Programmation orientée objets

Documents autorisés, appareils électroniques interdits. 1.5 heures.
La présentation sera prise en compte.

Rappel : pensez à mettre votre nom, numéroter les copies (avec nombre total) et cacheter.

Exercice 1 (Questions) (12 points)

Question 1 : En programmation orientée objet, qu'est-ce qu'un objet ?

Question 2 : Expliquez le but et le fonctionnement des exceptions.

Question 3 : Pourquoi faut-il toujours redéfinir `hashCode` (ou équivalent) quand on redéfini `equals` (ou équivalent) ?

Question 4 : Quelle est la différence entre la redéfinition de méthodes et la surcharge ?

Question 5 : Quel est l'intérêt d'une machine virtuelle (dans le contexte de Java) ?

Question 6 : En Java, quelle est la différence entre l'opérateur `==` et la méthode `equals` ?

Question 7 : Pourquoi est-il préférable de ne dépendre que des interfaces ?

Question 8 : Donner les critères permettant de garantir la substituabilité d'un sous-type.

Question 9 : Que signifie le mot-clé Java `final` utilisé sur :
- un attribut,
- une méthode,
- une classe.

Question 10 : Qu'est-ce que l'héritage ? Quelles alternatives peut-on utiliser ?

Question 11 : Expliquez ce que sont les classes génériques.

Question 12 : Quels sont les intérêts d'utiliser des accesseurs ?

Exercice 2 (2 points)

Soient les deux classes Java suivante :

```
class A {
  private int a;
  A(int v) {
    a = v;
  }
  public int foo(A other) {
    return a + other.a;
  }
}
```

```
class B extends A {
  B(int v) {
    super(v);
  }
  @Override
  public int foo(B other) {
    return 42;
  }
}
```

Ce programme affiche une erreur à la compilation. Pourquoi ?

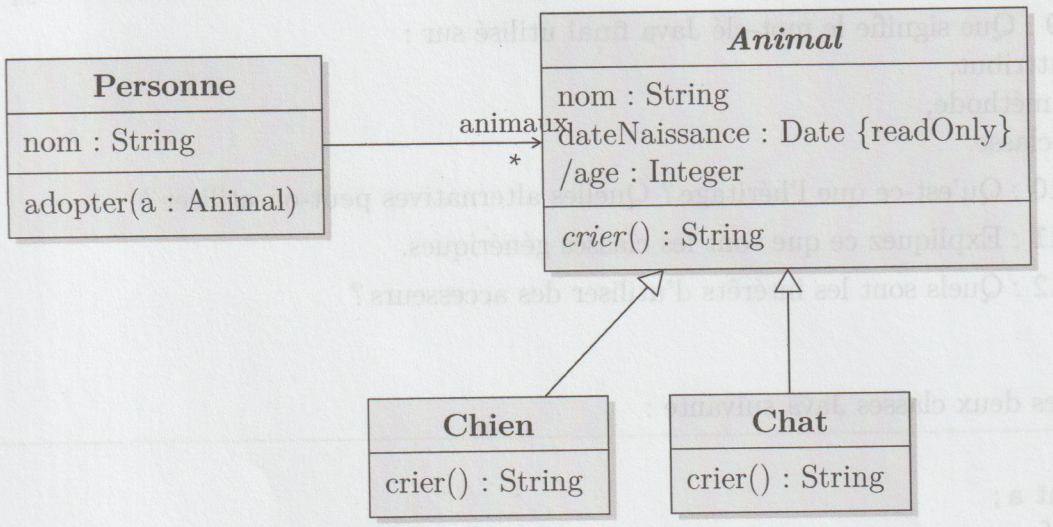
Exercice 3 (To be or not to be...) (2 points)

Le programme Java suivant compile et s'exécute sans erreurs. Qu'affiche-t-il et *pourquoi* ?

```
public class Indecision {
  public static void main(String[] args) {
    System.out.println(decision());
  }
  static String decision() {
    try {
      return "to be";
    } finally {
      return "not to be";
    }
  }
}
```

Exercice 4 (5 points)

Soit le diagramme de classe conceptuel suivant :



Donnez le code Java correspondant, en respectant les conventions du langage (attention, le diagramme n'est pas au niveau d'abstraction de l'implémentation).

Vous pourrez utiliser la class `java.time.LocalDate` pour représenter la date. La méthode statique `now` de cette classe retourne la date courrante. Le détail de son utilisation n'est cependant pas important ici.