Licence 3
Juin 2013

Première Session (2h00)

Aucun document autorisé

Soignez la présentation de votre copie (Illisible => Pas lu)
Les téléphones portables doivent être éteints, même si vous n'avez pas d'autre moyen de lire l'heure...

Partie I : Questions de cours et compréhension de code

Question 1 :

Sur votre feuille, écrivez la (ou les) bonne(s) réponse(s).

```
public class Entier {
	private int i;
	public Entier(int i) {
		this.i = i;
	}
	public void setVal(int i) {
		this.i = i;
	}
	public int getVal() {
		return i;
	}
	public String toString() {
		return i + "";
	}
}
public class Main {
	public static void main(String[] args) {
		Entier i = new Entier(1);
		Entier j = i;
		j.setVal(2);
		System.out.println(i);
	}
}
```

Réponses possibles :
A. L'exécution de Main affiche 1
B. L'exécution de Main affiche 2
C. L'exécution de Main affiche autre chose que 1 ou 2.

Question 2 :

Sur votre feuille, écrivez la (ou les) bonne(s) réponse(s).

```
public class A {
	public void m() {
		System.out.println("A");
	}
}
public class B extends A {
	public void m() {
		System.out.println("B");
	}
}
public class Main {
	public static void main(String[] args) {
		A b = new B();
		((A)b).m();
	}
}
```

Réponses possibles :
A. L'exécution de Main affiche A.
B. L'exécution de Main affiche B.
C. Ce code ne compile pas (indiquez la ligne en cause).
D. Ce code compile mais provoque une erreur à l'exécution (indiquez la ligne en cause).

Question 3 :

Sur votre feuille, écrivez la (ou les) bonne(s) réponse(s).

```
public abstract class A {
	public abstract int m();
	public int n() {
		return 2 * m();
	}
}
```

Réponses possibles :
A. La méthode n doit être déclarée abstraite.
B. La méthode n ne doit pas être déclarée abstraite.
C. On peut choisir de déclarer la méthode n abstraite ou pas abstraite. Les 2 solutions sont bonnes.

Question 4 :

Expliquez le fonctionnement du Garbage Collector.

Question 5 :

Quels résultats fournit le programme suivant ?

```
class Except extends Exception {}

public class FinReth {

	public static void f(int n) throws Except {
		try {
			if(n!=1) throw new Except();
		}
		catch(Except e) {
			System.out.println("catch dans f avec n = " + n);
			throw e;
		}
		finally {
			System.out.println("dans finally de f avec n = " + n);
		}
	}

	public static void main(String args[]) {
		int n = 0;
		try {
			for(n=1;n<5;n++)
				f(n);
		}
		catch(Except e) {
			System.out.println("catch dans main avec n = " + n);
		}
		finally {
			System.out.println("dans finally de main avec n = " + n);
		}
	}
}
```

Question 6 :

Quelles sont les deux façons de créer des processus légers en Java ? Donnez un exemple pour chaque cas.

Question 7 :

A quoi sert le mot-clé `synchronized` ? Donnez un exemple.

Question 8 :
As an example for a generic class we will use a very simple container. A Basket can contain only one element.
Here the source code:

```
public class Basket<E> {
        private E element;

        public void setElement(E x) {
                element = x;
        }
        public E getElement() {
                return element;
        }
}
```

We will store fruits in the baskets:

```
class Fruit {
}
class Apple extends Fruit {
}
class Orange extends Fruit {
}
```

8.1) What would Java 1.5 do with the following source code?

```
Basket<Fruit> basket = new Basket<Fruit>(); // 1
basket.setElement(new Apple()); // 2
Apple apple = basket.getElement(); // 3
```

a) The source code is OK. Neither the compiler will complain, nor an exception during the runtime will be thrown.
b) Compile error in the line 2.
c) Compile error in the line 3.
Explain in english !!!

8.2) Let's stay with our baskets. What do you think about the following source code?

```
Basket<Fruit> basket = new Basket<Fruit>();
basket.setElement(new Apple());
Orange orange = (Orange) basket.getElement();
```

a) The source code is OK. Neither the compiler will complain, nor an exception during the runtime will be thrown.
b) Compile error in the line 2.
c) Compile error in the line 3.
d) A **ClassCastException** will be thrown in the line 3.
Explain in english !!!

Partie II : Serveur

Écrire une application client serveur qui cumule dans un même fichier texte sur le serveur les chaînes de caractères envoyées par les différents clients. On ne demande pas l'écriture du client on se contentera d'utiliser telnet.

Question 1 :
Expliquez la démarche et les problèmes à prendre en compte.

Question 2 :
Donnez le code nécessaire.

Partie III : Course automobile

Dans cette partie, on s'intéresse à la réalisation d'une application permettant de gérer une course automobile par équipes. Chaque équipage compte deux pilotes et une liste des temps réalisés pour chaque tour de piste.

Question 1 :
Un temps s'exprime en minutes, secondes, millisecondes. Écrire la classe **Temps** avec son constructeur, ses méthodes d'accès en lecture et la redéfinition de la méthode `toString`.

Question 2 :
Un pilote se caractérise par son nom et son prénom qui sont obligatoirement connus à la création. Les pilotes possèdent un ordre naturel (qui permet de comparer deux pilotes, et pas des choux avec des carottes) qui correspond à l'ordre alphabétique de leur nom. On fera l'hypothèse que deux pilotes ne peuvent avoir le même nom. Vous veillerez à respecter la consistance avec `equals`.
Écrire la classe **Pilote**.

Question 3 :
Écrire la classe **Equipage** avec son constructeur qui prend deux pilotes en argument et une méthode d'instance `addTemps` qui ajoute un temps passé en paramètre à la liste des temps de l'équipage.

Question 4 :
On veut pouvoir trier les temps réalisés par un équipage selon l'ordre croissant ou décroissant. Écrire les deux comparateurs nécessaires.

Question 5 :
Ajouter à la classe **Equipage**, les méthodes d'instance `listeTempsCroissant` et `listeTempsDecroissant` qui retournent une copie de la liste des temps de l'équipage triée respectivement selon l'ordre croissant et décroissant.

Question 6 :
Ajouter à la classe **Equipage**, les méthodes d'instance `meilleurTemps` et `pireTemps` qui retournent respectivement le meilleur temps et le moins bon réalisé par l'équipage.

Question 7 :
Ajouter à la classe **Equipage**, la méthode d'instance `tempsTotal` qui retourne la somme de tous les temps de la liste.

Question 8 :
Écrire la méthode de classe `vainqueur` qui prend un liste d'équipage en paramètre et retourne le plus rapide (en temps total).

Question 9 :
Écrire la méthode de classe `equipageMeilleurTemps` qui prend un liste d'équipage en paramètre et retourne celui ayant réalisé le meilleur temps au tour.

Annexe

Class java.utils.Collections

static <T> T min(Collection<? extends T> coll, Comparator<? super T> comp)
Returns the minimum element of the given collection, according to the order induced by the specified comparator.

static <T> T max(Collection<? extends T> coll, Comparator<? super T> comp)
Returns the maximum element of the given collection, according to the order induced by the specified comparator.

static <T> void sort(List<T> list, Comparator<? super T> c)
Sorts the specified list according to the order induced by the specified comparator.