420-V41-SF
PROGRAMMATION D'APPLICATIONS
TRAVAIL PRATIQUE #1 – Partie 1
« COMPOSÉS CHIMIQUES »

- Compte pour 10% de la session.
- Travail à réaliser seul
- 1 remise (voir la date de remises sur LÉA).

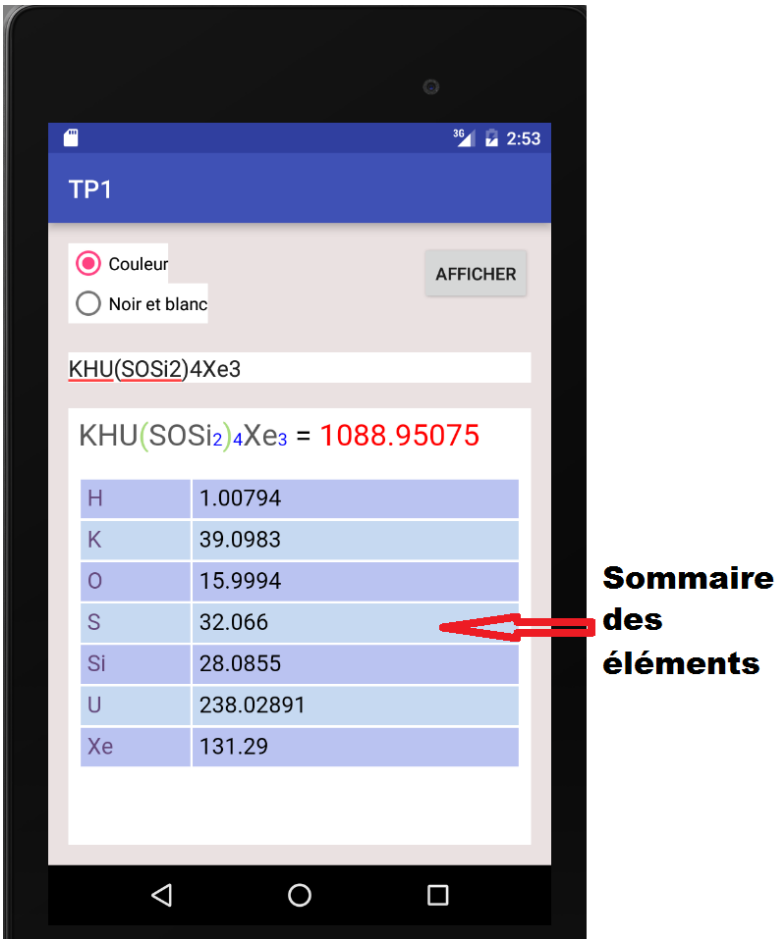
Objectifs pédagogiques

Ce premier travail vise à développer les compétences suivantes :

016T	Appliquer une approche de développement par objets.
0176	Apporter des améliorations fonctionnelles à une application.
017D	Concevoir et développer une application hypermédia dans des réseaux internes et mondiaux.
0177	Assurer la qualité d'une application

Fonctionnalités

À partir d'un composé chimique saisi et d'une vue sélectionnée, l'application doit afficher des informations concernant le composé chimique. Exemple :



Le poids d'un composé chimique

Un composé chimique à un poids qui est calculé à partir de sa formule chimique. Exemple :

Composé chimique	**Poids atomique**	**Détail du calcul** (la masse atomique d'un élément chimique se trouve dans le tableau périodique)
H_2O	18,01528	H = 1,00794 O = 15,9994 Poids = (H * 2) + O
$Cr_2(SO_4)_3$	392,183	Cr = 51,9961 O = 15,9994 S = 32,066 Poids = (Cr * 2) + (S + (O * 4)) * 3

⇨ Voir le site http://www.webqc.org/mmcalc.php pour connaître le poids d'un composé chimique

Structure et configuration de la solution Visual studio

La solution suit le plus possible le patron de conception MVC et est composée de 3 packages principaux :

- **com.v41.tp1.modele**
  - C'est le domaine de traitement des données (logique, modèle)
  - Un composé chimique est supposé recevoir une formule chimique déjà valide.

- **com.v41.tp1.vuecontroleur**
  - Interface utilisateur console dans un premier temps, puis Android dans un second
  - Comprend le contrôleur qui analyse la forme de la formule chimique entrée
  - Doit consulter le tableau périodique (du modèle) pour savoir si les atomes présentés dans le composé existent.

- **com.v41.tp1.vuecontroleur.test / com.v41.tp1.modele.test**
  - Ce sont les tests liés au composé chimique et au validateur chimique
  - Les tests sont exécutés directement dans IntelliJ
  - Le passage de tous les tests assure que vos composés chimiques seront correctement analysés

Contraintes de réalisation

- **Pour la partie 1, vous devez** poursuivre le développement de l'application qui se trouve dans le fichier **TP1-Départ.zip**.

- **Il est recommandé** de vous baser sur le digramme de classe qui se trouve dans le fichier ***TP1-Partie1.class.violet/ TP1-Partie1.png***

- **Vous ne devriez pas avoir besoin** d'ajouter de nouvelles classes.

- **Il est recommandé** de ne pas ajouter de méthodes et propriétés ***public*** aux classes, mais…

- **Vous pouvez** sans contrainte ajouter des méthodes et propriétés ***private*** aux classes.

- **Vous ne pouvez pas** modifier les tests existants, mais **vous pouvez** en ajouter ou en créer d'autres.

- **Toutes modifications aux diagrammes de classes ou aux tests existants** doivent être acceptées par votre professeur.

- Une fois la version IntelliJ fonctionnelle, vous l'adapterez pour Android avec Android Studio.
    - *Une partie II du TP sera présentée à la semaine 5 qui expliquera consignes et détails techniques pour le port du TP vers Android*

Quelques précisions sur les classes et projets

- La classe **ElementChimique** représente un élément chimique du tableau périodique. Par exemple, l'hydrogène est l'élément chimique #1 du tableau périodique. Son symbole est H et sa masse atomique est de 1,00794.

- La classe **TableauPeriodique** est un singleton (enum). Cette classe contient l'ensemble des éléments chimiques du tableau périodique. Le contenu du tableau périodique est chargé à partir du fichier **tableauPeriodique.txt**

- **Important** : normalement, vous ne deviez pas à avoir à modifier **TableauPeriodique. ElementChimique** n'a qu'une méthode à compléter et ne devrait pas être modifié au-delà de ça.

- La classe **ComposeChimique** représente une formule chimique composée de **Jetons**. Normalement, cette classe reçoit une formule chimique valide. Elle calcule le poids de la molécule reçu et crée une composition où on retrouve le symbole et le poids de chaque molécule, sans doublon.

- La classe **Jetons** représente une partie syntaxique d'un **ComposeChimique** en vue de traitement subséquent.

    - Exemple : **Cr2(SO4)3**
    - Un jeton peut être :
        - Un symbole d'un élément chimique : Cr, S, O, Al, U, etc.
        - Une parenthèse ouvrante (ou fermante)
        - Un nombre positif 2, 4, 3, 12, 25, 536
        - **Attention, les nombres 0 et 1 sont invalides**

- L'interface PortailModele, permet à ComposeChimique, classe centrale dans le modèle de ce projet, d'envoyer des événements à la vue, suivant le modèle MVC

- Les classes de tests ne devrait pas être modifiés, sinon pour dé-commenter les tests unitaires présentés ou alors pour en ajouter de nouveaux.

- Le StringWrapper permet de contourner un problème avec les String immutables de Java (Aucune simulation de passage par référence possible avec un String).

- Le contrôleur va traiter les entrées de l'utilisateur à l'aide du validateur chimique (qui est aussi une classe faisait partie du contrôleur selon le modèle MVC), puis il signale à la vue si la formule chimique est valide ou non. Si elle l'est, elle envoie la molécule au modèle qui calcule le poids de celle-ci et envoi un signale à la vue que la molécule est valide, pesée et est prête à être affichée. Si elle est invalide, elle doit dire à la vue quelle est précisément la nature de l'erreur.

- Le validateur Chimique vérifie si l'utilisateur a entré une molécule valide (il aura besoin de consulter le tableau périodique pour cela – c'est ici qu'il faut faire ça et non dans le modèle, qui ne fait jamais de traitement de validation). Si c'est le cas, il peut construire la liste des jetons à ce point-ci, pour éviter de la duplication de code (moins MVC, mais plus efficace). S'il y a erreur, le validateur doit mentionner la nature précise de l'erreur.

- La vue, si la formule est valide, va afficher le poids atomique total de la molécule et donner le sommaire que chaque atome dans la molécule. Même si un atome est présent plusieurs fois dans la molécule, on le retrouve une seule fois dans le sommaire. En cas de formule invalide, signale la chose avec la nature de l'erreur. Vous pouvez afficher la première situation invalide trouvée; c'est suffisant dans le cadre de ce TP.

- Vous devez commenter le code en mode Javadoc, y compris le code fourni. La forme est ce qui est le plus important, mais le fond doit décrire avec justesse ce que vous faites. Vous devrez générer et remettre votre Javadoc.

- Une fois le projet IntelliJ parfaitement fonctionnel vous aurez à adapter votre projet pour Android. Les seules modifications seront :
  1- la partie vue en entier 2 - la méthode de chargement du fichier des éléments chimiques.

Évaluation

Tests unitaires - Exécution des tests existants	30%
Documentation du code - Respect des standards de Javadoc - Documentation claire et explicative - Javadoc générée correctement	15%
MVC - Le programme doit respecter le patron de conception MVC pour son fonctionnement.	10%
Fonctionnalités - Le programme est entièrement fonctionnel (Java et Android) - Respect des contraintes - Il n'y a aucune erreur à l'exécution - Il n'y a aucun bogue connu - Les modifications entre les versions Java et Android doivent être minimales (seuls les points soulignés plus haut devraient être modifiés)	25%
Android - Présentation et fonctionnement de l'application Android tel que demandé (voir partie 2)	20%

L’opérationnalité des fonctionnalités ne garantit pas la cote de passage. La conception de l’algorithme, des jeux d’essais de même que l’architecture et la qualité du code réalisé ainsi que des documents complet et professionnel sont primordiaux. Si le professeur le juge à propos, toute étudiante ou tout étudiant pourra être convoqué à une rencontre d'évaluation pour vérifier son degré d’acquisition des connaissances et d'appréhension de la solution proposée.

Documents à remettre

- Projet IntelliJ fonctionnel
- Projet Android Studio fonctionnel (effacez le contenu de votre répertoire out)
- Javadoc générée (dossier à part)

Pour le dimanche 4 mars à minuit : groupe 02
Pour le lundi 5 mars à minuit : groupe 01

Bonus

Des éléments supplémentaires pourront être mis en place afin de potentiellement obtenir un bonus.

- Chaque fonctionnalité supplémentaire pourra valoir un bonus maximum de 5%, pour un bonus total maximal de 10%.
- La note maximale du travail pratique reste de 100%
- Pour se faire proposer une fonctionnalité supplémentaire, l'étudiant doit avoir un projet dans un état d'avancement jugé satisfaisant par le professeur (en général 80%).
- Les fonctionnalités supplémentaires seront présentés par le professeur ou suggérés par l'étudiant et approuvé par le professeur.