Interrogation écrite - Groupe G

Durée : 1h (tiers temps : 1h20) -- Documents autorisés : polys et notes de cours, programmes des exercices des séances de travaux pratiques

Nom :

Prenom :

Exercice 1: Questions de base (5 pts)

A. Quelle est la différence entre un attribut static et non-static ? (1 pt)

Les attributs static sont des variables globales partagées entre tous les objets de la classe. Les attributs non-static sont spécifiques à chaque objet.

B. À quoi servent les constructeurs de classe en Java ? (1 pt)

Les constructeurs permettent d'initialiser les attributs d'un objet lors de sa création.

C. Combien d'instances de la classe A crée le code suivant ? (1 pt)

```
A x,u,v;
x = new A();
A y = x;
A z = new A();
```

2

D. Quelles sont les valeurs affichées par le programme suivant ? Expliquer pourquoi. (2 pts)

```
static void changeInt(int i) {
      i = i + 1;
}

static void changeIntArray(int[] arr) {
      arr[0] = arr[0] + 1;
}

public static void main(String[] args) {
      int n = 0;
      int[] vec = new int[1];
```

```
            vec[0] = 0;

            changeInt(n);
            changeIntArray(vec);

            System.out.println(n);
            System.out.println(vec[0]);
      }
```

0 1 : L'entier qui est modifié dans changeInt correspond à une copie de la variable n passée en paramètre.
La fonction changeIntArray utilise la référence passée en paramètre pour aller modifier la valeur d'une des cases du tableau qui se trouve dans le tas et cette modification reste donc une fois que l'on revient au programme main().

Exercice 2: Programme à corriger (5 pts)

Le programme suivant devrait imprimer

```
Pierre
Lucas
Gabriel
```

mais à cause de **deux** erreurs, la sortie n'est pas celle attendue.
Donner ce qu'affiche le programme (1 pt) et expliquer quelles sont les erreurs (2 pts).
Ensuite, corriger le programme pour qu'il affiche Pierre Lucas Gabriel (2 pts).

```
public class Personne {
    static String nom;
    Personne (String nom) {
       nom = nom;
    }
    public static void main(String[] args) {
       Personne[] l = new Personne[3];
       l[0] = new Personne("Pierre");
       l[1] = new Personne("Lucas");
       l[2] = new Personne("Gabriel");

       for (Personne p : l) {
            System.out.println(p.nom);
       }
    }
}
```

Le programme imprime null null null. Comme chaque personne a un nom différent, l'attribut ne doit pas être static. De plus, comme le constructeur n'utilise pas le mot clé this, l'affectation nom = nom ne change pas l'attribut de la classe car le paramètre a le même nom que l'attribut. Le code suivant produit la sortie attendue.

```
public class Personne {
    String nom;
    Personne (String nom) {
      this.nom = nom;
    }
    public static void main(String[] args) {
      Personne[] l = new Personne[3];
      l[0] = new Personne("Pierre");
      l[1] = new Personne("Lucas");
      l[2] = new Personne("Gabriel");

      for (Personne p : l) {
           System.out.println(p.nom);
      }
    }
}
```

Exercice 3: Programme à écrire (10 pts)

Dans cet exercice, on modélisera des avions et des compagnies aériennes à travers les classes suivantes (à être complétées).

```
public class Avion {
    // à remplir
}

public class CompagnieAerienne {
    static int MAX_NB_AVIONS = 10;
    Avion[] avions;
    int nb_avions;
    String nom;

    CompagnieAerienne(String nom){
      // à remplir
    }
    void ajouterAvion(Avion a) {
      // à remplir
    }
```

```
    void afficherListeAvions() {
      // à remplir
    }
    void acheterCompagnie(CompagnieAerienne ca) {
      // à remplir
    }
    public static void main(String[] args) {
      CompagnieAerienne airParis = new CompagnieAerienne("Air
Paris");
      CompagnieAerienne gammaAirlines = new CompagnieAerienne("Gamma
Airlines");

      airParis.ajouterAvion(new Avion(135, "AirBoeing 474"));
      airParis.ajouterAvion(new Avion(279, "AirBoeing 474"));

      gammaAirlines.ajouterAvion(new Avion(29, "AirBoeing 373"));
      gammaAirlines.ajouterAvion(new Avion(321, "AirBoeing 373"));

      airParis.afficherListeAvions();
      gammaAirlines.afficherListeAvions();

      airParis.acheterCompagnie(gammaAirlines);
      airParis.afficherListeAvions();

    }
}
```

A. Un avion est caractérisé par un identifiant du type `int` et un modèle du type `String`. Compléter la classe `Avion` avec ces attributs, un constructeur qui prend des valeurs pour ces attributs en paramètre et une méthode d'affichage. (2 pts)
B. Une compagnie aérienne est toujours créée avec un nom et sans aucun avion. De plus, la loi interdit à une compagnie d'avoir plus de 10 avions. En utilisant ces informations, compléter le constructeur `CompagnieAerienne(String nom)`. (2 pts)
C. Compléter la méthode `ajouterAvion`, utilisée par les compagnies aériennes lors de l'acquisition d'un avion. (1 pt)
D. Compléter la méthode `afficheListeAvions`, utilisée par les compagnies aériennes pour voir leurs flottes d'avions. On utilisera la méthode d'affichage d'avion programmée dans la partie A. (2 pts)
E. Lorsque les compagnies A et B ont ensemble moins de 10 avions, la loi permet à A d'acheter la compagnie B et tous ses avions. On appellera alors

A.`acheterCompagnie` (B). Compléter le code de la méthode `acheterCompagnie`. (2 pts)

F. Quel est l'affichage produit par la méthode main ? (1 pt)

```
public class Avion {
    int id;
    String modele;
    Avion(int id, String modele){
      this.id = id;
      this.modele = modele;
    }
    void afficheAvion() {
      System.out.println("id: " + id);
      System.out.println("modele: " + modele);
    }
}

public class CompagnieAerienne {
    static int MAX_NB_AVIONS = 10;
    Avion[] avions;
    int nb_avions;
    String nom;

    CompagnieAerienne(String nom){
      this.nom = nom;
      avions = new Avion[10];
      nb_avions = 0;
    }

    void ajouterAvion(Avion a) {
      If (nb_avions == MAX_NB_AVIONS) return;
      avions[nb_avions] = a;
      nb_avions = nb_avions + 1;
    }

    void afficherListeAvions() {
      System.out.println(nom);
      for (int i = 0; i < nb_avions; i++) {
           avions[i].afficheAvion();
      }
    }

    void acheterCompagnie(CompagnieAerienne ca) {
      if (nb_avions + ca.nb_avions > 10) return;
```

```
      for (int i = 0; i < ca.nb_avions; i++) {
           ajouterAvion(ca.avions[i]);
      }
    }


    public static void main(String[] args) {
      CompagnieAerienne airParis = new CompagnieAerienne("Air
Paris");
      CompagnieAerienne gammaAirlines = new CompagnieAerienne("Gamma
Airlines");

      airParis.ajouterAvion(new Avion(135, "AirBoeing 474"));
      airParis.ajouterAvion(new Avion(279, "AirBoeing 474"));

      gammaAirlines.ajouterAvion(new Avion(29, "AirBoeing 373"));
      gammaAirlines.ajouterAvion(new Avion(321, "AirBoeing 373"));

      airParis.afficherListeAvions();
      gammaAirlines.afficherListeAvions();

      airParis.acheterCompagnie(gammaAirlines);
      airParis.afficherListeAvions();

    }

}
```

La sortie produite est la suivante.

```
Air Paris
id: 135
modele: AirBoeing 474
id: 279
modele: AirBoeing 474
Gamma Airlines
id: 29
modele: AirBoeing 373
id: 321
modele: AirBoeing 373
Air Paris
id: 135
modele: AirBoeing 474
id: 279
modele: AirBoeing 474
```

```
id: 29
modele: AirBoeing 373
id: 321
modele: AirBoeing 373
```