PAC 3

Associacions

Universitat Oberta de Catalunya

Informació rellevant:

- Data límit de lliurament: 27/11/2023.
- Pes a la nota d'AC: 30%.

Contingut

Informació docent

Aquesta PAC està vinculada al mòdul teòric “Associacions (relacions entre objectes)” dels apunts de l'assignatura. Llegeix-ho abans de començar la PAC.

Prerequisits

Per fer aquesta PAC necessites:

- Tenir adquirits els coneixements de les PAC 1 i PAC 2. Per això et recomanem que miris les solucions que es van publicar a l'aula i les comparis amb les teves.

Objectius

Amb aquesta PAC l'equip docent de l'assignatura cerca que:

- Entenguis les diferències entre associació binària, d'agregació i de composició.
- Codifiquis associacions a Java.
- Entenguis què és una enumeració i el potencial que tenen a Java.
- Utilitzis una llibreria proporcionada per tercers.

Resultats d'aprenentatge

Amb aquesta PAC has de demostrar que ets capaç de:

- Codificar una classe a partir d'uns requisits i la seva representació com a diagrama de classes.
- Entendre la diferència entre els diferents modificadors d’accés.
- Saber com codificar una associació de composició amb Java (i en qualsevol llenguatge).
- Saber com codificar una associació d’agregació amb Java.
- Codificar un `enumeració` a Java afegint-hi funcionalitats.
- Saber com tractar les excepcions d’un programa Java.
- Usar amb certa soltesa un entorn de desenvolupament integrat (IDE) com IntelliJ.
- Utilitzar tests unitaris per determinar que un programa és correcte.
- Saber utilitzar Gradle per afegir llibreries de tercers.

Enunciat

Aquesta PAC conté 4 exercicis avaluables. Has de lliurar la teva solució dels 4 exercicis (veure el darrer apartat).

Com que les activitats estan encadenades (i.e. per fer-ne una s'ha d'haver comprès l'anterior), **és molt recomanable fer els exercicis en l'ordre en què apareixen en aquest enunciat.**

Exercici 1 (2 punts)

Obre el projecte `PAC3Ex1` amb IntelliJ. Al *package* `edu.uoc.pac3` del directori `/src/test/java` hi ha el fitxer amb els tests unitaris que et proporcionem. Al *package* `edu.uoc.pac3` del directori`/src/main/java` has de codificar les classes `Passenger` i `Passport`, la representació de les quals en diagrama de classes UML la tens disponible a la imatge `PAC3Ex1.png` que t'adjuntem amb el fitxer .zip d'aquest enunciat.

Com pots veure al diagrama UML, modelarem el comportament d'un passatger (`Passenger`) i la seva relació amb el seu passaport (`Passport`). La classe `Passport` abstreu la informació relacionada amb les dades del passaport del passatger (número de passaport, data d'expiració…). **La relació entre les classes `Passanger` i `Passport` és una associació unidireccional de composició**, amb la classe `Passenger` com a classe composta i la classe `Passport` com a component, de manera que si desapareix l'objecte `Passanger`, desaparegui l'objecte `Passport` que estava vinculat.

Per a la codificació d'aquestes classes has de tenir en compte les especificacions/consideracions següents:

Classe `Passport`

- Els atributs de tipus "constant" han de ser inicialitzats a la pròpia declaració, no al constructor.
- Els atributs declarats en majúscules en un diagrama de classes UML, p.ex. `PASSPORT_NUMBER_ERROR`, són declarats com a `final`. Podeu llegir sobre què significa i com s'aplica aquest modificador a un atribut a l'apartat 3.8.3 de la Guia de Java. Parlarem més detingudament sobre el modificador `final` a la PAC 4.
- **Sempre que hi hagi conflicte de noms**, sobretot en els mètodes *setters* (és a dir, `setXXX`), **has d'utilitzar la paraula reservada `this`**.

- Si el valor que es vol assignar a l'atribut `passportNumber` és `null` o buit, aleshores no s'ha d'assignar aquest valor i, en el seu lloc, s'ha de llançar una excepció de tipus `IllegalArgumentException` (és una excepció de tipus *unchecked*, consulta l'apartat 3.12.5 de la Guia de Java) amb el missatge indicat a `PASSPORT_NUMBER_ERROR`.

- Si el valor que es vol assignar a l'atribut `issueDate` és `null`, o és més de 10 anys anterior a la data d'avui (el temps d'expiració habitual d'un passaport són 10 anys), o és posterior a la data d'avui (no es pot expedir un passaport en el futur) aleshores no s'ha d'assignar aquest valor i, en el seu lloc, s'ha de llançar una excepció de tipus `IllegalArgumentException` amb el missatge indicat a `ISSUE_DATE_ERROR`.

- Si el valor que es vol assignar a l'atribut `expirationDate` és `null`, o és més de 10 anys posterior a la data d'`issueDate` (i.e. passaport caducat: el temps d'expiració habitual d'un passaport son 10 anys), o anterior a `issueDate`, llavors no s'ha d'assignar aquest valor i, en el seu lloc, s'ha de llançar una excepció de tipus `IllegalArgumentException` amb el missatge indicat a `EXPIRATION_DATE_ERROR`.

- Finalment, si el valor a assignar `visaType` és negatiu, s'ha de llançar una excepció de tipus `IllegalArgumentException` amb el missatge indicat a `VISA_TYPE_ERROR`.

Classe `Passenger`

- Els atributs de tipus "constant" han de ser inicialitzats a la pròpia declaració, no al constructor.

- Els atributs declarats en majúscules en un diagrama de classes UML, p.ex. `NAME_MAX_LENGTH`, són `final`. Podeu llegir sobre què significa i com s'aplica aquest modificador a un atribut a l'apartat 3.8.3 de la Guia de Java. Parlarem més detingudament sobre el modificador `final` a la PAC 4.

- En crear un objecte de tipus `Passenger` cal crear l'objecte `Passport` que li correspongui (pot ser `null`). Com que tenir passaport no és obligatori, la clase `Passenger` ofereix també un constructor que permet crear el passatger sense el passaport. Cap dels 2 constructors ha de llançar una excepció si alguna dada del passaport és incorrecta, en aquest cas el passaport ha de ser `null`.

- El mètode `setPassaport` serà l'encarregat de crear el nou objecte `Passport` amb els valors rebuts com a paràmetres.

- Aquells constructors que no tinguin paràmetres per a tots els atributs han d'assignar els valors per defecte definits al diagrama de classes.

- En el cas del primer constructor, l'objecte `Passenger` creat tindrà un `Passport` amb valor `null`.

- El mètode `setName` ha de comprovar si el nou nom no és `null`, no està buit ni excedeix el nombre màxim de caràcters definit a `NAME_MAX_LENGTH`. Si compleix els requeriments, aquest mètode assignarà el valor corresponent. Si es dóna algun cas d'error, s'ha de llançar una `IllegalArgumentException` amb el missatge indicat a `INVALID_NAME`.

- El mètode `setBirthday` assigna el valor passat com a paràmetre, amb les restriccions que la data no pot ser `null`, ni posterior al dia en curs, ni pot ser anterior a 110 anys. Si es dóna algun cas d'error, s'ha de llançar una `IllegalArgumentException` amb el missatge indicat a `INVALID_BIRTHDAY`.

- El mètode `setAddress` assigna el valor de l'adreça. En aquest cas, els únics requisits que s'han de complir són que la nova adreça no sigui `null` ni tampoc buida. Si es dóna algun cas d'error, s'ha de llançar una `IllegalArgumentException` amb el missatge indicat a `INVALID_ADDRESS`.

- El mètode `setPhoneNumber` ha d'assignar el valor del telèfon. En aquest cas, el `String` del telèfon no podrà ser `null` i haurà de complir el format internacional, fent servir un guió (-) per a separar el codi del país (que sempre comença amb el símbol `+`) del telèfon. Per exemple, per a Espanya, un telèfon vàlid seria +34-675849345 (i.e. `codi-telèfon`). El format no accepta espais en blanc de cap mena ni entre els números ni abans ni després. El codi del país tindrà entre 1 i 3 dígits, mentre que el número que va després del codi de país, hi podrà tenir 12 números com a màxim i 1 com a mínim. Si el `String` passat per paràmetre no compleix amb el format indicat, s'ha de llançar una `IllegalArgumentException` amb el missatge indicat a `INVALID_PHONE_NUMBER_FORMAT`.

Pista: en PACs anteriors ja has vist com fer servir *regex* per a validar cadenes de text en determinats formats.

- El mètode `setNationality` assigna la nacionalitat. En aquest cas, els únics requisits que s'han de complir són que la nova nacionalitat no sigui `null` ni buida. Si hi ha un error, s'ha de llançar una `IllegalArgumentException` amb el missatge indicat a `INVALID_NATIONALITY`.

- El mètode `setHeight` abans d'assignar el nou valor, ha de comprovar que el valor rebut per paràmetre no sigui inferior a `50` ni major a `250`. En cas què es doni aquesta situació, ha de llançar una `IllegalArgumentException` amb el missatge indicat a `INVALID_HEIGHT`.

- El mètode `hasSpecialNeeds` ens permet indicar si el passatger té necessitats especials. En aquest cas, no hi ha requisits a complir, només passar el valor booleà desitjat per paràmetre.

- El mètode `setOccupation` assigna el valor de l'ocupació. En aquest cas, els únics requisits que s'han de complir són que l'ocupació no sigui `null` ni tampoc buida. Si es dóna el cas d'error, s'ha de llançar una `IllegalArgumentException` amb el missatge indicat a `INVALID_OCCUPATION`.

Requisit mínim per avaluar aquest exercici: el programa ha de passar els tests de `PassportTest`.

Nota: l'estudiant pot rebre una penalització de fins **0.5 pts.** de la nota obtinguda en aquest exercici en funció de la qualitat del codi proporcionat.

(2 punts: 1 pt. PassportTest; 1 pt. PassengerTest)

Exercici 2 (5 punts)

Obre el projecte `PAC3Ex2` des d'IntelliJ. Al *package* `edu.uoc.pac3` del directori `/src/test/java` hi ha el fitxer amb els tests unitaris que et proporcionem. Copia el *package* `edu.uoc.pac3` del directori`/src/main/java` de l'Exercici 1 d'aquesta PAC3 un cop l'hagis completat.

En aquest exercici ampliem el programa anterior afegint-hi una nova classe, concretament la classe `Flight`, la qual modela/representa viatges d'avió. El nou diagrama de classes UML el pots trobar a la imatge `PAC3Ex2.png` que t'adjuntem amb el fitxer .zip d'aquest enunciat:

Tingues present les següents especificacions per a la classe `Flight`:

Important: per realitzar aquest exercici només es poden fer servir *arrays*. No es poden fer servir col·leccions de Java (p.ex. `ArrayList`) ni la classe `Stream`, inclòs mètodes com `Arrays.stream` i `Stream.of`.

Classe `Flight`

- Els atributs de tipus "constant" han de ser inicialitzats a la pròpia declaració, no al constructor.
- A l'atribut `id` se li ha d'assignar aquell valor que tingui l'atribut `nextId` en el moment d'instanciar l'objecte. Un cop fet, el valor de `nextId` s'ha d'incrementar una unitat. Tot aquest procés només s'ha de fer si no hi ha cap problema amb la resta de paràmetres rebuts en el constructor.
- El mètode `incNextId` incrementa una unitat `nextId`.
- El constructor ha de llançar una `IllegalArgumentException` amb el missatge de l'atribut `ERROR_DATES` si es dóna alguna de les següents circumstàncies: (1) si la data de sortida (`departureDate`) és `null`, (2) la data d'arribada (`arrivalDate`) és `null`, o (3) la data de sortida és posterior a la d'arriba.
- Un vol podrà tenir, com a màxim, el número de passatgers indicat com argument a través del constructor.
- El mètode `setOrigin` llançarà una `IllegalArgumentException` amb el missatge d'`ERROR_ORIGIN` si el valor passat com a paràmetre és `null` o buit. En cas contrari, assignarà el nou valor a l'atribut `origin`.

- El mètode `setDestination` llançarà una `IllegalArgumentException` amb el missatge d'`ERROR_DESTINATION` si el valor passat com a paràmetre és `null` o buit. En cas contrari, assignarà el nou valor a l'atribut `destination`.

- El mètode `setDepartureDate` només assignarà el nou valor si aquest és `null`, o anterior a la data d'arribada o la data d'arribada és `null`. En cas contrari, llançarà una `IllegalArgumentException` amb el missatge d'`ERROR_DATES`.

- El mètode `setArrivalDate` només assignarà el nou valor si aquest és `null`, o posterior a la data de sortida o la data de sortida és `null`. En cas contrari, ñlançarà una `IllegalArgumentException` amb el missatge d'`ERROR_DATES`.

- El mètode `getDuration` retorna el número d'hores que dura el vol, és a dir, el temps que transcorregut entre la data de sortida i la d'arribada.

Nota: investiga a Internet sobre la classe `Duration` per obtenir el temps entre dues dates. Pensa en termes de factor de conversió i tingues present els problemes de pèrdua de precisió dels tipus numèrics quan es fan certes operacions i les possibles solucions per evitar-ho. Revisa la Guia de Java.

- El mètode `findPassenger` retorna l'índex/posició on es troba el passatger passat com a paràmetre. Si el passatger a cercar no es troba al vol, llavors ha de retornar `-1`.

  Si aquest mètode rep com a paràmetre el valor `null`, llavors ha de llançar una `NullPointerException` amb el missatge `ERROR_NULL`. Si el passatger passat com a paràmetre no té passaport, llavors ha de llançar una `NullPointerException` amb el missatge `ERROR_NO_PASSPORT`.

Nota: l'atribut que permet identificar unívocament un passatger és el seu número de passaport.

- El mètode `containsPassenger` retorna `true` si el passatger passat com a paràmetre existeix en el vol. En cas contrari, retorna `false`.

  Si aquest mètode rep com a paràmetre el valor `null`, llavors ha de llançar una `NullPointerException` amb el missatge `ERROR_NULL`. Si el passatger passat com a paràmetre no té passaport, llavors ha de llançar una `NullPointerException` amb el missatge `ERROR_NO_PASSPORT`.

- El mètode `addPassenger` afegeix un passatger al vol. Tingues en compte:

- Si el passatger ja existeix, aleshores ha de llançar una `IllegalStateException` amb el missatge indicat a `ERROR_PASSENGER_ALREADY_IN_FLIGHT`.
- Si aquest mètode rep com a paràmetre el valor `null`, llavors ha de llançar una `NullPointerException` amb el missatge `ERROR_NULL`.
- Si el passatger passat com a paràmetre no té passaport, llavors ha de llançar una `NullPointerException` amb el missatge `ERROR_NO_PASSPORT`.
- El passatger es desarà a la primera posició lliure que hi hagi a l'*array* `passengers`. Una posició està lliure si el seu valor és `null`.
- Si el passatger s'ha afegit al vol correctament, aquest mètode retornarà `true`. En cas contrari (p.ex. el vol està ple), retornarà `false`.

- El mètode `removePassenger` elimina del vol el passatger passat com a argument deixant la seva posició lliure. Si el passatger s'ha esborrat del vol correctament, aquest mètode retornarà `true`. En cas contrari (p.ex. el passatger no hi és al vol), retornarà `false`.

  Si aquest mètode rep com a paràmetre el valor `null`, llavors ha de llançar una `NullPointerException` amb el missatge `ERROR_NULL`. Si el passatger passat com a paràmetre no té passaport, llavors ha de llançar una `NullPointerException` amb el missatge `ERROR_NO_PASSPORT`.

- El mètode `getNumPassengers` retorna un `int` amb el total de passatgers que hi ha al vol.

Requisit mínim per avaluar aquest exercici: tots els test proporcionats a la classe `FlightTest` han de ser superats amb èxit de manera independent.

IntegrationTest: la classe de test anterior no executa els mètodes implicats en la inserció i eliminació de passatgers. Aquests són comprovats per `IntegrationTest`, que correspon a 3 punts de la nota de l'exercici (0.375 punts per test superat).

Nota: l'estudiant pot rebre una penalització de fins **1 punt** de la nota obtinguda en aquest exercici en funció de la qualitat del codi proporcionat.

(5 punts: 2 pts. FlightTest; 3 pts. Integrationtest)

Exercici 3 (1.5 punts)

Abans de començar has de:

> Llegir l'apartat 3.13 de la Guia de Java que parla de les enumeracions.
>
> Et recomanem veure el vídeo sobre enumeracions que trobaràs a UOCoders.

A continuació, **obre el projecte `PAC3Ex3` des d'IntelliJ**. Al *package* `edu.uoc.pac3` del directori `/src/test/java` hi ha el fitxer amb els tests unitaris que et proporcionem. Crea el *package* `edu.uoc.pac3` dins del directori`/src/main/java` i copia en ell només el fitxer `Passport.java` de l'Exercici 1 d'aquesta PAC3 un cop l'hagis completat.

En aquest exercici has de fer una petites modificacions a la classe `Passport` i codificar l'`enum VisaType` seguint les especificacions del diagrama de classes UML que trobaràs a la imatge `PAC3Ex3.png` adjuntada dins del fitxer .zip d'aquest enunciat. Fixa't com es defineixen els `enums` a UML. Hi ha una altra notació que consisteix a posar l'atribut de tipus `enum` dins de la classe que l'utilitza indicant el tipus, igual que si fos un atribut de tipus, per exemple, `int`. En aquest cas no hi hauria fletxa cap a l'`enum`.

Enum `VisaType`

`VisaType` modela, com pots suposar, el tipus de visat per a un passaport determinat. Per això, has de tenir en compte:

- `VisaType` té un atribut `description` que emmagatzema una descripció breu del visat, un atribut `stringIdentifier` que emmagatzema una representació curta del visat en format `String`, i un atribut `intIdentifier` (de tipus `int`) que assigna un identificador numèric al visat. La relació "visat-descripció-stringIdentifier-intIdentifier" és la següent:

```
TOURIST             | Tourist Visa             | 'T'         | 123
BUSINESS            | Business Visa            | 'B'         | 456
STUDENT             | Student Visa             | 'S'         | 789
WORK                | Work Visa                | 'W'         | 321
TRANSIT             | Transit Visa             | 'TR'        | 654
FAMILY              | Family Visa              | 'F'         | 987
JOURNALIST          | Journalist Visa          | 'J'         | 234
MEDICAL             | Medical Visa             | 'M'         | 567
RETIREMENT          | Retirement Visa          | 'R'         | 890
INVESTOR            | Investor Visa            | 'I'         | 432
DIPLOMATIC          | Diplomatic Visa          | 'D'         | 765
SCHENGEN            | Schengen Visa            | 'SCH'       | 198
EMPLOYMENT          | Employment Visa          | 'E'         | 543
VISITING_FRIENDS    | Visiting Friends Visa    | 'VF'        | 876
RELIGIOUS           | Religious Visa           | 'RL'        | 321
OTHER               | Other Visa               | 'O'         | 654
```

Nota: codifica els valors de l'enum en l'ordre en què apareixen al llistat anterior.

- `VisaType`: és el constructor, el qual ha de desar els valors dels paràmetres `description`, `stringIdentifier` i `intIdentifier`.
- `getDescription`, `getStringIdentifier` i `getIntIdentifier`: quan són invocats utilitzant un valor de l'`enum`, retornen el seu valor de `description`, `stringIdentifier` i `intIdentifier`, respectivament.
- `getVisaTypeByStringIdentifier`: aquest mètode rep un `String` que representa la codificació d'un visat, i retorna quin dels valors de l'enumeració correspon a aquesta codificació. És a dir, retorna el `VisaType` corresponent, o `null` si el valor de la codificació passat com a argument no correspon amb cap valor de l'`enum`.
- `next`: aquest mètode retorna el valor de l'enum `VisaType` posterior al valor amb el que s'invoca aquest mètode. Així, si fem `VisaType.TOURIST.next()`, el resultat ha de ser `BUSINESS`. Si el valor amb el que invoquem `next` és l'últim de l'enumeració, llavors ha de retornar el primer valor, i.e. `VisaType.OTHER.next()` → `TOURIST`.

(1.5 punts: 0.5 punts `testVisaTypeDescriptions`***,*** `testVisaTypeStringIdentifier`***,*** `testVisaTypeIntIdentifiers`***;*** ***0.5 punts*** `testGetVisaTypeByStringIdentifier`***; 0.5 punts*** `next`***)***

Classe `Passenger`

Has d'adaptar aquesta classe perquè l'atribut `visaType` ja no sigui de tipus `int`, sinó `VisaType`. El missatge de `VISA_TYPE_ERROR` ara ha de ser `"Visa type must be a correct value"` i s'ha de llançar si el paràmetre de `setVisaType` és `null`.

Requisit mínim per avaluar aquest exercici: els test `PassportTest`, `testVisaTypeDescriptions`, `testVisaTypeStringIdentifiers`, `testVisaTypeIntIdentifiers` i `testGetVisaTypeByStringIdentifier` han de ser superats.

Nota: l'estudiant pot rebre una penalització de fins **0.5 pts.** de la nota obtinguda en aquest exercici en funció de la qualitat del codi proporcionat.

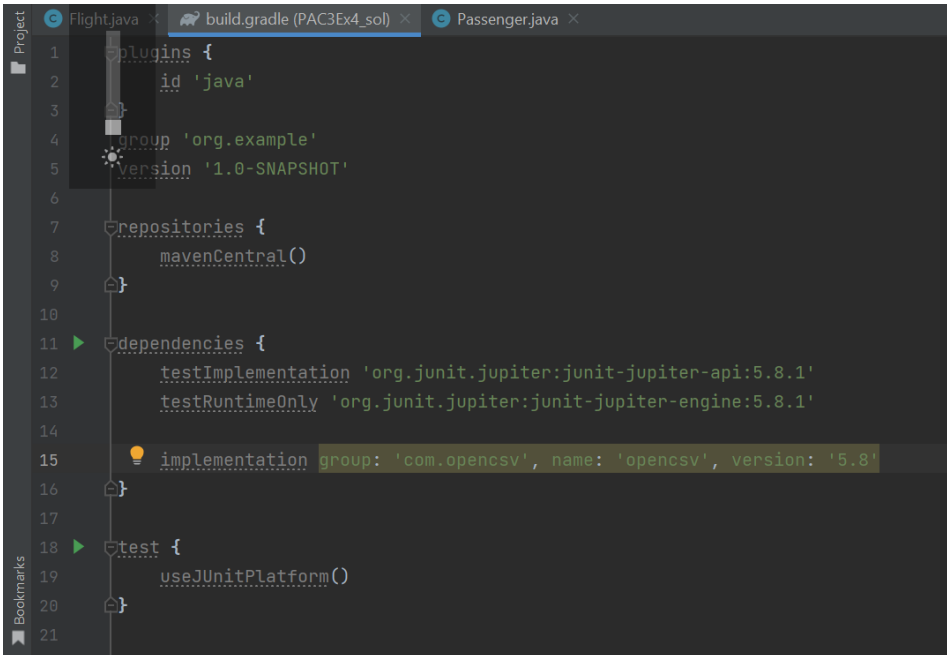
Exercici 4 (1.5 punts)

Obre el projecte `PAC3Ex4` des d'IntelliJ. Al *package* `edu.uoc.pac3` del directori `/src/test/java` hi ha el fitxer amb els tests unitaris que et proporcionem. Al directori `/src/main/java` us proporcionem la implementació d'una versió reduïda de la classe `Passenger` i una versió reduïda de la classe `Flight` (ambdues versions seran les que utilitzaràs per fer aquest exercici).

En aquest exercici utilitzarem la llibreria `OpenCSV`. Aquesta llibreria encapsula un conjunt de funcionalitats per a la manipulació de fitxers CSV. Per utilitzar `OpenCSV` el primer que hem de fer és afegir la llibreria al nostre projecte. Atès que estem treballant amb Gradle, només haurem d'afegir la següent línia de codi al bloc `dependencies` del fitxer `build.gradle`:

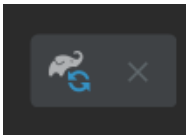
```
implementation group: 'com.opencsv', name: 'opencsv', version: '5.8'
```

De manera que el fitxer `build.gradle` tindrà ara el següent aspecte:

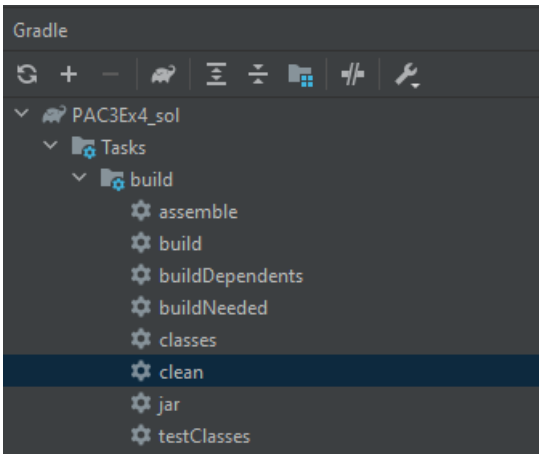


Aquest línia de codi a afegir a `build.gradle` l'hem trobada a mvnrepository.com, el cercador de llibreries Java. Si es busca "OpenCSV", apareixeran diversos resultats. Escollim el primer i després fem clic a la versió 5.8. Així arribem a: https://mvnrepository.com/artifact/com.opencsv/opencsv/5.8, on està la línia de codi anterior.

Un cop afegida, fem clic en el botó flotant que haurà aparegut a dalt a la dreta amb el símbol d'un elefant:



D'aquesta manera es descarregarà la llibreria `OpenCSV` dins del projecte. Si tens problemes, et recomanem executar les tasques *clean* i *build* de Gradle per assegurar que les noves llibreries són afegides correctament com a dependències del projecte. Per això, has d'obrir el panell lateral dret de *Gradle* i executar les tasques `PAC3Ex4 > Tasks > build > clean` i `PAC3Ex4 > Tasks > build > build`.



Ara ja estem preparats per utilitzar la llibreria `OpenCSV`.

Pista: per realitzar aquest exercici, investiga com utilitzar la llibreria `OpenCSV` i els mètodes que ofereix. Pots començar fent una ullada a la següent documentació:

https://opencsv.sourceforge.net/#reading

En aquest exercici et demanem que codifiquis el mètode `populate` de la classe `Flight`. Aquest mètode rep un fitxer (classe `File`) que conté, a cada fila, informació d'un passatger que pertany a aquell vol. Concretament:

```
name, passaportNumber, age, specialNeeds
```

El mètode `populate` el que ha de fer és llegir el fitxer passat per paràmetres i inserir cada passatger a l'*array* `passengers`. Haurà d'actualitzar el número de passatgers (atribut `numPassengers`) que hi ha al vol.

Requisit mínim per avaluar aquest exercici: el programa ha de passar tots els test proporcionats a la classe `FlightTest`.

Format i data de lliurament

Has de lliurar un fitxer *.zip, el nom del qual ha de seguir aquest patró: `loginUOC_PAC3.zip`. Per exemple: `dgarciaso_PAC3.zip`. Aquest fitxer comprimit ha d'incloure els elements següents:

- El projecte IntelliJ `PAC3Ex1` completat seguint les peticions i les especificacions de l'Exercici 1.
- El projecte IntelliJ `PAC3Ex2` completat seguint les peticions i les especificacions de l'Exercici 2.
- El projecte IntelliJ `PAC3Ex3` completat seguint les peticions i les especificacions de l'Exercici 3.
- El projecte IntelliJ `PAC3Ex4` completat seguint les peticions i les especificacions de l'Exercici 4.

El darrer dia per lliurar aquesta PAC és el dia **27/11/2023** abans de les 23:59. Qualsevol PAC lliurada més tard serà considerada com a no presentada.