TD n°1 : Introduction au diagramme de classes & premiers pas en java

Exercice 1 : Un petit échauffement sur le diagramme de classes ...

Pour chaque question suivante, proposez un diagramme de classes correspondant à la description donnée :

1. Une **personne** possède un ou plusieurs **ordinateurs.**

2. Considérons les **processus** qui s'exécutent dans un ordinateur.
La liste des processus faisant appel à une ressource est gérée par une **file d'attente** (comme dans le cas d'une imprimante par exemple). Cette file stocke les processus en attente de la ressource.
Chaque processus a bien sûr un **numéro** unique dans la file et ne peut appartenir qu'à une seule file d'attente. Proposez une modélisation pour la file d'attente.

3. Un **fichier** est accessible par un **utilisateur** selon des **droits d'accès** en **lecture** et en **écriture**.

4. Un message électronique (**e-mail**) a un **titre.** Un e-mail est toujours composé d'un **corps** et d'un **en-tête**. Le corps d'un e-mail a un **message** et une **signature**. Des **pièces jointes** peuvent éventuellement être ajoutées à un e-mail. Une même pièce jointe peut être partagée par plusieurs e-mail.
Un e-mail contient également 1 ou plusieurs **destinataires** que l'on peut sélectionner soit par leur **nom**, soit par leur **adresse électronique**.
Dans votre modélisation, utilisez à bon escient des relations de composition et d'agrégation...
Remarques :
- Nous considérerons que le ***corps d'un e-mail*** a un message et une signature.
- Quant à ***l'en-tête d'un e-mail***, elle permet de connaître certaines données concernant le "parcours" du message sur Internet. Les données contenues dans une en-tête d'e-mail ne nous intéressent pas pour l'instant...Vous pourrez toujours enrichir plus tard votre modélisation en visualisant sur votre messagerie l'entête d'un de vos e-mail...

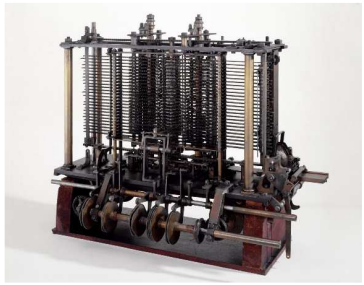
Exercice 2 : Les débuts de l'informatique moderne : Charles Babbage, Ada Lovelace, Alan Turing & Co...

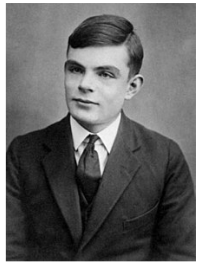
Vous avez choisi de faire des études dans le domaine de l'Informatique (Computer Science).
Mais connaissez-vous l'histoire de l'informatique ?

L'histoire de l'informatique a commencé bien avant la discipline moderne des sciences informatiques, généralement par les mathématiques ou la physique. Les développements des siècles précédents ont évolué vers la discipline que nous connaissons aujourd'hui sous le nom d'informatique. Cette progression, des inventions mécaniques et des théories mathématiques vers les concepts et les machines informatiques modernes, a conduit au développement d'un domaine académique majeur, à un progrès technologique spectaculaire à travers le monde occidental et à la base d'un commerce et d'une culture mondiale massive (extrait https://fr.wikipedia.org/wiki/Histoire_de_l%27informatique)

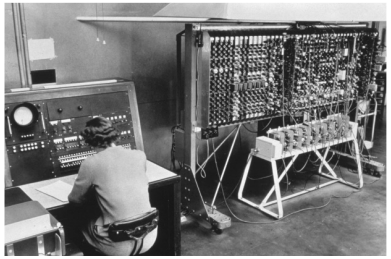
Ada Lovelace, de son nom complet Augusta Ada King, comtesse de Lovelace, née Ada Byron est une pionnière de la science informatique.

Elle est principalement connue pour avoir réalisé **le premier véritable programme informatique**, lors de son travail sur un *ancêtre de l'ordinateur* : la **machine analytique de Charles Babbage**.





Alan Mathison **Turing** est un mathématicien et cryptologue britannique, auteur de travaux qui fondent scientifiquement l'informatique.
Pour résoudre le problème fondamental de la décidabilité en arithmétiques, il présente en 1936 une expérience de pensée que l'on nommera ensuite **machine de Turing** et des concepts de programme et de programmation, qui prendront tout leur sens avec la diffusion des ordinateurs, dans la seconde moitié du XX^{e} siècle.



Textes et images extraits de :
https://fr.wikipedia.org/wiki/Ada_Lovelace , https://fr.wikipedia.org/wiki/Alan_Turing ,
https://www.sciencemuseum.org.uk/objects-and-stories/lovelace-turing-and-invention-computers

Votre cahier des charges : Imaginez maintenant que le département Informatique soit votre nouveau client et que pour les JPO, il vous demande de créer un jeu (une sorte de memory par exemple) qui permettrait d'associer à quelques personnages célèbres de l'histoire de l'informatique la machine sur laquelle il a travaillé.

Mettre en place ce jeu nécessite donc de pouvoir manipuler des personnages célèbres (***famous computer pioneer***) et des machines (**device**).

❑ Focus sur la classe `Device` :

L'application devra manipuler des objets de type **`Device`**, tels que la machine de Babbage ou la machine de turing, …
Pour l'instant, seuls le nom et l'année d'invention de cette machine nous intéresse.

1. Après cette rapide **analyse**, vous pouvez passer à la **phase de conception**.
   Commencez par modéliser la classe **`Device`** à l'aide d'un diagramme UML en tenant compte des besoins du client
   Quels sont ses attributs, ses méthodes ?
   A-t-on besoin d'ajouter des setteurs et/ou des getteurs dans le contexte du client ?
   Que pouvez-vous dire sur le(s) constructeur(s) à écrire ?
   Justifiez vos réponses.

2. Après la conception vient la **phase d'implémentation** et votre langage de prédilection est désormais le Java !
   Implémentez la classe **`Device`** en Java

3. Après la phase d'implémentation vient la **phase de test**.

   Pour tester votre implémentation, implémentez une classe **`Main`** avec une méthode **main** de signature `public static void main(String[] args)` qui contiendra le premier jeu d'essai suivant :

   - **création d'une *instance*** *(objet)* de type **`Device`**.
     Cet objet sera nommé **`babbageMachine`** et instancié avec les *bonnes* valeurs qui permettront de procéder à l'affichage suivant sur la console :
     `Babbage Analytical Machine has been invented in 1837`

   - l'**affichage** sera obtenu en utilisant l'instruction **`System.out.println`** qui prendra juste en paramètre un appel à la méthode **toString()** (qui comme son nom l'indique renvoie un **String**) sur l'objet **`babbageMachine.`**
     La méthode **toString()** devra bien sûr être ajoutée à la fois dans votre diagramme UML et dans votre implémentation Java ! C'est une méthode qui, comme ici, est habituellement utilisée pour renvoyer l'état d'un l'objet (rappel : l'état d'un objet correspond à la valeur de ces attributs).

   Vous pouvez maintenant implémenter un second jeu d'essai qui procédera cette fois-ci à :

   - **Instanciation** d'un objet de **`Device`** nommé **`turingEngine`** dont l'appel de la méthode **`toString()`** sur cet objet permettra de procéder via un **`System.out.println`** à l'**affichage** console suivant :
     `Turing Engine has been invented in 1936`

❑ Focus sur la classe `ComputerPioneer` :

L'application devra manipuler des objets de type **`ComputerPioneer`**, tels que Ada Lovelace ou Alan Turing, …
Pour commencer, le problème sera simplifié ***en posant l'hypothèse suivante*** :
Seule l'identité d'un personnage célèbre nous intéresse c.-à-d. :
- un (seul) prénom (le premier)
- un (seul) nom

4. Après cette rapide **analyse**, vous pouvez passer à la phase de **conception**.
   Sur votre diagramme UML précédent, à côté de la classe **`Device`**, modéliser la classe **`ComputerPioneer`** à l'aide d'un diagramme UML.

5. **Phase d'implémentation** : Implémentez la classe **`ComputerPioneer`** en Java

6. **Phase de tests**

   Complétez la méthode **main** en implémentant les deux jeux d'essais suivants :

   - Instanciation d'un objet de type **`ComputerPioneer`** nommé **`adaLovelace`** dont l'appel de la méthode **`toString()`** sur cet objet permet de procéder via un **`System.out.println`** à l'affichage console suivant :

     `Ada Lovelace is a pioneer in Computer Science`

   - Instanciation d'un objet de type **`ComputerPioneer`** nommé **`adaLovelace`** dont l'appel de la méthode **`toString()`** sur cet objet permet de procéder via un **`System.out.println`** à l'affichage console suivant :

     `Alan Turing is a pioneer in Computer Science`

❑ Ajouter une relation entre la classe `ComputerPioneer` et `Device`

Dans notre contexte métier (celui du jeu pour les JPO), on souhaite qu'un personnage célèbre (pioneer) puisse être associé au matériel (device) sur lequel il travaille.
Pour commencer ce projet le plus rapidement possible, nous simplifierons le problème en **posant l'hypothèse suivante :**

Un personnage célèbre travaille uniquement sur un appareil (à un instant t)

D'un point de vue **Conception Orienté Objet**, cela signifie :
qu'un ***objet*** de type `ComputerPionner` ***est (re)lié à*** un seul ***objet*** de type `Device`

comme le montre le diagramme d'objets (photo instantanée du système à un instant t) suivant entre `adaLovelace` et `BabbageMachine`.

Comment ce lien entre objets (qui permet la communication entre ces deux objets) se répercute-t-elle au niveau du diagramme de classes ?

- Après avoir rappelé la **différence entre une classe et un objet**,
- Interrogez-vous sur :
  - Quel **type de relation** ?
  - Quel(s) **sens de navigabilité** et pourquoi ce(s) choix (en précisant quel sera l'impact du choix de la navigabilité lors de la phase d'implémentation)
  - Quelle **multiplicité** ?

7. **Phase de conception** : Après avoir répondu aux questions précédentes, complétez le diagramme UML (qui contient déjà les classes **`Device`** et `ComputerPionner`) de manière à modéliser la solution que vous avez retenue pour répondre à ce problème.

8. **Phase d'implémentation :** Modifiez le code Java précédemment implémenté pour qu'il reflète la nouvelle conception (le nouveau diagramme de classes) permettant de savoir sur quel appareil un personnage célèbre travaille.
   Hypothèse : Pour simplifier le problème, on considère que pendant toute la durée du programme ce *device* restera inchangé c-a-d que le personnage célèbre travaille toujours sur un seul et même *device* pendant la durée du jeu.

9. **Phase de tests :**

   ➔ Réécrire la méthode **`main`** de manière à obtenir les deux jeux d'essais suivants :

   - ***Jeu d'essai n°1 :***
     Après l'instanciation d'un **`Device`** nommé **`babbageMachine`** et d'un **`ComputerPioneer`** nommé **`adaLovelace`**, l'affichage attendu en mode console sur l'objet **`adaLovelace`** devra être :

```
Ada Lovelace is a pioneer in Computer Science who works on Babbage Analytical Machine invented in 1837
```

   - ***Jeu d'essai n°2 :***
     Après l'instanciation d'un **`Device`** nommé **`turingEngine`** et d'un **`ComputerPioneer`** nommé **`alanTuring`**, l'affichage attendu en mode console sur l'objet **`alanTuring`** devra être :

```
Alan Turing is a pioneer in Computer Science who works on Turing Engine invented in 1936
```


❑ Ajouter du comportement à la classe `ComputerPioneer` :
Être capable de dire si un pionnier travaille sur un device donné

Pour mettre en place le jeu, il paraît indispensable de pouvoir disposer de la méthode suivante dans la classe **`ComputerPioneer`** permettant ainsi de savoir si le **`device`** passé en paramètre est le même device sur lequel travaille actuellement le `computerPionner`

```
public boolean worksOn(Device device) ;
```

10. **Conception :** Ajouter cette opération dans le diagramme de classes

11. **Implémentation :** Implémenter cette méthode dans votre code Java déjà écrit

12. **Test :**

    - Pour tester le code que vous venez d'écrire vous ajoutez le bout de code suivant dans votre **`main`** comme **jeu d'essai n°3** :

```
public static void main(String[] args) {

      //… code déjà écrit précédemment

      System.out.println("Test case 3 ");
      System.out.println("--------------------");
      System.out.println(adaLovelace.worksOn(babbageMachine));
      System.out.println(adaLovelace.worksOn(turingEngine));
      System.out.println(alanTuring.worksOn(babbageMachine));
      System.out.println(alanTuring.worksOn(turingEngine));
      System.out.println("--------------------");
}
```

      ⇨ Quel affichage obtenez-vous sur la console ?

    - Vous décidez d'ajouter un dernier jeu d'essai dans votre **`main`** (**jeu d'essai n°4)** :

```
public static void main(String[] args) {

      //… code déjà écrit précédemment
      System.out.println("Test case 4 ");
      System.out.println("--------------------");
      Device babbage = new Device ("Babbage Analytical Machine",1837);
      Device turing = new Device ("Turing Engine",1936);
      System.out.println(adaLovelace.worksOn(babbage));
      System.out.println(adaLovelace.worksOn(turing));
      System.out.println(alanTuring.worksOn(babbage));
      System.out.println(alanTuring.worksOn(turing));
      System.out.println("--------------------");
}
```

      ⇨ Quel affichage obtenez-vous sur la console ?

    - Est-ce que les affichages votre jeu d'essai n°3 et votre jeu d'essai n°4 sont identique ?
    - Est-il souhaitable que les affichages du jeu d'essai n°3 et du jeu d'essai n°4 soient identique ?

Pour votre culture personnelle :

Vous connaissez surement bien d'autres personnalités célèbres en informatique que vous pourrez pendre le temps de (re)découvrir (après ce TD) avec le jeu des 7 Familles de l'informatique disponible à l'adresse suivante : https://interstices.info/jeu-de-7-familles-de-linformatique/

Une liste plus complète et plus formelle de pionniers de l'informatique est également disponible sur : https://history.computer.org/pioneers/ ainsi qu'une timeline qui présente l'évolution de l'informatique de l'antiquité à nos jours : https://ieeecs-media.computer.org/assets/pdf/timeline.pdf

Travail à rendre pour la prochaine séance de TD :

Vous allez participer au développement du projet qui va piloter le prochain robot (rover) sur la lune. Pour simplifier le problème, on considère dans un premier temps que le robot se déplace sur une carte 2D. Le robot démarre toujours du centre de la carte c-a-d en (0,0) et il est dirigé vers le nord. Le robot (rover) doit pouvoir avancer, reculer, tourner à droite et gauche.

Proposez un diagramme de classes qui permet de modéliser ce problème.
Vous dessinerez ce diagramme ***à la main*** sur une feuille libre où vous mentionnerez nom, prénom et groupe.