Université A. Belkaïd
Tlemcen
Faculté des Sciences
Département d'Informatique
L2 informatique 2015-2016

Examen de Rattrapage de Programmation Objet

Nom : .....

Prénom : .....

Mercredi 6 avril 2016
Durée : 1h30

Éteignez vos téléphones SVP - Documents non autorisés

Exercice 1 : (10 pts)

Examinez le code Java suivant (constitué de 2 fichiers) :

```
1.  /* fichier Schtroumpfable.java */
2.  public interface Schtroumpfable
3.  {
4.      void Schtroumpfer() {
5.          System.out.println("Je suis schtroumpfé");
6.      }
7.  }
```

```
1.  /* fichier Truc.java */
2.  public abstract class Truc implements Schtroumpfable
3.  {
4.      private String nom;
5.      public Truc(String unNom){
6.          nom = unNom;
7.      }
8.  }
```

1) L'interface Schtroumpfable est-elle correcte ? Si non expliquez pourquoi et réécrivez-la correctement.

2) Dans le code du fichier Truc.java, peut-on enlever le mot-clé **abstract** ? Donnez une réponse détaillée.

3) Écrivez le code Java d'une classe nommée **Machin** concrète (pas abstraite) et qui hérite de la classe Truc. Attention, elle devra être simple mais sans erreurs !

4) On ajoute à notre programme Java la classe **Chose** suivante :

```
1.  /* fichier Chose.java */
2.  public class Chose extends Machin
3.  {
4.      public Chose(String nom)
5.      {
6.          super(nom);
7.      }
8.      public void Schtroumpfer(){
9.          System.out.println(nom + " est une Chose Schtroumpfée");
10.     }
11. }
```

Le compilateur signale une erreur à la ligne 9. Laquelle et pourquoi ? Expliquez comment éviter cette erreur en donnant une solution valide.

5) Une fois les problèmes réglés, nous ajoutons au projet la classe qui nous permettra d'exécuter le programme :

```
1.  /* fichier Bazinga.java */
2.  public class Bazinga
3.  {
4.      public static void main(String[] args){
5.          Schtroumpfable[] tab = new Schtroumpfable[6];
6.          tab[0] = new Chose("c");
7.          tab[1] = new Machin("m");
8.          tab[2] = new Truc("t");
9.          tab[3] = new Schtroumpfable("s");
10.         tab[4] = (Machin) tab[0];
11.         tab[5] = (Chose) tab[1];
12.         for (Schtroumpfable x : tab)
13.             if (x != null)
14.                 x.Schtroumpfer();
15.     }
16. }
```

a) Les instructions de cette classe sont-elles toutes correctes ? Si non, donnez les numéros des lignes à supprimer en expliquant quelle erreur elles provoquent et pourquoi.

b) Donnez le résultat qui sera affiché à l'exécution de cette classe (après suppression des lignes erronées s'il y en a).

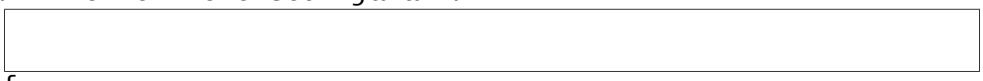
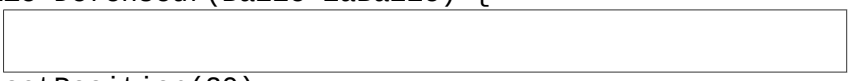
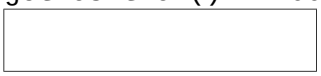
Exercice 2 : (10 pts)

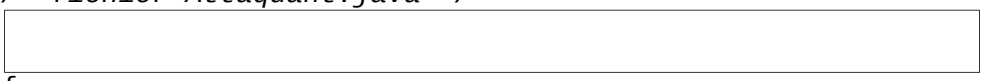
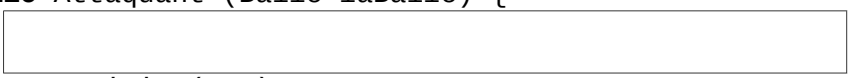
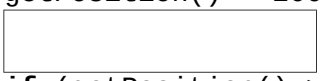
1) Complétez le code Java suivant en remplissant directement les cadres vides *(répondez sur le sujet)* :

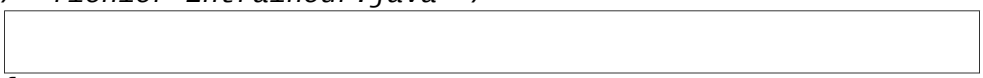
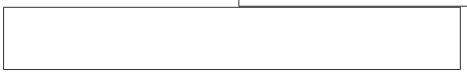
```
/* fichier Balle.java */
public class Balle
{
     public Balle() {
     }
     public void bouge() {
         System.out.println("la balle bouge");
     }
}
```

```
/* fichier Joueur.java */
[                                                    ]
{
     private int posSurLeTerrain;
     private Balle laBalle;
     public Joueur(Balle laBalle) {
          [                          ] = laBalle;
     }
     public int [                                   ]
          [                                         ]
     }
     public void [                                  ]
          [                                         ]
     }
     public void joueLaBalle() {
         System.out.println("Je tape la balle avec le pied");
         laBalle.bouge();
     }
     public String toString() {
         return getClass().getName() ;
     }
     public void avance() {
         System.out.println("Position du " + this + " = " + posSurLeTerrain);
         posSurLeTerrain += 20;
     }
}
```

```
/* fichier Gardien.java */
[                                                    ]
{
    public Gardien(Balle laBalle) {
         [                                           ]
         setPosition(0);
    }
    public void joueLaBalle() {
         [                  ].joueLaBalle();
         System.out.println("Je prends la balle avec les mains");
    }
    public void avance() {
        if (getPosition() < 10)
            System.out.println("Gardien : Je peux prendre la balle avec les mains");
        if (getPosition() < 20)
             [                  ].avance();
    }
}
```

```java
/* fichier Defenseur.java */

{
    public Defenseur(Balle laBalle) {
        
        setPosition(20);
    }
    public void avance() {
        if (getPosition() < 100)
             .avance();
    }
}
```

```java
/* fichier Attaquant.java */

{
    public Attaquant (Balle laBalle) {
        
        setPosition(100);
    }
    public void avance() {
        if (getPosition() < 200) {
             .avance();
            if (getPosition() > 150)
                System.out.println("Attaquant : Je fais attention au hors-jeu");
        }
    }
    public void marqueUnBut() {
        System.out.println("Youpiiiii..... j'ai marqué... !!");
    }
}
```

```java
/* fichier Entraineur.java */

{
    private 
    public Entraineur(  lesJoueurs){
         = lesJoueurs;
    }
    public void panique(){
        System.out.println("C'est la panique");
        for (  )  // boucle for-each
            j.avance();
    }
}
```

```
1.  /* fichier Football.java */
2.  public class Football
3.  {
4.      public static void main(String[] args) {
5.          Balle uneBalle =
6.          Joueur[] lesJoueurs = new Joueur[3];
7.          lesJoueurs[0] = new Gardien(uneBalle);
8.          lesJoueurs[1] = new Defenseur(uneBalle);
9.          lesJoueurs[2] = new Attaquant(uneBalle);
10.         Entraineur unEntraineur = new Entraineur(lesJoueurs);
11.         System.out.println("******* d'abord les joueurs *****");
12.         for (                                    )  // boucle for-each
13.             j.joueLaBalle();
14.         System.out.println("******* puis l'entraineur *****");
15.         for (int i=0; i<6; i++)
16.             unEntraineur.panique();
17.
18.     }
19. }
```

2) Donnez le résultat affiché après l'exécution de ce programme.

3) Réécrivez les lignes 12 et 13 du fichier Football.java en remplaçant la boucle for-each par une boucle for (avec un compteur *i* entier).

4) À la ligne 6 du fichier Football.java, un *tableau* est utilisé. On veut remplacer ce tableau par une *collection paramétrée* de type ArrayList. Indiquez quelles lignes de quel(s) fichier(s) il faudra modifier et réécrivez-les, en prenant en compte les modifications faites dans la question 3 précédente.

5) Dans le fichier Football.java de la question 1 *(sans les modifications de la 4)*, on ajoute à la ligne 17 l'instruction suivante :

```
17.             LesJoueurs[2].marqueUnBut();
```

a) Cette instruction est incorrecte. Provoque-t-elle une erreur à la compilation ou à l'exécution ?
b) Réécrivez-la correctement pour qu'elle ne provoque plus d'erreur et puisse s'exécuter.

6) En vue d'améliorer notre programme, nous appliquons un *patron de conception* (*design pattern*) à la classe Balle. Voici la nouvelle version du fichier Balle.java :

```
1.  /* fichier Balle.java */
2.  public class Balle
3.  {
4.      private static Balle balle = null;
5.      private Balle() {
6.      }
7.      public static Balle getBalle() {
8.          if (balle == null)
9.              balle = new Balle();
10.         return balle;
11.     }
12.     public void bouge() {
13.         System.out.println("la balle bouge");
14.     }
15. }
```

a) Faudra-t-il faire des modifications dans les autres fichiers pour utiliser cette nouvelle version de Balle ? Si oui, lesquelles ?
b) Quelle est l'utilité de ce patron de conception (à quoi sert cette modification de la classe Balle) ?