POOA REFLEXIVITE

28/09/2017 – INGOUF Camille

Question 1 :

La méthode toString() de Object est héritée par TOUTES les classes. Elle est appelée automatiquement par Java lorsque l'on essaye d'afficher un objet instancié dans une console ou autre.

Elle peut être redéfinie pour que l'affichage corresponde à ce que le développeur souhaite.

Exemple : System.out.println(monObjet) ; //La méthode tostring() est appelée implicitement sur monObjet

Question 2-3 :

La méthode toString() nous apprend que les éléments 0 1 et 3 n'ont pas leur méthode toString de redéfinie, car Java nous donne un affichage par défaut. Mais l'élément 2 nous affiche « Mouhahahaha » ce qui signifie que le développeur a bien redéfini cette méthode pour cette classe.

Question 4 :

Les classes Java ne sont pas des objets. Mais toutes les classes Java ont une instance de java.lang.Class qui permet de les décrire. Cet instance est un objet.

Question 5 :

Grâce à cette description, on peut obtenir toutes les méthodes d'une classe, avec leur signature et leur valeur de retour si elles en ont une. On peut aussi obtenir ces attributs, sa SuperClass directe, ses Interfaces directes, son nom, son package….En gros TOUTES les infos disponibles quand on a accès au code source de la classe.

Question 6 :

```
Class cl = o.getClass();
Method met[] = cl.getDeclaredMethods();
Field fields[] = cl.getDeclaredFields();
Constructor consts[] = cl.getDeclaredConstructors();
```

La méthode « caribou » du deuxième élément affiche « mooooo le caribou »

Question 7 :

Le méthode getDeclaredFields() donne TOUS les attributs d'une classe, contrairement à la méthode getFields() qui ne donne que les attributs publiques.

Cela pose un certain problème de sécurité car certains de ces attributs pourraient contenir des infos secrètes ou sensibles dans leur nom.

Question 8 :

Dans un certain cas où un programme reçoit des objets , il peut exécuter les méthodes des objets qu'il reçoit sans les connaître d'avance.

Autre exemple, je peux savoir si une classe est final ou pas, puis ensuite instancier dynamiquement une classe est héritant de la première, si celle-ci n'est pas final.

Question 9 :

La classe Class est final pour ne pas qu'on puisse pas la faire hériter et la redéfinir. Cela apporte une certaine sécurité.

Question 10 :

```
for(Field f : fields)
		{
			System.out.println(f.getName());
			System.out.println(f.get(o));
			if(f.getType().isPrimitive())
			{
				System.out.println(f.getType());
				f.setInt(o, 33);
				System.out.println(f.get(o));
			}

		}
```

Question 11 :

Cette méthode permet de rendre accessible ou non des attributs de classe. Il est possible de rendre accessible des attributs privés puis ensuite de les voir et les modifier.

Question 12 :

Si un logiciel est mal conçu, il est tout à fait possible pour quelqu'un ayant accès aux classes compilées, de modifier ou d'avoir accès à des données sensibles ou de faire planter le programme complètement, dans une entreprise par exemple.