NOM : **Prénom :** **Signature :**

Examen CPO

(1h30, feuille A4 autorisée et à rendre)

— Pas de question. En cas de doute, noter le choix fait sur la copie.
— Il est conseillé de lire complètement le sujet avant de commencer à y répondre !
— Inutile de mettre commentaires de documentation et clauses d'importation.
— Barème indicatif (sur 30 points) :

exercice	1	2	3	4	5	6
points	8	4	5	4	5	4

Compréhension du cours

Exercice 1 Répondre de manière concise aux questions suivantes.
1.1. Indiquer et expliquer ou corriger les erreurs de compilation qui seront signalées sur le programme du listing 1 sachant que la classe A ne contient pas d'erreur. **Répondre sur le sujet.**
1.2. Comparer les notions de `Set` et `List`.
1.3. À quelles conditions sur `y` peut-on écrire le code suivant et quel est le mécanisme mis en œuvre ?

```
for (T x : y) {
    m(x);
}
```

1.4. Dessiner un diagramme de classe UML faisant apparaître les éléments suivants de l'API Swing : MouseListener, MouseEvent, Event, addMouseListener, JButton et mouseClicked.
1.5. Indiquer ce que font les deux méthodes de l'API introspection de Java : `getMethods` et `getDeclaredMethods`. En particulier on soulignera les différences entre les deux.

Lecteur de caractères

Notre objectif est de pouvoir lire des caractères depuis plusieurs sources (chaîne de caractères, tableau, fichier...) en disposant de diverses fonctionnalités.

Exercice 2 : Reader
Un `Reader` permet d'obtenir le prochain caractère d'une source grâce à une méthode `read()` sans paramètre. L'exception `EOS` sera levée si on veut lire un caractère d'une source vide. On ne sait actuellement rien sur la source (chaîne, tableau, fichier...).
2.1. Écrire l'exception `EOS` qui ne doit pas être vérifiée par le compilateur.
2.2. Écrire en Java l'interface `Reader`.
2.3. Expliquer ce qui justifie de faire de `Reader` une interface.


```
1  class A {
2      protected int a;
3      public A(int a)      { this.a = a; }
4      public void m(int b) { System.out.println("A::m(" + a + ")"); }
5  }
6  class E implements Exception {}
7  class B implements A {
8      public B() {
9          a = 10;
10     }
11
12     public void m(int b) {
13         if (a < b) {
14             throw new E();
15         }
16         System.out.println("B::m(" + a + ")");
17 }   }
18 class Main {
19     public static void main(String[] args) {
20         B x = new A();
21         x.m(4.5);
22 }   }
```

Listing 1 – Un programme avec des erreurs

Exercice 3 : StringReader

Un `StringReader` permet de lire des caractères depuis une chaîne de caractères. Le listing 2 en donne un exemple d'utilisation.

```
1  import org.junit.*;
2  import static org.junit.Assert.*;
3
4  public class StringReaderTest {
5
6      @Test
7      public void testNominal() {
8          StringReader reader = new StringReader("ok");
9          assertEquals('o', reader.read());
10         assertEquals('k', reader.read());
11     }
12
13     @Test(expected=EOS.class)
14     public void testEOS() {
15         StringReader reader = new StringReader("ok");
16         reader.read();
17         reader.read();
18         reader.read();
19     }
20
21     @Test(expected=EOS.class)
22     public void testLimite() {
23         StringReader reader = new StringReader("");
24         reader.read();
25     }
26 }
```

Listing 2 – La classe `StringReaderTest`

3.1. Que signifie `@Test` ?
3.2. Que signifie `expected=EOS.class` ?
3.3. Les deux premières méthodes commencent par la même instruction. Comment la factoriser ?
3.4. Écrire la classe `StringReader`. La classe `String` définit les méthodes `charAt(int)` et `length()`.

Exercice 4 : Première architecture

La figure 1 présente une première architecture pour les fonctionnalités dont nous souhaitons doter nos lecteurs. On veut pouvoir mettre en majuscule les caractères obtenus de la source (`UpperCaseStringReader`), obtenir le numéro de ligne et de colonne du dernier caractère obtenu (`LineNumberStringReader`), décoder les caractères lus (`DecoderStringReader`)...
4.1. Écrire la classe `UpperCaseStringReader`. On utilisera la méthode de classe `toUpperCase(char)` de la classe `Character`.
4.2. Cette architecture permet-elle de combiner deux fonctionnalités, par exemple la mise en majuscule des caractères et l'accès aux numéros de ligne et colonne ? La réponse doit être justifiée.
4.3. Les caractères peuvent venir d'un fichier. Comment faire pour pouvoir lire (via un `Reader` bien sûr) les caractères depuis un fichier ?

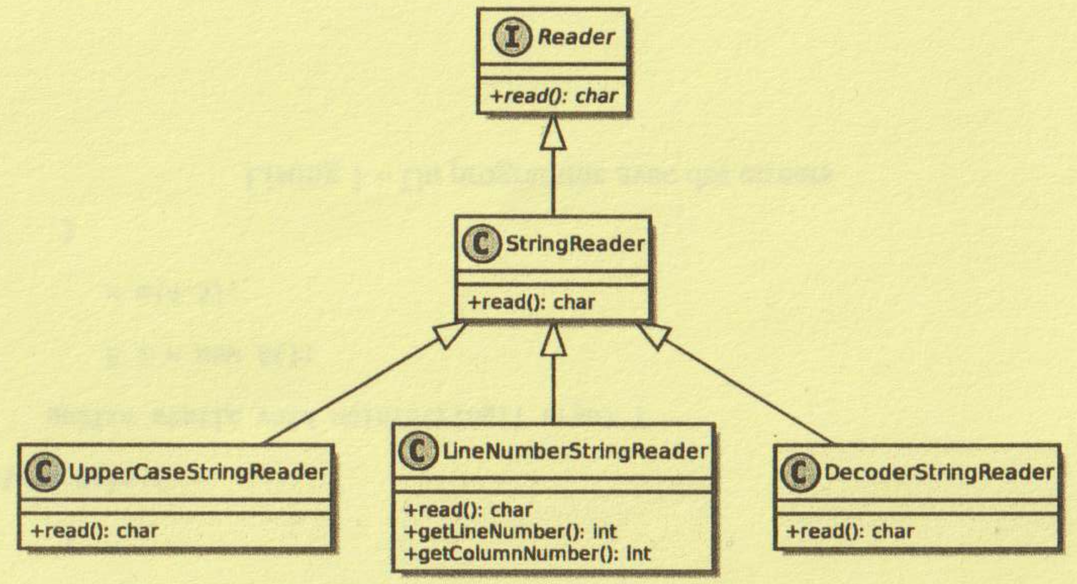

FIGURE 1 – Première architecture

4.4. Les fonctionnalités précédentes (mise en majuscules, accès aux numéros...) peuvent-elles être utilisées avec un lecteur depuis fichier avec cette architecture ? La réponse doit être argumentée.

Exercice 5 : Meilleure architecture

On constate qu'il nous faut distinguer les fonctionnalités à appliquer à un lecteur d'une part et les lecteurs pour une source donnée d'autre part. Le principe est de définir un filtre (`ReaderFilter`) qui est un `Reader` qui lit des caractères d'un lecteur et implante une fonctionnalité. Le filtre par défaut n'implante aucune fonctionnalité et renvoie donc directement le caractère lu. Les fonctionnalités envisagées seront donc des sous-types de ce filtre.
5.1. Dessiner le diagramme de classe de cette nouvelle architecture.
5.2. Expliquer pourquoi on doit faire de `ReaderFilter` une classe et pourquoi elle doit être abstraite même si on sait coder toutes ses méthodes.
5.3. Écrire la classe `ReaderFilter` et la fonctionnalité qui met les caractères en majuscule.
5.4. Depuis le fichier `f1`, on veut mettre les caractères en majuscules et avoir accès aux numéros de ligne. Quels sont les objets à créer ?

Exercice 6 : Décodeur

La fonctionnalité « décodeur » permet de remplacer certains caractères par d'autres ('@' par 'a', '1' par 'i', '3' par 'e', etc.). Ainsi, si la source contient "X@v13r", le résultat du décodeur sera "Xavier" (si on concatène les caractères).
6.1. Quelles est la collection Java adaptée pour conserver les informations de traduction.
6.2. Écrire la classe correspondante en considérant que les informations de traduction sont celles de ce sujet.