|  |  |
| --- | --- |
| Formatif C | **Introduction aux diagrammes**  **de classes UML** |

|  |
| --- |
| Légende pour tous les formatifs:  ✪ Question "pour les Pro" (optionnelle, non requis dans les mini-projets à remettre)  ❓ Réflexion: vous devriez connaître la réponse à cette question, mais on ne vous demande pas  de fournir cette réponse (ni d'intégrer votre solution au code du projet à remettre) |

1. ❓Vocabulaire: complétez les équivalences dans le tableau suivant:

|  |  |
| --- | --- |
| **Langage Java** | **Langage UML** |
| propriété **ou** champ **ou**  variable d'instance |  |
| méthode |  |

1. Pratiquez autant de fois que désiré le quiz *UML*, dont le numéro d'identification vous sera donné en classe (5 2 8 6 5 9). Le même quiz sera donné en classe: le résultat remis ce jour-là vaudra pour la question 2 de ce formatif.
2. Normalement, le programmeur conçoit d'abord son diagramme, puis le code ensuite. Pour vérifier vos connaissances, cet exercice vous demande de faire l'inverse! À l'aide du logiciel ArgoUML, tracez le diagramme de classes UML qui correspond à chacun des projets suivants, selon les conventions apprises en classe. Ajouter tous les détails connus sur vos diagrammes (associations avec multiplicité et navigabilité; stéréotypes discutés en classe, opérations avec paramètres et type de retour s'il y a lieu; etc.).
   * Placez tous vos diagrammes dans un seul et même fichier *MesDiagrammes.zargo*
   * Tous vos types doivent être en français (entier, réel, etc.)
   * Utilisez une zone de texte pour indiquer la lettre de chaque numéro (a, b, c, d etc.)



* + Les sections vides des classes doivent être masquées.
  + Quand vous aurez terminé, exportez l’ensemble de vos diagrammes sous la forme d’une image *MesDiagrammes.****png***, en faisant la commande *Fichier /Enregistrer les graphiques*.

|  |  |
| --- | --- |
| a | public class Vol{  private Avion avion;  }  public class Avion{  } |
| b | public class Voiture {  private Proprietaire proprio;  private Fabricant fabricant;  }  public class Adresse {  }  public class Proprietaire {  private Adresse adresse;  }  public class Fabricant {  private Adresse adresse;  } |

|  |  |
| --- | --- |
| c | public class Etudiant {  private Adresse adresse;  }  public class Adresse {  }  public class EtudiantSIM extends Etudiant {  } |
| d | public class EtudiantCegep {  private DossierDeNotes dossier;  private Cours[ ] lesCours;  private ArrayList<Prof> lesProfs;  }  public class DossierDeNotes { }  public class Cours { }  public class Prof { } |
| e | *Sachant que Comparable est une interface standard dans la plupart des langages:*  public class Point implements Comparable {  } |
| f | public class Point implements ElementGeometrique { }  public interface ElementGeometrique { } |
| g | public class Contrat {  private TexteLegal texte;  public void approuver( Notaire notaire ) { } //supposez que cette méthode est  //pertinente et apparait donc sur le diagramme UML  }//fin classe  public class Notaire {  }  public class TexteLegal {  } |
| h | public class Client {  private Adresse adrDomicile;  private Adresse adrBureau;  }  public classe Adresse {  }  *Plusieurs diagrammes différents UML auraient pu être à l'origine de ce code. Dessinez-en deux.* |
| i | public class Metal {  private String typeMetal; //supposez que cette propriété est pertinente  //et apparait donc sur le diagramme UML  private Coefficient coeff;  }  public class Coefficient {  } |
| j | *Pour ce numéro, supposez que toutes les propriétés et toutes les méthodes et leurs détails sont pertinents et apparaissent donc sur le diagramme UML*  public class EtudiantSIM {  private String nom;  private double coteR;  private double evaluations[ ];  private static double nbInscrits;  private final String COLLEGE = “Maisonneuve”;  public EtudiantSIM ( ) {  }  private double bonification( double *uneCote* ) {  }  } |
| k | public **enum** TypeTaille {  PETIT, MOYEN, GRAND  }  public class TriangleEqui {  private TypeTaille laTaille;  } |

1. ~~Chargez le fichier~~ *~~questionC4.docx~~*~~. Vous devez y écrire tout le code Java qui correspond fidèlement au diagramme de classe suivant.~~

* ~~Il ne s'agira évidemment qu'un ensemble de "coquilles" de classes avec leurs propriétés et méthodes vides (comme celles présentées au numéro précédent), puisque les fonctionnalités de l'application sont inconnues.~~

