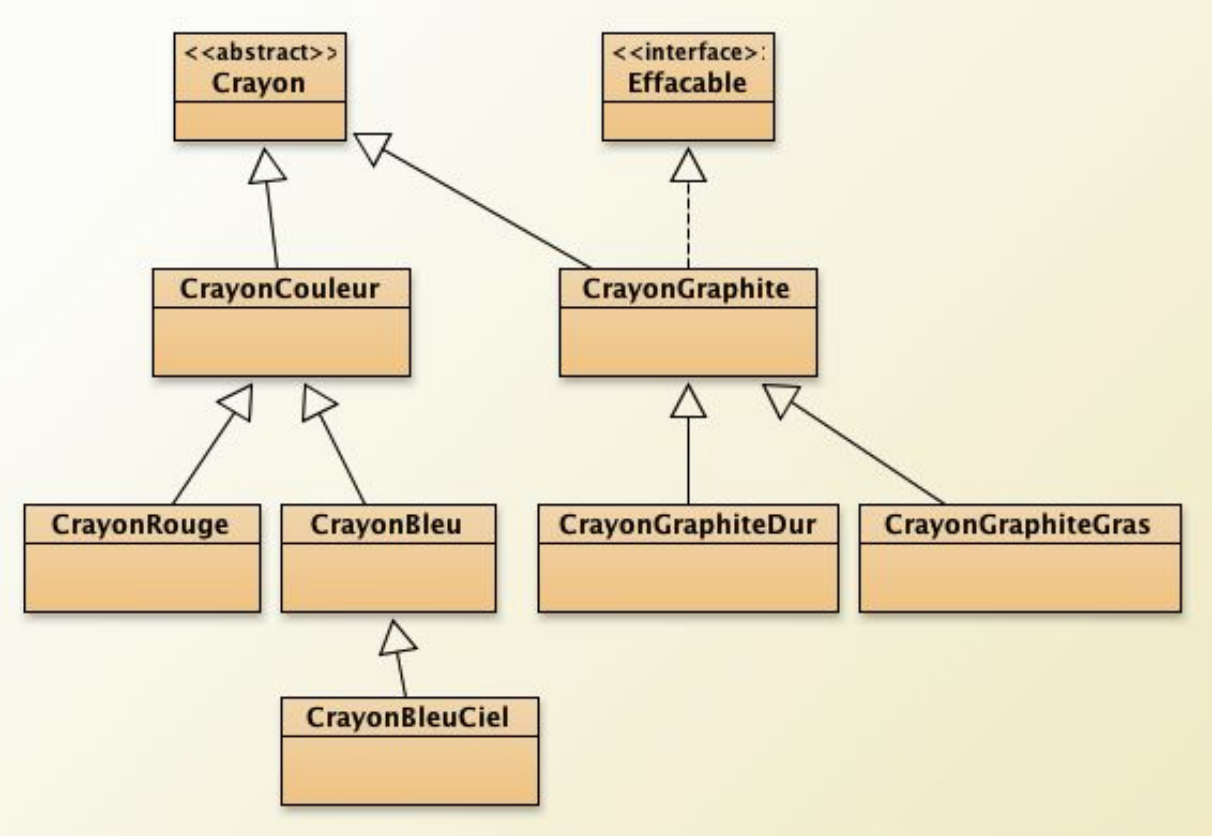
UEF4.3. Programmation Orientée Objet

EXAMEN FINAL

02 heures-Documents interdits

Exercice 1 (6pts): Soit les classes suivantes :

1. Tracez le diagramme de classe (1pt)



s'il manque au plus 2 classes -0,25, plus de 2 -0,5
s'il manque une relation d'héritage -0,5
s'il manque l'implémentation -0,5

2. Qu'affiche le programme suivant :

Solution :**1,25=0,25*5**
Crayon Bleu Ciel
Crayon Gris
Crayon Couleur
Crayon Graphite
Crayon Bleu

2. On effectue les affectations suivantes. Pour chaque affectation, dites si elle s'exécute sans erreur, si elle provoque une erreur à la compilation ou si elle provoque une erreur à l'exécution. Si il y a une erreur expliquez pourquoi.

Solution **3,75**

Ligne	Solution
CrayonCouleurc0 = trousse[0];	Erreur compilation =>cast**0,25** CrayonCouleur c0 = (CrayonCouleur) trousse[0];**0,25**
CrayonRougec2 = trousse[2];	Erreur compilation CrayonRouge c2 = (CrayonRouge) trousse[2];**0,5**
CrayonBleuc3= (CrayonBleu) trousse[3];	Erreur exécution Crayon Graphite n'est pas un Crayon bleu**0,5**
Crayon c4=trousse[4];	Ok**0,25**
c0 = trousse[1];	Erreur compilation c0 = (CrayonCouleur) trousse[1]; et changer l'affectation de trousse de 1 sinon erreur à l'exec **0,5**
CrayonGraphitec5 = trousse[0];	Erreur compilation CrayonGraphite c5 = (CrayonGraphite) trousse[0]; et changer l'affectation de trousse de 1 sinon erreur à l'exec**0,5**
c3 = trousse[0];	Erreur compilation c3 = (CrayonBleu) trousse[0];**0,5**
Effacablee1= trousse[1];	Erreur compilation Effacable e1= (Effacable) trousse[1];**0,5**

Exercice 2 : 14pts

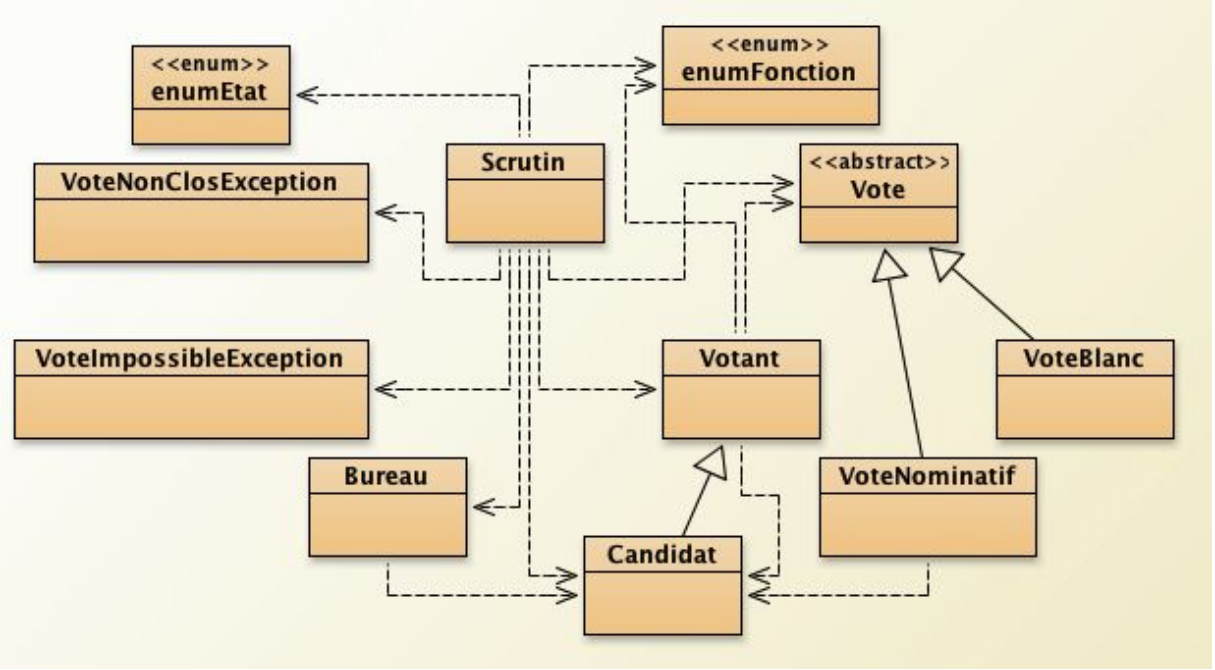
1.

A.Quels sont l'intérêt et l'importance du qualificatif final dans la méthode vote**(1pt).**
Intérêt 0,5 : Pour assurer qu'elle ne soit pas redéfinie dans une classe dérivée
Importance 0,5 : pour la sécurité, ne pas falsifier les résultats. Pour na pas changer les règles du vote (le comportement) dans une classe dérivant de Scrutin

b. Quel est le rôle de *VoteImpossibleException*?**(0,5pt)**

Réponse :Elle est lancée si le votant essaie de voter alors qu'il a déjà voté ou que le vote est clos

2. le diagramme de classe**(2,75)**



a. Pourquoi Candidat hérite–elle de Votant ? (0,25)

Rep : Car le candidat est aussi un employé qui peut voter « un Votant », il a donc les mêmes caractéristiques, mais en plus, on a besoin du score et/ou du nombre d'année d'expérience pour le bureau

b. Quel est l'intérêt de créer un héritage entre VoteBlanc, VoteNominatif et Vote ? (0,5)

Rep : c'est pour laisser le Scrutin ouvert à d'autres types de vote (Oui, Non par exemple) et pour faciliter le calcul des statistiques

c. Expliquez brièvement quelle collection doit-on utiliser pour : (total : 1,5)

i. assurer qu'un vote n'est enregistré qu'une fois ? **0,5**

Rep : un Set<Vote>, car ne permet pas les doublons, le tri n'est pas important donc a priori un HashSet (si treeset -0,25 si set ou hashset 0,5)

ii. récupérer l'ensemble des candidats par fonction ? **0,5**

Rep : une Map, clé est la fonction, valeur l'ensemble des candidats pour cette fonction

iii. vérifier qu'un votant a déjà voté ? **0,5**

Rep : une Map, (HashMap ou TreeMap), la clé est de type Votant, et la valeur de type Boolean, qui sont initialisés au départ à false

d. Comment peut on assurer l'anonymat du vote ? (0,5)

1. Vote ne doit pas être un attribut de Votant. Il est retourné par la méthode choisir qui est appelée dans la méthode voter de Scrutin 0,25
2. En enregistrant le vote et les votants séparément dans la classe Scrutin c-à-d qu'on ne doit pas avoir les deux informations dans la même structure de donnée 0,25

3. Détaillez dans un tableau les attributs, constructeurs et méthodes des classes Scrutin, Vote et dérivés, Votant, Candidat, Bureau (**7,75). +bonus 1,75**

Class Scrutin/**3,25 et peut aller jk 4**	Dés**ignation**
Liste attributs	
private String date;	0,25
private enumEtatetat;	0,25
private Map<enumFonction, Set<Candidat>> listeCandidats;	0,25 **Accepter les list, arraylist car déjà sanctionner dans la question précédente.**
private Map<Votant, Boolean>listeVotants ;	0,25**Accepter leslist,arraylist car déjà sanctionner dans la question précédente.**
private Map<enumFonction, Set<Vote>>lesVotes ;	0,25 accepter aussi un Set<Vote>**Accepter leslist,arraylist car déjà sanctionner dans la question précédente.**
private Bureau bureau ;	Si mis +0,25 sinon pas de sanction
private int nbrVotant	On peut le déduire autrement
private int scoreVoteNul ; private Map<Candidat, Integer>resultats ;	Noté si manque un autre attribut, utile pour les résultats dans candidat
Liste des méthodes	
public Scrutin(Set<Votant>listeVotants, Map<enumFonction,Candidat>listeCandidats, String date)	0,5 , si pas date 0,25**Accepter leslist,arraylist car déjà sanctionner dans la question précédente.**
public final void voter(Votantvotant) throws VoteImpossibleException	0,25 +0,25 pour le throws
public Bureau getBureau() **throwsVoteNonClosException**	0,25+0,25 pour le throws
public void getStatistique() **throwsVoteNonClosException**	0,25+0,25 pour le throws
public double ***tauxParticipation()***	0,25
public void ***depouille()***	0,25

Votant**(/1pt peut aller jk 1,25)**	Désignation
Liste attributs	
private String identifiant ;	0,25
private enumFonction fonction ;	0,25
Liste des méthodes	
public Votant(String identifiant,enumFonction fonction)	0,25
public Vote choisir(Set<Candidat>choixPossible)	0,25
public boolean equals(Object)	Si ce n'est pas mis dans candidat la compter ici
public int hashcode()	S'il met un HashSet, bonus +0,25

Candidat extends Votant **(Σ /1pts peut aller jk**	Je n'ai pas comptabilisé compareTo

1,5)implemets Comparable<Candidat>	et Comparable, ça sera un bonus de 0,25 chacune=> 0,5
Liste attributs	
private int experience ;	0,25
private int score=0 ;	0,25 accepter aussi une structure de résultat dans scrutin et mettre ce point la dedans
Liste des méthodes	
public Candidat(String identifiant,enumFonction fonction, int experience)	0,25
public boolean equals(Object)	0,25 Car on a un Set dans choisir, la compter une fois ici ou dans votant
public int compareTo(Candidat c)	Pour ordonner selon le score. Si le score n'est pas ici ça sera selon l'ancienneté

Bureau**(Σ /0,75)**	Désignation
Liste attributs	
private Candidat president;	0,25
private Set<Candidat> membres;	0,25 accepter aussi ArrayList ou Map<enumFonction, Candidat>
Liste des méthodes	
public Bureau(Candidat president, Set<Candidat> membres)	0,25

abstract class Vote Σ **/0,5 peut aller jk 1**	0,25 pour abstract class
Liste attributs	
	Désignation
Liste des méthodes	
public abstract String getVote()	0,25
public boolean equals(Object)	Bonus 0,25
public int hashcode()	Bonus 0,25

class VoteBlancextends Vote Σ **/0,5**	Si rien n'est mis compter extends 0,25
Liste attributs	
private final String val="blanc";	0,25
Liste des méthodes	
public String getVote()	0,25

class VoteNominatif extends Vote Σ **/0,75**	Si rien n'est mis compter extends 0,25
Liste attributs	
private final String candidat ;	0,25
Liste des méthodes	
public VoteNominatif(String candidat) ;	0,25
public String getVote() ;	0,25

4. Donnez le code JAVA de la méthode *voter* de la classe Scrutin. **2pt**

```
public final void voter(Votantvotant) throws VoteImpossibleException{
// vérifier que le votant n'a pas déjà voté sinon lancer l'exception0,5
if ((listeVotants.getValue(votant)==true)|| this.etat ==enumEtat.Clos)) throw new
VoteImpossibleException();
else {
      //récupérer la liste de candidat correspendant à la focntion du votant0,5
enumFonction e=votant.getFonction();
      //appeler la méthode choisir du votant0,5
      Vote v=votant.choisir(listeCandidats.getValue(e));
      //enregistrer le vote indépendament du votant0,5
lesVotes.put(e,lesVotes.get(e).add(v));  //Selon la structure
      }
  }
```