Pràctica 2

1. Descripció del Problema

L'objectiu d'aquesta pràctica es desenvolupar un sistema que permeti modelar, consultar i actualitzar informació sobre l'**estació de tren *Green***. A la Figura 1 es pot veure un mapa esquemàtic de l'estació.

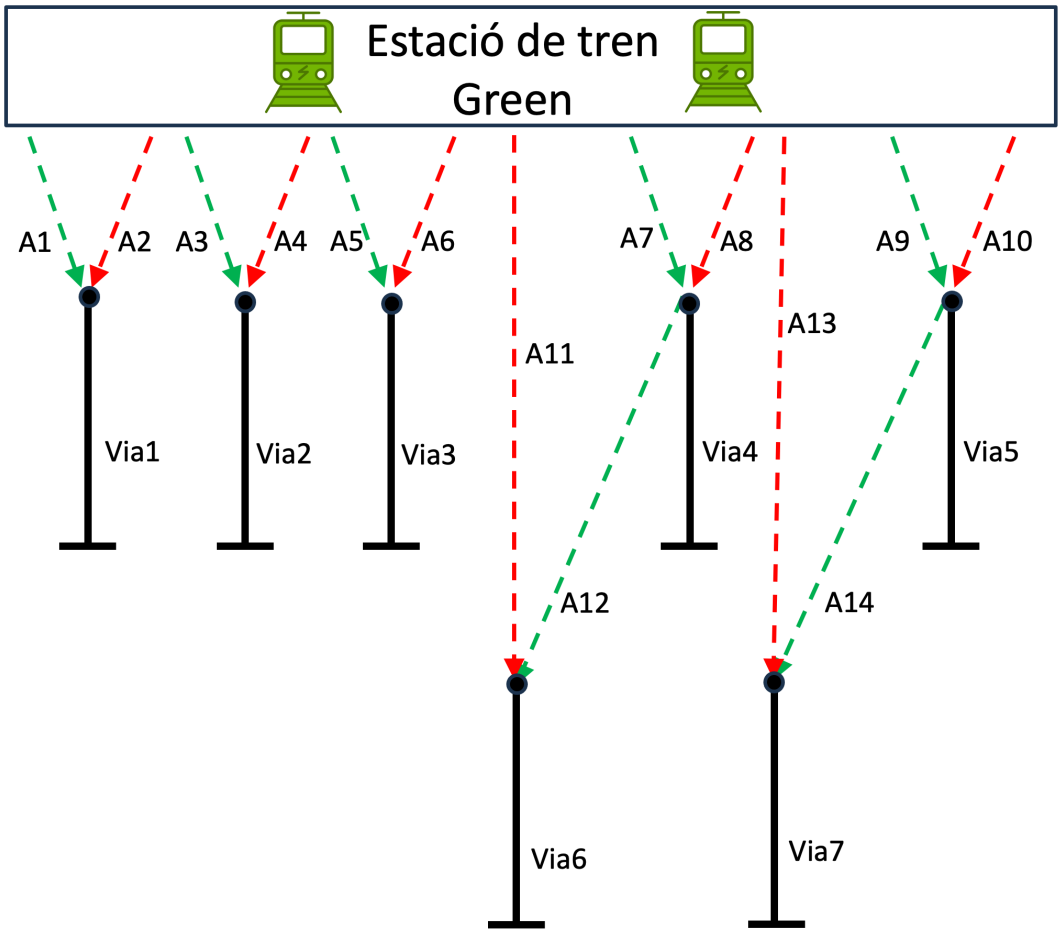

Figura 1. Mapa esquemàtic de l'estació de tren *Green*. Les vies estan representades amb una línia contínua negra i els accessos estan representats per una línia discontínua verda (amb accessibilitat) o vermella (sense accessibilitat).

Les vies de tren. L'estació té 7 vies de tren amb les següents propietats: un nom, una amplada, un número de túnels fins la sortida de l'estació, un estat de la via (que pot set Oberta o Tancada) i un estat de la il·luminació (que pot ser 100%, 50% o 0%). Els detalls de cada via, així com l'estat actual de les mateixes, es pot trobar a la Taula 1.

Nom	Amplada via	Número de túnels	Estat	Estat de la il·luminació
Via1	Ibérica	2	Oberta	100%
Via2	Ibérica	2	Oberta	100%
Via3	Internacional	2	Oberta	100%
Via4	Ibérica	1	Oberta	100%
Via5	Ibérica	1	Oberta	100%
Via6	Estreta	1	Oberta	100%
Via7	Estreta	1	Oberta	100%

Taula 1: Informació detallada sobre les vies de tren de l'estació *Green* (estats a l'inici).

Els accessos. L'estació disposa de 14 accessos que permeten als viatgers desplaçar-se per arribar a les vies de tren (tal com es pot veure a la Figura 1). Cada accés té un nom i guarda la informació de si està obert o no i la llista de les vies a les que permet l'accés.

Per tal de facilitar el trànsit per l'estació als viatgers amb mobilitat reduïda hi ha accessos que fomenten l'accessibilitat a les vies. En concret, a l'estació hi ha 5 tipus d'accessos amb diferent accessibilitat (Acc): passadís (Acc Sí), escala (Acc No), escala mecànica (Acc No), cinta transportadora (Acc No) i ascensor (Acc Sí).

Els accessos es divideixen en accessos a nivell (passadís i cinta transportadora) que guarden els metres de longitud, i els accessos amb desnivell (ascensor, escala i escala mecànica) que guarden els metres de desnivell.

A més, cal que es guardi certa informació sobre les característiques de 3 dels accessos: pes màxim de càrrega de l'ascensor, marca de l'escala mecànica i velocitat de la cinta transportadora. Els detalls dels accessos de l'estació es poden trobar a la Taula 2.

Nom	Tipus	Característiques	Accessibilitat	Estat	Accés a
A1	Ascensor	Desnivell 9 metres i pes màxim de càrrega 200 Kg.	Sí	Oberta	Via 1
A2	Escala mecànica	Desnivell 9 metres i marca Schindler.	No	Oberta	Via 1
A3	Passadís	Longitud 30 metres.	Sí	Oberta	Via 2
A4	Cinta transportadora	Longitud 30 metres Velocitat 0,3 metres/segon.	No	Oberta	Via 2
A5	Ascensor	Desnivell 7 metres i pes màxim de càrrega 300 Kg.	Sí	Oberta	Via 3
A6	Escala	Desnivell 7 metres.	No	Oberta	Via 3
A7	Ascensor	Desnivell 8,5 metres i pes màxim de càrrega 250 Kg.	Sí	Oberta	Via 4; Via 6
A8	Escala	Desnivell 8,5 metres.	No	Oberta	Via 4; Via 6
A9	Passadís	Longitud 25 metres	Sí	Oberta	Via 5; Via 7
A10	Cinta transportadora	Longitud 25 metres i velocitat 0,4 metres/segon	No	Oberta	Via 5; Via 7
A11	Escala mecànica	Desnivell 8 metres i marca Otis.	No	Oberta	Via 6
A12	Passadís	Longitud 20 metres	Sí	Oberta	Via 6
A13	Escala	Desnivell 20 metres.	No	Oberta	Via 7
A14	Ascensor	Desnivell 20 metres i pes màxim de càrrega 350 Kg.	Sí	Oberta	Via 7

Taula 2: Informació detallada sobre els accessos existents a l'estació de tren (estats a l'inici).

Les incidències. D'altra banda, es vol gestionar les incidències que es produeixen a les vies. Hi ha 3 tipus d'incidència: *reparació* (la via està en reparació), *objecte* (hi ha un objecte que impedeix la circulació per la via) i *tancament* (la via està tancada per una altra causa). Cada incidència té un número únic que serveix per identificar-la i a més guarda la via on s'ha produït, el tipus d'incidènica i la data.

La gestió automàtica de la il·luminació. Com que es tracta d'una estació de tren "verda" es vol minimitzar l'ús energètic de l'estació mitjançant l'optimització dels recursos d'il·luminació. Per això, l'estat de la il·luminació de les vies i els seus accessos canviarà depenent de les incidències produïdes a les vies.

Per tal de gestionar de forma eficaç la il·luminació de l'estació, s'ha desenvolupat un sistema automàtic. Aquest sistema entra en acció cada vegada que es s'afegeix o s'elimina una incidència en alguna de les vies, i controla que les vies i els accessos actualitzin automàticament el seu estat en funció de la nova incidència. S'entén que **una via no pot tenir més d'una incidència**. Llavors, a l'hora de crear i afegir una incidència es verificarrà que no existeixi ja una incidència sobre aquesta via.

Quan s'afegeix una incidència d'una via, **s'haurà de tancar la via i el tipus d'incidència** (que pot ser: reparació, objecte, tancament) **determinarà l'estat de la seva il·luminació** (que es deixarà al: 100%, 50% o 0%, respectivament per cada tipus d'incidència).

A més, quan s'afegeix una incidència d'una via, també **s'han de tancar els accessos que permeten arribar-hi a aquesta via**. Per fer-ho s'haurà de consultar la llista de vies dels accessos (última columna de la Taula 2 anomenada *Accés a*). S'ha de tenir en compte que no s'ha de tancar un accés si aquest permet accedir a una altra via a més de la via on s'ha produït la incidència. Per exemple, si s'afegeix una incidència de la via 4, l'accés A7 no s'ha de tancar, perquè també permet l'accés a la via 6. En canvi, una incidència a la via 6 tancaria els accessos A11 i A12

S'entén que quan un accés està tancat la seva il·luminació està apagada (i per tant s'aconsegueix l'estalvi desitjat) i quan està obert la seva il·luminació està oberta.

A l'inici, a l'estació no hi ha cap incidència i totes les vies i accessos estan oberts i il·luminats com es pot veure a la Taula 1 i Taula 2.

2. Objectiu i Funcionalitats

L'objectiu d'aquesta pràctica és modelar la informació corresponent a l'estació de tren *Green* utilitzant el paradigma de programació orientat a objectes. L'aplicació desenvolupada haurà d'oferir el següent conjunt de funcionalitats:

1. *Llistar la informació de totes les vies*

2. *Llistar la informació de les vies obertes*

3. *Llistar la informació de les vies tancades*

4. *Llistar la informació dels accessos oberts*

5. *Llistar la informació dels accessos tancats*

6. *Llistar la informació de les incidències actuals*

7. *Afegir una incidència*

   - L'aplicació haurà de demanar a l'usuari que doni el número de la incidència, el nom de la via on s'ha produït la incidència, el tipus d'incidència que s'ha produït i la data de la incidència. A més, aquesta operació pot produir errors que s'hauran de gestionar.

8. *Eliminar una incidència*

   - L'aplicació haurà de llistar les incidències que hi ha actualment a la llista d'incidències i demanar a l'usuari que doni el número de la incidència que es vol eliminar.

9. *Calcular i mostrar el número total d'accessos que proporcionen accessibilitat*

10. *Calcular i mostrar el número total de metres de longitud d'accessos de nivell.*

    - És a dir, la suma de metres de longitud que tenen els accessos de nivell (passadís i cintes transportadores).

11. *Guardar estació*

    - Guarda les dades de l'estació de tren en un fitxer.

12. *Recuperar estació*

    - Carrega d'un fitxer les dades de l'estació prèviament guardades.

13. *Sortir de l'aplicació*

Com a la pràctica 1, organitzareu el codi en dos paquets [paquet Vista] (**prog2.vista**) o [paquet Model] (**prog2.model**). Les classes de la vista s'encarregaran de la interacció amb l'usuari i les classes del model contindran totes les dades que s'han de manipular i faran totes les accions que afectin a les dades. Des de la classe principal de la vista es demanaran els valors dels atributs necessaris per passar-los a les classes del model. Tingueu en compte que **tots els prints s'hauran de fer a la vista, mitjançant el tractament d'excepcions**, i no hi haurà cap print a cap classe del model.

3. Material pel lliurament

Per aquesta pràctica us proporcionem un codi base que conté les classes i interfícies necessàries per la pràctica:

- ***IniciadorEstacioTren***
- ***Menu***
- ***ExempleMenu (Per veure com s'ha d'utilitzar la classe Menu)***
- ***InAcces***
- ***InVia***
- ***InLListaVies***
- ***InLlistaAcessos***
- ***InLlistaIncidencies***

Trobareu aquest codi al Campus Virtual junt amb aquest enunciat en la secció de PRÀCTIQUES: "Pràctica 2 (enunciat i codi base)". L'heu de descarregar i afegir-lo al vostre projecte.

4. Descripció del Codi Base

Al Campus Virtual de l'assignatura podràs trobar les classes i interfícies que et serviran de base. Un cop hagis creat el teu projecte NetBeans (veure secció **4.1**) has d'afegir aquestes classes i interfícies per poder-les fer servir

4.1 Creació del Projecte Base amb NetBeans

El primer pas serà crear un projecte al NetBeans (en particular s'ha de fer servir la categoria "**Java with Ant**"), al qual li heu de posar com a nom Cognom1Nom1Cognom2Nom2, on 1 i 2 fa referència als dos membres de la parella, i tenint en compte les següents consideracions:

- La primera lletra de cada part en majúscula i la resta en minúscula.
- Eviteu utilitzar accents i caràcters del tipus ñ o ç.

  Per exemple, una estudiant amb nom Dolça Martínez Castaña, hauria de crear un projecte amb el nom MartinezCastanaDolca.
- La classe principal s'ha de dir ***IniciadorEstacioTren***, i ha d'estar en el paquet **prog2.vista**.

4.2 Codi Base: Classe *IniciadorEstacioTren*

Aquesta classe té com a responsabilitat llançar el bucle principal de l'aplicació. La classe de la vista ***IniciadorEstacioTren*** té un mètode estàtic ***main()*** on es crea un objecte de tipus ***VistaEstacioTren*** anomenat *vistaEstacioTren*. Després crida el mètode ***gestioEstacioTren()*** de l'objecte *vistaEstacioTren*, on es troba el bucle principal de l'aplicació.

4.3 Codi Base: Classe *Menu* i *ExempleMenu*

Al codi base disposeu de la classe ***Menu*** i un exemple de com s'utilitza la classe ***Menu*** a ***ExempleMenu***. La classe ***Menu*** gestiona un menú genèric, permetent mostrar la llista d'opcions i controlar la selecció d'una opció per part de l'usuari. Podeu executar la classe ***ExempleMenu*** per veure l'output i veure com s'utilitza la classe ***Menu***.

4.4 Codi Base: Interficies *InAcces, InVia, InLListaVies, InLlistaAcessos, InLlistaIncidencies*

Les interfícies ***InAcces***, ***InVia***, ***InLListaVies***, ***InLlistaAcessos*** i ***InLlistaIncidencies*** donaran forma a algunes de les classes que s'han de desenvolupar durant el projecte.

5. Desenvolupament del Projecte

Les classes necessàries per a desenvolupar el sistema de gestió de l'estació de tren han de formar part del paquet **prog2.model**.

En tot moment tingueu en compte que es valorarà la modularitat del vostre codi, la reutilització i el seguiment de les bones pràctiques de programació. Com a regla general, totes les classes del model hauran de tenir un constructor i un conjunt de mètodes *getters* y *setters*.

5.1 Classe *VistaEstacioTren*

Aquesta classe és responsable de la interacció amb l'usuari.

Tal com es pot veure al mètode *main* de la classe ***IniciadorEstacioTren*** on s'instancia l'objecte de la classe ***VistaEstacioTren***, la classe ***VistaEstacioTren*** ha de tenir un mètode *gestioEstacioTren*() que gestionarà el menú principal de l'aplicació. El menú s'implementarà fent servir la classe ***Menu*** del codi base.

La classe ***VistaEstacioTren*** haurà de contenir un atribut privat de tipus ***EstacioTren***, anomenat *estacioTren*, que contindrà tota la informació sobre l'estació. Al constructor de la classe ***VistaEstacioTren*** s'haurà d'instanciar aquest atribut i cridar al mètode *inicialitzaDadesEstacioTren* de la classe ***EstacioTren***.

5.2 Classe *EstacioTren*

La classe ***EstacioTren*** conté tota la informació sobre l'estació. A través d'aquesta classe, la vista (és a dir, la classe ***VistaEstacioTren***) podrà demanar informació sobre l'estació i canviar el seu estat.

Aquesta classe té un mètode *afegirIncidencia* i *eliminarIncidencia* (amb les signatures indicades a continuació). El primer mètode farà el següent: recuperar la via amb el nom donat, invocar el mètode per *afegirIncidencia* de l'objecte de tipus ***LlistaIncidencies*** i per últim actualitzar l'estat de tots els accessos, mitjançant el mètode *actualitzaEstatAccessos* de la classe ***LlistaAccessos***. El segon mètode per eliminar farà el següent: recuperar la incidènica amb el número donat, invocar el mètode per *eliminarIncidencia* de l'objecte de tipus ***LlistaIncidencies*** i per últim actualitzar l'estat de tots els accessos, mitjançant el mètode *actualitzaEstatAccessos* de la classe ***LlistaAccessos***.

```
public void afegirIncidencia(int num, String tipus, String nomVia, String data) throws ExcepcioEstacio;

public void eliminarIncidencia(int num) throws ExcepcioEstacio;
```

La resta de la implementació l'hauràs de fer de forma autònoma, seguint l'enunciat el codi proporcionat i el teu criteri de programador/a.

Inicialització de les dades de l'Estació de Tren *Green*: En el codi base del Campus Virtual pots trobar un mètode *inicialitzaDadesEstacioTren* (dins del fitxer *inicialitzaDadesEstacioTren.txt*) que inicialitza les dades de l'estació amb la informació de les Taules 1 i 2. Cal copiar aquest mètode en la classe ***EstacioTren***. Aquesta funció s'haurà de cridar des del constructor de la vista (***VistaEstacioTren***).

5.3 Classe *Acces, AccessNivell, AccessDesnivell i classes* filles

Per modelar els diferents tipus d'accessos utilitzarem el mecanisme d'herència. S'haurà de crear una classe abstracta anomenada ***Acces***, de la següent manera:

```
public abstract class Acces implements InAcces{

}
```

Aquesta classe contindrà els atributs comuns a tots els accessos, és a dir, nom, accessibilitat, estat (un boolean per indicar obert o tancat), i les vies a les que permet l'accés.

També crearem les classes abstractes ***AccessNivell*** i ***AccessDesnivell*** (subclasse de Access) per guardar la seva informació de la longitud i alçada de desnivell, respectivament.

A continuació s'hauran de crear 2 classes filles d'***AccessNivell*** anomenades ***Passadis*** i ***CintaTransportadora*** i 3 classes filles d'***AccesDesnivell*** anomenades ***Ascensor***, ***Escala***, ***EscalaMecanica***. Aquestes 5 classes implementaran cadascuna el seu mètode isAccessibilitat(), que retorna si l'accés proporciona accessibilitat o no tal com està explicat en la secció 1. Els mètodes *getters* i *setters* així com el mètode *toString()* s'hauran de definir en els accessos que afegeixen atributs (ascensor, escala mecànica i cinta transportadora). Aquest *toString* haurà de cridar al mètode *toString* de la superclasse.

5.4 Classe *Via*

La classe ***Via*** conté tots els atributs d'una via, és a dir, nom, amplada de via, número de túnels, estat de la via i estat de la il·luminació. A part dels *getters* i *setters* i el *toString* els mètodes de la classe ***Via*** estan definits en la interfície ***InVia***.

5.5 Classe *Incidencia*

La classe ***Incidencia*** conté la informació d'una incidència: el número d'identificador, la via, la data (que pot ser un String) i el tipus d'incidència (que es pot definir fent servir el ***enum TipusIncidencia*** que s'ha de definir en la pròpia classe).

```
public static enum TipusIncidencia {
      Reparacio,
      Objecte,
      Tancament
   };
```

A més del mètode constructor, el mètode *toString* i els mètodes *getters* i *setters* dels atributs s'haurà de definir el següent mètode que retorna la informació del percentatge

de la il·luminació a la que s'ha de deixar la via on s'ha produït la incidència segons el tipus de la mateixa incidència:

```
public String getIlluminacioVia();

}
```

5.6 Classes *LlistaVies, LlistaAccessos, LlistaIncidencies*

Crea la classe ***LlistaVies, LListaAccessos i LlistaIncidencies*** que, com el seu nom indica, té com a objectiu guardar i gestionar una llista de vies, accessos i incidències, respectivament. Cal que aquestes classes tinguin un atribut privat de tipus *ArrayList<Via>*, *ArrayList<Acces>* i *ArrayList<Incidenica>*, respectivament, que s'inicialitzarà en el constructor de la classe. A més, la forma d'aquestes classes ve donada en les interfícies ***InLListaVies***, ***InLlistaAcessos*** i ***InLlistaIncidencies***.

Tingueu en compte que una Via no pot tenir més d'una incidència i això s'ha de controlar en el moment que es vol afegir una nova incidència (opció 7 del menú).

Llegiu la descripció dels mètodes que hi ha als comentaris de les interfícies per seguir les indicacions i fer la implementació de les classes. Per exemple, el mètode ***afegirIncidencia*** de ***LListaIncidencies*** haurà de comprovar i avisar si la via ja té una incidència prèvia o si el *tipus* d'incidència que es vol afegir no existeix.

5.7 Persistència de dades

Amb les **opcions 11 i 12 del menú** volem poder guardar les dades (***EstacioTren***) a un fitxer en disc i poder-les carregar posteriorment.

Per fer-ho implementarem els següents mètodes guardar i carregar a la classe ***EstacioTren:***

```
public void guardar(String camiDesti) throws ExcepcioEstacio;

public static EstacioTren carregar(String camiOrigen) throws ExcepcioEstacio;
```

I seguirem els següents passos:

1. Obtenir la ruta al fitxer on voleu guardar les dades o des del qual voleu carregar-les. Necessitareu guardar aquesta informació en un objecte de tipus File. Per exemple, si volem utilitzar el fitxer "EstacioTren.dat" (Nota: no cal que existeixi a disc, es crea en el moment de fer new), farem:

```
File fitxer = new File("EstacioTren.dat");
```

2. L'accés de lectura i escriptura a un fitxer es fa mitjançant *Streams*. Per llegir d'un fitxer utilitzarem un objecte de tipus ***FileInputStream***, i per escriure a un fitxer utilitzarem un objecte de tipus ***FileOutputStream***:

```
FileInputStream fin=new FileInputStream(fitxer);

FileOutputStream fout= new FileOutputStream(fitxer);
```

3. Finalment, existeixen objectes per gestionar la lectura i escriptura d'un objecte a un *Stream*. Per escriure un objecte utilitzarem un objecte de tipus ***ObjectOutputStream***, mentre que per llegir-lo utilitzarem un objecte de tipus ***ObjectInputStream***:

```
ObjectOutputStream oos = new ObjectOutputStream(fout);

ObjectInputStream ois = new ObjectInputStream(fin);
```

Teniu més informació sobre *streams* en el document d'ajuda a la pràctica 2 al Campus Virtual. No oblideu tancar correctament els objectes d'accés als fitxers.

5.8 Excepcions

Tal com heu fet a la práctica 1, heu d'utilitzar excepcions per fer la gestió dels errors, com per exemple, l'error d'afegir una incidencia sobre una via que ja té una incidènica actualment o l'error de llistar les vies quan no hi ha cap via a la llista.

Creeu la classe ***ExcepcioEstacio*** dins el [paquet Vista] (***prog2.vista***) per representar aquests errors particulars. A la part del codi on es produeix un problema a controlar, heu de crear un objecte d'aquesta excepció i llançar-la, fent servir ***throw*** dins del mètode corresponent. La paraula ***throws*** al final de la capçalera del mètode ens permetrà delegar la captura d'aquesta excepció fins al mètode on es vol gestionar. Utilitzeu el mateix constructor de l'excepció per introduir-hi el missatge d'error adequat i específic de cada error. Finalment, feu servir *try-catch* per capturar la excepció i llavors informar a l'usuari de l'error produït.

Fixeu-vos que la majoria dels mètodes definits a les interfícies permeten llançar ***ExcepcioEstacio***, però a l'hora de fer la sobreescriptura d'aquests mètodes a les vostres classes no heu d'afegir el "throws ExcepcioEstacio;" a la signatura dels vostres mètodes si no ho necessiteu, es a dir, si no heu de llançar una excepció. El fet de posar-ho a la definición dels mètodes de les interfícies és per fer-los més genèrics.

6. Format del Lliurament

El lliurament consistirà en tot el codi generat en els diferents punts de l'enunciat, juntament amb la documentació especificada en aquest apartat.

En concret, cal generar un fitxer comprimit (ZIP) amb el nom dels dos membres de la parella: **Cognom1Nom1Cognom2Nom2_Pr2**, que contingui:

A. El projecte sencer de NetBeans

Tot el codi generat ha d'estar correctament comentat per a poder executar el JavaDoc, generant automàticament la documentació en línia del codi.

B. La memòria de la pràctica (amb un màxim de 6 pàgines).

La memòria ha de contenir els punts descrits en la normativa de pràctiques i els punts següents:

1. Detallar la llista de tasques de cada membre del grup. Per exemple:
    - Implementar menú de la pràctica (Alumne 1)
    - Implementar la classe Via (Alumne 2)
2. Explicar **breument** les classes implementades.
3. Dibuixar el diagrama de relacions entre les classes que has utilitzat a la teva pràctica. No cal incloure la llista d'atributs i mètodes.
4. Explicar quins són els atributs de la classe ***EstacioTren*** i perquè.
5. Expliqueu perquè el mètode isAccessibilitat de la classe ***Acces*** és abstracte.
6. Creieu que l'opció 8 del menú per eliminar una incidència es podria implementar demanant el nom de la via de la incidència, en lloc del número de la incidència que es vol eliminar?
7. Per què creus que es demana que el mètode carregar de la classe ***EstacioTren*** sigui *static*?
8. Detallar les proves realitzades per comprovar el correcte funcionament de la pràctica, resultats obtinguts i accions derivades.

9. Observacions generals.

7. Data Límit del Lliurament

Consultar la tasca oberta pel lliurament al Campus Virtual.