TP de Compléments en Programmation Orientée Objet n° 3 bis

Exercice 1 : Transtypages primitifs (déjà dans le TP3)

Voici un programme (`TranstypagesPrimitifs.java` sur Moodle) :

```
1  public class TranstypagesPrimitifs {
2      public static void main(String[] args) {
3          int vint = 1234567891;
4          short vshort = 42;
5          float vfloat = 9.2E11f;
6          System.out.println("vint = " + vint +
7              ", vshort = " + vshort +
8              ", vfloat = " + vfloat);
9
10     }
11 }
```

1. Compilez et exécutez ce programme (assurez-vous de comprendre la notation `9.2E11f`).
2. Nous allons regarder superficiellement le code-octet produit : dans un terminal, allez dans le répertoire où se trouve `TranstypagesPrimitifs.class` et tapez la commande "`javap -c -v TranstypagesPrimitifs`". Le code-octet apparaît ainsi sous une forme désassemblée quasi lisible. Nous nous intéresserons en particulier au début de la partie `Code :`, qui correspond à la déclaration et l'initialisation de nos trois variables. On peut repérer l'appel à l'instruction suivante, `println`, par l'instruction **getstatic** dans le code-octet.

   Il n'y a donc que 6 ou 7 lignes à regarder. Constatez que certaines variables sont initialisées par une séquence d'instructions comme : **bipush** `42;` **istore_2**, alors que d'autres ont la séquence **ldc** suivie de **istore** ou **fstore** (le i ou le f désigne clairement un type)
3. Nous allons nous intéresser à la façon dont sont fait les transtypages. Ajoutez une ligne avant l'instruction d'affichage : `vint=vshort;` et interpréter les opérations `load,store` qui apparaissent.

   Avec les 3 variables présentes il y a théoriquement 6 transtypages, certains qu'il faut rendre explicites. Essayez les tous et complétez le tableau ci-dessous avec vos remarques. Notamment :
   - Est ce que ça compile directement, faut-il ajouter un *cast* explicite etc
   - Quelle est la nature des instructions ajoutées dans le code-octet. (notez que les instructions de la forme `f2i` expriment un changement de type)
   - Quel est l'affichage produit après conversion

=	vint	vshort	vfloat
vint	XXX		
vshort		XXX	
vfloat			XXX

4. Vous pouvez regarder (sans vous attarder) le code-octet correspondant au premier exercice.
5. Faites le même travail sur le programme suivant. Remarquez les instructions qui correspondent au boxing et à la vérification de types.

```
1  public class TranstypagesMixtes {
2      public static void main(String[] args) {
3          Object vObject = Integer.valueOf(9);
4          Integer vInteger = 42;
5          int vint = 111;
```

```
6           System.out.println("vObject = " + vObject +
7               ", vInteger = " + vInteger +
8               ", vint = " + vint);
9
10      }
11  }
```

Exercice 2 : Transtypages d'objets (sur machine)

Même exercice que le précédent mais sur le programme suivant :

```
1   public class TranstypagesObjets {
2       public static void main(String[] args) {
3           Object vObject = new Object();
4           Integer vInteger = 42;
5           String vString = "coucou";
6           System.out.println("vObject = " + vObject + ", vInteger = "
7               + vInteger + ", vString = " + vString);
8       }
9   }
```

Différence, vous ne verrez plus l'ajout de l'instruction **u2t** mais parfois celle de **checkcast**. Dans quels cas ?

Dans certains cas vous aurez eu besoin, pour compiler, d'un *cast* explicite. Lesquels ? Est-ce les-mêmes que dans la question précédente ?

Dans certains cas, le programme quittera sur `ClassCastException`, lesquels ?

Exercice 3 : Transtypages mixtes (sur machine)

Même exercice que le précédent sur le programme suivant :

```
1   public class TranstypagesMixtes {
2       public static void main(String[] args) {
3           Object vObject = Integer.valueOf(9);
4           Integer vInteger = 42;
5           int vint = 111;
6           System.out.println("vObject = " + vObject +
7               ", vInteger = " + vInteger +
8               ", vint = " + vint);
9
10      }
11  }
```

Mêmes questions sachant que d'autres instructions peuvent être insérées par le compilateur dans le code-octet.