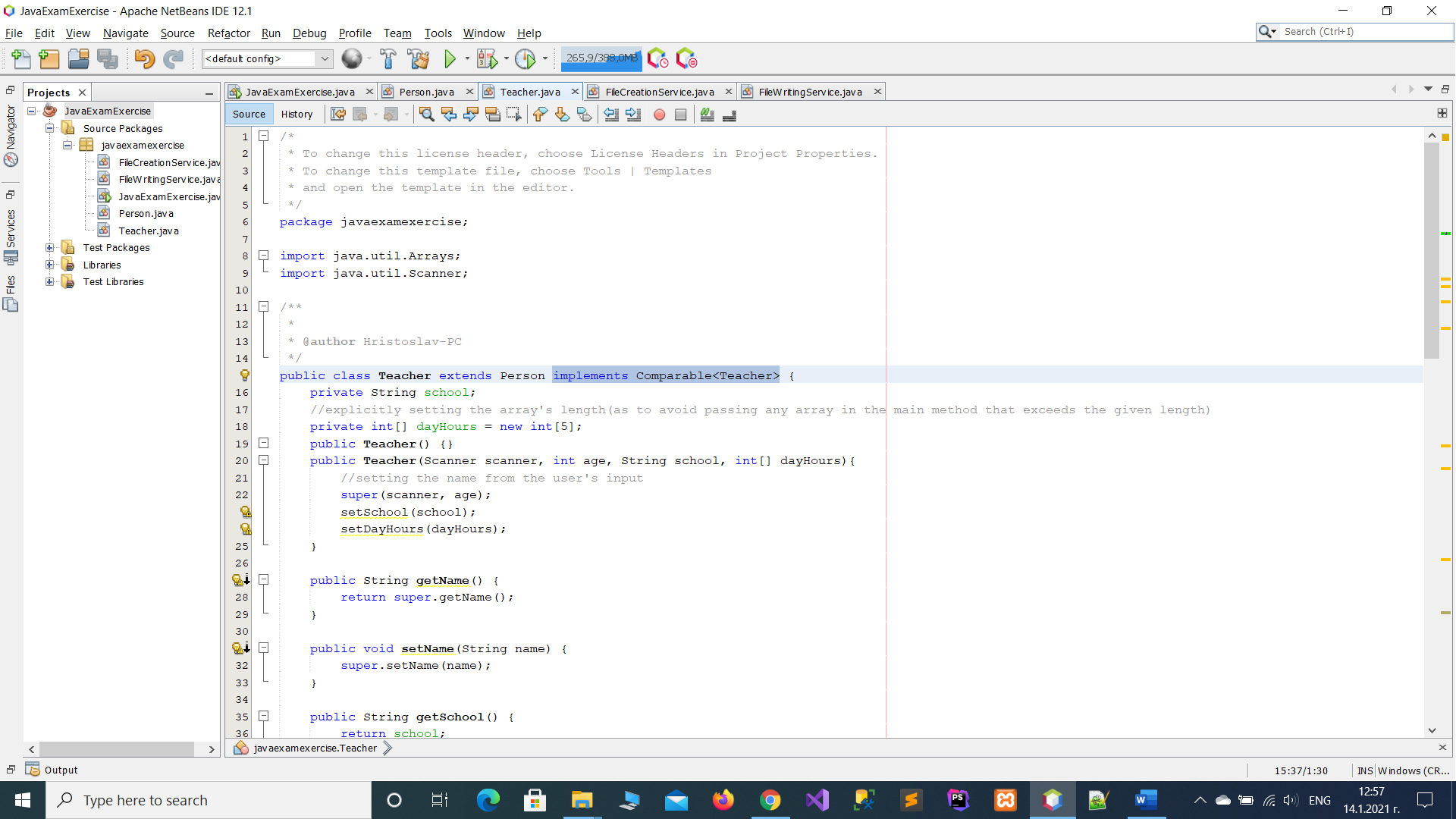
Преминаваме към класа Teacher. Той наследява класа Person и имплементира интерфейса(служещ за по – нататъшната сортировка на данни – Comparable<>, приемащ в генерика „<>“ дадения клас, с който ще работи – в случая: Comparable<Teacher>). Класът Teacher съдържа 4 полета – две, от които идват при наследяване на класа Person(т.е. age и name) и две полета, които описват училището, в което работи дадения учител, както и неговите работни часове – school от тип String и масив от „int-ове“ dayHours, зададен със „хард-коудната“ дължина – 5(по условие). Класът Teacher съдържа два оувърлоуда на конструктора – един празен и един, задаващ стойностите на гореспоменатите полета чрез свойствата + базовия конструктор. Също така, има пропъртита – гетъри и сетъри – където се извършва връщане на стойността или задаването й(в зависимост дали е „гет-ър“ или „сет-ър“). За които полета се налага, се използва базовата логика от класа-родител. Класът Teacher съдържа „оувъррайднат“ метод toString() връщат информация за полетата на текущата инстанция + средноаритметичните часове, взети от по – долу споменатия метод avgHours(). Методът avgHours(int[] dayHours) – приема дадените работни часове за седмицата на текущия учител и им намира средноаритметичната стойност. В него просто взимаме работните часове на даден учител, държим сумата от тези часове в една променлива, държим средноаритметичната стойност в друга променлива, и получаваме нейния резултат чрез пресмятане на сумата, разделена на броя елементи от масива(броя работни часове за дадената седмица). Връщаме този резултат и го използваме в toString() метода, както и в по – долу споменатия метод за сортировка в нарастващ ред.

Методът compareTo(Teacher teacher). Както вече споменахме, в класа Teacher имплементираме интерфейса Comparable<Teacher>, за да можем да пренапишем логиката на метода compareTo(). „Оувъррайд-нат“ e, като приема дадения учител, с който искаме да сравним на друг учител(този в текущата инстанция), работните часове. Връщаме като стойност 0, ако са им равни часовете, стойност по – малка от 0, ако на учителя, взет от подадения аргумент стойността на броя часове е по – висока и стойност по – голяма от 1, ако на текущата инстанция стойността на броя часове е по – висока от тази на подадения като аргумент учител.

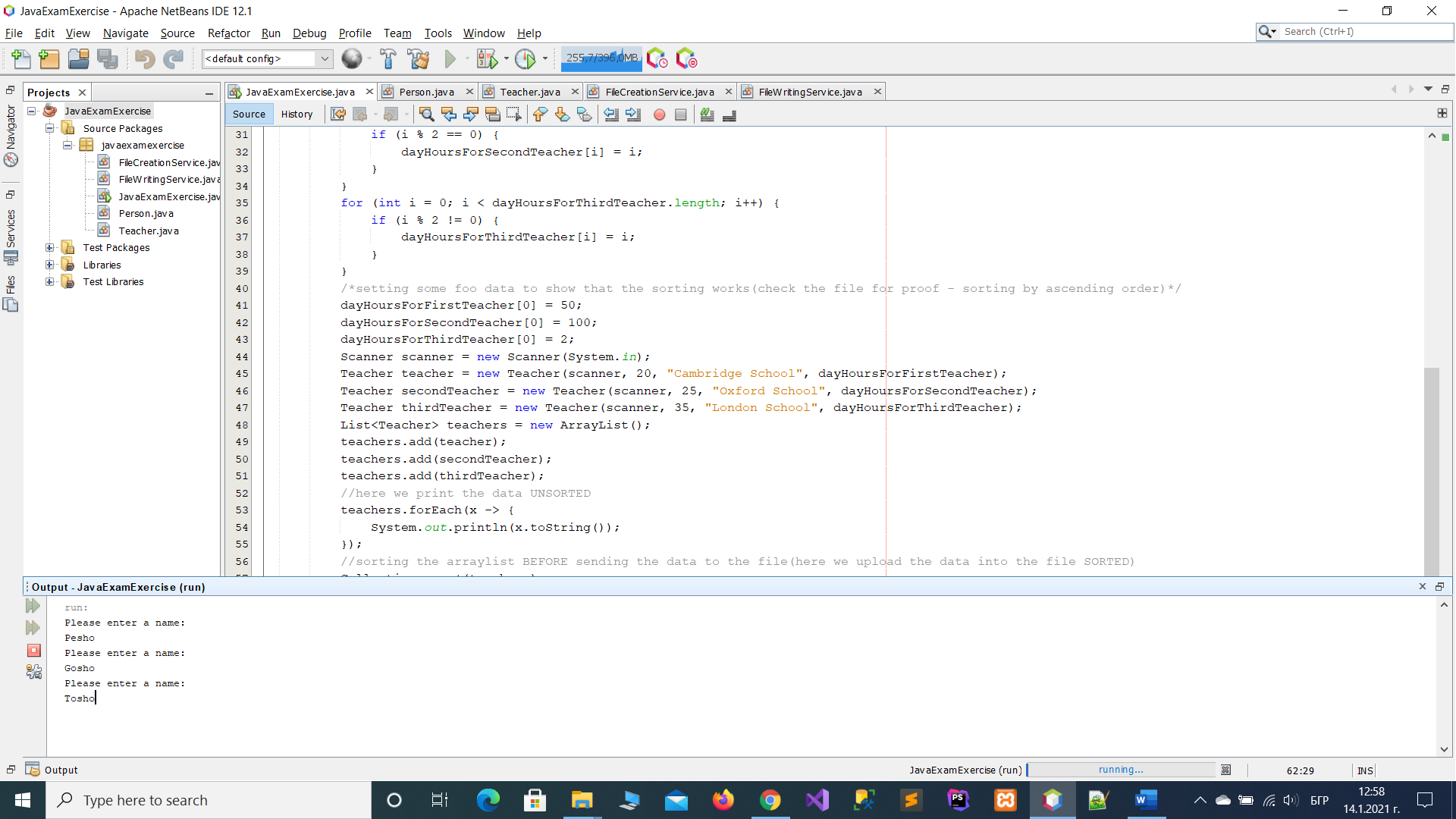
Класовете, изпълняващи функцията на сървиси – FileCreationService и FileWritingService, се използват за направата на(и връщането на името на файла) и писането по файл. Класът FileCreationService има метод createAFileAndReturnItsName(File file), съответно, приемащ файл от тип File и в него проверяваме дали дадения файл е бил създаден преди това, ако не е бил, го създаваме, казваме, че е бил създаден успешно, и казваме(изпринтваме на конзолата) къде е бил създаден, в противен случай, ако този файл вече, съществува, изпринтваме на конзолата тази информация и указваме къде съществува във файловата система. Всичко това е обвито в try-catch блок, за да може да се прихванат нежеланите runtime грешки. Като резултат от catch-a се изписва съобщение, указващо, че е станало грешка. Методът връща името на файла.

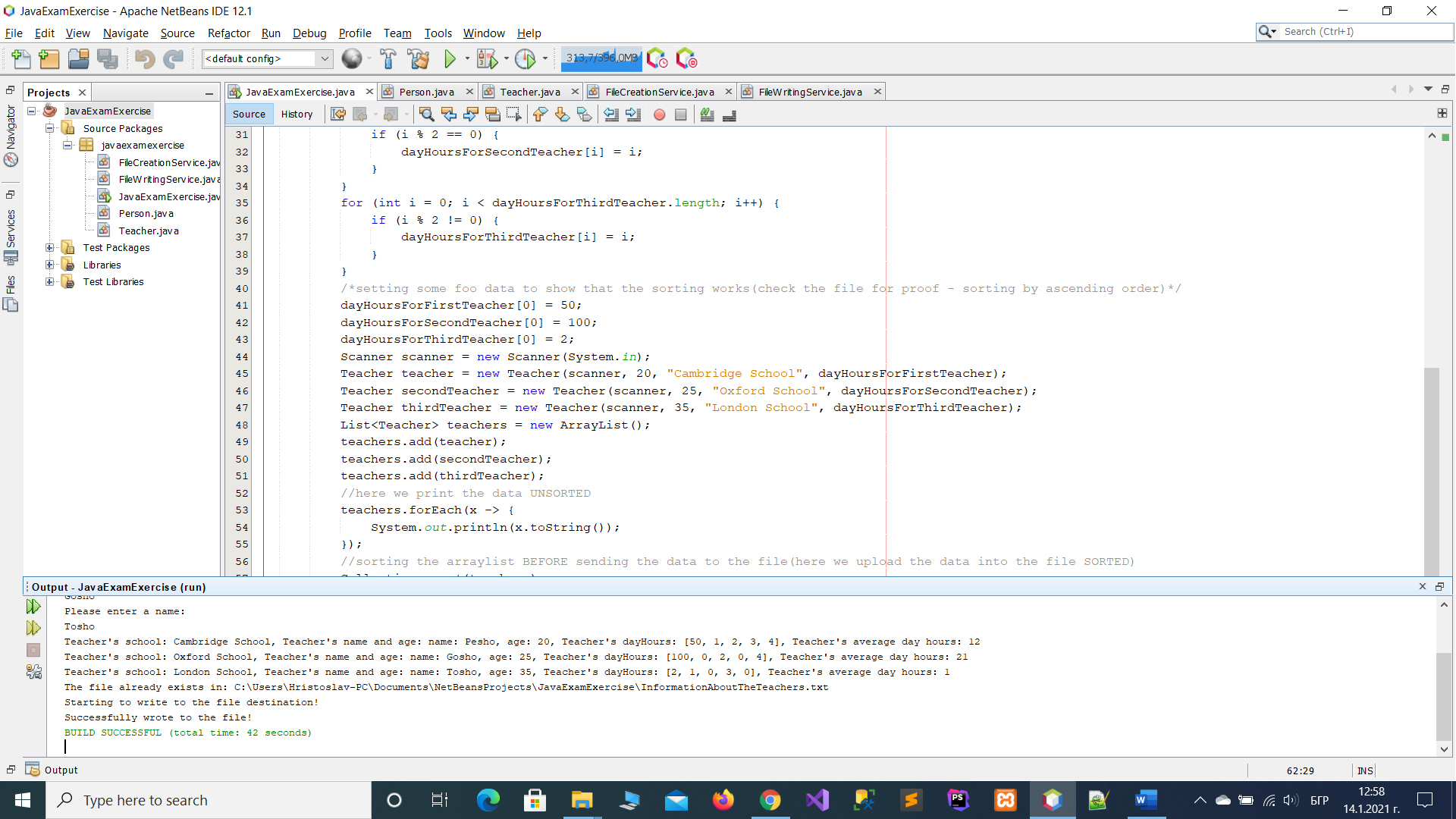
Класът FileWritingService от своя страна, има метод, на име WriteToAFile приема името на файла, по който ще пишем и List<> - съответно с генерик, приемащ типа на колекцията от учители – List<Teacher> teachers. Кодът вътре, аналогично, също е обвит от try-catch блок, за да се прихванат дадените изключения. Вътре си създаваме инстанция на класа FileWriter(отваряме поток) и подаваме на конструктора аргументът име на файла, който вземаме като стойност от параметъра. Изписваме на конзолата съобщение, че започва да се пише по указания файл и в него се вкарват всички обекти, взети от втория параметър(дефакто всички подадени като аргумент учители). Потокът се затваря, за да се избегнат “memory-leaks” и изписваме на конзолата, че успешно сме писали по дадения файл. В main метода на програмата са попълнени примерни данни за работните часове на всеки един учител: 

Това се прави с цел да се демонстрира, че сортировката по нарастващ ред е работеща. Тя се извършва СЛЕД изпринтването на стойностите на всеки обект от ArrayList-а, съответно сортировката става преди да качим данните във файла(т.е. там можем да видим сортираните данни, а преди това - несортирани). Използваме статичния клас Collections и неговия метод sort(), който приема нашия ArrayList от учители и съответно от този метод, се извиква логиката, която бяхме написали в класа Teacher, имплементирайки интерфейса Comparable:

Демонстрация и на приемането на входни данни от конзолата:





Това е съответно и резултатът, който виждаме във създадения от нас файл, както можем да видим данните вътре са сортирани в нарастващ ред:

