Examen OBJET

Nom : Prénom :

DS OBJET

Durée : 2h – Tous documents autorisés

Cet examen comporte deux parties : une partie de questions de cours et une partie de réflexion et de modélisation UML sur une application imaginaire. Nous vous conseillons de lire l'intégralité du sujet avant de débuter. NB : le barème est indicatif ! (et volontairement sur 21 points)

RENDEZ LE SUJET AVEC VOTRE COPIE !

Questions de cours

Question 1 : Notions Java (3 points)

Décrivez brièvement ce que sont :

- Le surclassement en Java
- Le polymorphisme en Java
- La notion d'interface en Java

Question 2 : Exceptions (1 point)

Étant donné le code Java suivant (voir Figure 1) : quel est le résultat d'exécution de la méthode `main` de la classe `WhatAMess` ?

```
public class WhatAMess {

    public static void main(String[] args) {
        System.out.println("1");

        try {
            System.out.println("2");
            if(true) {
                throw new Exception();
            }
            System.out.println("3");
        } catch(Exception e1) {
            try {
                System.out.println("4");
                if(true){
                    throw new Exception();
                }
                System.out.println("5");
            } catch(Exception e2) {
                try {
                    System.out.println("6");
                    if(false) {
                        throw new Exception();
                    }
                    System.out.println("7");
                } catch(Exception e3) {
                    System.out.println("8");
                } finally {
                    System.out.println("9");
                }
                System.out.println("10");
            }
        }
        finally{
            System.out.println("11");
        }

    }
}
```

Figure 1 : Classe WhatAMess

Question 3 (1.5 points) :

Corrigez le code Java de la Figure 2 pour qu'il compile. Décrivez les changements à faire en précisant bien les lignes concernées et les modifications que vous proposez.

Une fois les modifications effectuées, donnez le résultat de l'exécution de ce programme si l'utilisateur rentre le **chiffre 6**.

```
4   public class TestMyClass {
5
6       public int fibo(int n) {
7           int res;
8           int fibo1 = 1;
9           int fibo2 = 1;
10
11          res = 1;
12          if(n<= 0) { res = 0; }
13          if(n == 1 || n == 2) { res = 1; }
14          for(int i=3; i<= n;i++) {
15              res = fibo1 + fibo2;
16              fibo1 = fibo2;
17              fibo2 = res;
18          }
19          return res;
20      }
21
22      public static void main(String[] args) {
23          System.out.println("Entrez un nombre : ");
24          Scanner sc = new Scanner(System.in);
25          int fiboNb = sc.nextInt();
26          System.out.println("Fibo("+fiboNb+") = "+fibo(fiboNb));
27
28      }
29  }
```

Figure 2 : Classe TestMyClass

Nom : Prénom :

Question 4 (4 points) :

En vous basant sur le code des classes `Truc`, `Machin` (voir Figure 3) remplissez le tableau suivant lorsque l'instruction proposée est insérée à la ligne 28 de la classe `Test`.

Instruction proposée	Erreur compilation	Erreur exécution	OK (compilation et exécution)	Affichage résultat (si OK)
`x.tutut(y) ;`	☐	☐	☐	
`y.tutut(y);`	☐	☐	☐	
`x.tutut(z);`	☐	☐	☐	
`z.tutut(y);`	☐	☐	☐	
`x.vroumvroum(z);`	☐	☐	☐	
`x.vroumvroum(y);`	☐	☐	☐	
`y.vroumvroum(y);`	☐	☐	☐	
`z.vroumvroum(y);`	☐	☐	☐	

```
1   public class Truc {
2       void tutut(Truc t) {
3           System.out.println("tutut de Truc");
4       }
5
6       void vroumvroum(Truc t){
7           System.out.println("vrouvroum de Truc");
8       }
9   }
10
11
12  public class Machin extends Truc{
13      void tutut(Truc t) {
14          System.out.println("tutut de Machin");
15      }
16
17      void vroumvroum(Machin t){
18          System.out.println("vrouvroum de Machin");
19      }
20  }
21
22  public class Test {
23      public static void main(String[] args){
24          Truc x = new Machin();
25          Machin y = new Machin();
26          Truc z = new Truc();
27
28          // l'instruction est à mettre ici
29      }
30  }
```

Figure 3 : Classes Machin, Truc et Test

Nom : Prénom :

Question 5 (1 point) :

En vous basant sur le code des classes `Point2D` et `Personnage` (Figures 4 et 5), affichez le résultat de l'exécution de la méthode main de la classe `Test1` (Figure 4).

```java
public class Test1 {

    public static void main(String[] args) {
        Point2D p1 = new Point2D(1,1);
        Point2D p2 = new Point2D(2,2);
        Personnage perso1;
        Personnage perso2;

        p2 = p1;
        perso1 = new Personnage(100,10,10,0,0,10,0,p1);
        p1.setX(3);
        perso2 = new Personnage(perso1);
        System.out.println(perso1);
        System.out.println(perso2);
        System.out.println(p1);
        p1.setX(2);
        System.out.println(perso1);
        System.out.println(perso2);
        System.out.println(p2);
    }
}
```

```java
public class Point2D {
    private int x;
    private int y;

    // Constructeurs
    Point2D(int x, int y) {
        this.x = x;
        this.y = y;
    }

    Point2D() {
        this(0,0);
    }

    Point2D(Point2D p) {
        this.x = p.x;
        this.y = p.y;
    }

    // Accesseurs
    public int getX() {
        return this.x;
    }
    public int getY() {
        return this.y;
    }

    // Modificateurs
    public void setX(int x) {
        this.x = x;
    }

    public void setY(int y) {
        this.y = y;
    }

    @Override
    public String toString() {
        String s = "["+this.x+","+this.y+"]";
        return s;
    }
}
```

Figure 4 : Classes Test1 et Point2D

Nom : Prénom :

```java
public class Personnage {
    private int ptVie;
    private int ptAtt;
    private int ptPar;
    private int ptMag;
    private int resistMag;
    private int degAtt;
    private int degMag;
    private Point2D pos;

    // Constructeurs
    public Personnage(int ptVie, int ptAtt,
                      int ptPar, int ptMag,
                      int resistMag, int degAtt,
                      int degMag, Point2D pos) {
        this.ptVie = ptVie;
        this.ptAtt = ptAtt;
        this.ptPar = ptPar;
        this.ptMag = ptMag;
        this.resistMag = resistMag;
        this.degAtt = degAtt;
        this.degMag = degMag;
        this.pos = pos;
    }

    // Constructeur de recopie
    public Personnage(Personnage p) {
        this.ptVie = p.getPtVie();
        this.ptAtt = p.getPtAtt();
        this.ptPar = p.getPtPar();
        this.ptMag = p.getPtMag();
        this.resistMag = p.getResistMag();
        this.degAtt = p.getDegAtt();
        this.degMag = p.getDegMag();
        this.pos = new Point2D(p.getPos());
    }

    // Getters and setters
    public int getPtVie() {
        return ptVie;
    }

    public void setPtVie(int ptVie) {
        this.ptVie = ptVie;
    }

    public int getPtAtt() {
        return ptAtt;
    }

    public void setPtAtt(int ptAtt) {
        this.ptAtt = ptAtt;
    }

    public int getPtPar() {
        return ptPar;
    }

    public void setPtPar(int ptPar) {
        this.ptPar = ptPar;
    }

    public int getPtMag() {
        return ptMag;
    }

    public void setPtMag(int ptMag) {
        this.ptMag = ptMag;
    }

    public int getResistMag() {
        return resistMag;
    }

    public void setResistMag(int resistMag) {
        this.resistMag = resistMag;
    }

    public int getDegAtt() {
        return degAtt;
    }

    public void setDegAtt(int degAtt) {
        this.degAtt = degAtt;
    }

    public int getDegMag() {
        return degMag;
    }

    public void setDegMag(int degMag) {
        this.degMag = degMag;
    }

    public Point2D getPos() {
        return pos;
    }

    public void setPos(Point2D pos) {
        this.pos = pos;
    }

    @Override
    public String toString() {
        String s = "Personnage en position : "+this.pos.toString();
        return s;
    }
}
```

Figure 5 : Classe Personnage

Nom : Prénom :

Question 6 (4.5 points) :

En vous basant sur les exemples fournis dans les vidéos additionnelles au cours, et sur le code suivant (Figure 6), réalisez des schémas de la mémoire (états de la pile et du tas) lors de l'exécution de la méthode `main()` de la classe principale `Main` (Figure 7). Vous ne devez réaliser ces schémas que pour les endroits indiqués par le marqueur suivant :

```
// POINT_X draw memory scheme at this point
```

Vous préciserez également l'affichage obtenu sur la sortie standard (résultats des `System.out.println(…)`) pour l'ensemble de l'exécution de cette application.

```
public class Carambar {

    /**
     * Le gout du Carambar
     */
    String gout;

    public Carambar(String g) {
        this.gout = g;
    }
    public String getGout() {
        return this.gout;
    }
    public void setGout(String newg) {
        this.gout = newg;
    }
    public void affiche() {
        System.out.println(this);
    }

    @Override
    public String toString() {
        String res;
        res="Je suis un carambar gout : "+this.gout;
        return res;
    }

    @Override
    public boolean equals(Object o) {
        if (o != null && o instanceof Carambar) {
            Carambar ing = (Carambar) o;
            return this.gout.equals(ing.getGout());
        } else {
            return false;
        }
    }

}
```

```
ublic class Smarties {
    /**
     * La couleur du Smarties
     */
    private String couleur;

    public Smarties(String couleur) {
        this.couleur = couleur;
    }
    public String getCouleur() {
        return couleur;
    }
    public void setCouleur(String couleur) {
        this.couleur = couleur;
    }
    public void affiche() {
        System.out.println(this);
    }

    @Override
    public String toString() {
        String res;
        res="Smarties de couleur : "+this.couleur;
        return res;
    }

    @Override
    public boolean equals(Object o) {
        if (o != null && o instanceof Smarties) {
            Smarties ing = (Smarties) o;
            return this.couleur.equals(ing.getCouleur());
        } else {
            return false;
        }
    }
```

Figure 6 : Classes Carambar et Smarties

```
public class Main {

    public static void caramb(Carambar c1, Carambar c2, String newgout){
        c1 = c2;
        c2.setGout(newgout);
    }

    public static void smart(Smarties smartA, Smarties smartB, String newcol) {
        smartA.setCouleur(newcol);
        smartB = new Smarties("jaune");
    }

    public static void main(String[] args) {
        // Creation des smarties
        Smarties smartR = new Smarties("rouge");
        Smarties smartB = new Smarties("bleu");

        // Creation des carambars
        Carambar caramCaramel = new Carambar("caramel");
        Carambar caramCitron = new Carambar("citron");

        System.out.println("POINT_1");
        // Affichage des smarties
        smartR.affiche();
        smartB.affiche();

        // Affichage des carambars
        caramCaramel.affiche();
        caramCitron.affiche();

        // POINT_1 draw memory scheme at this point

        // Appel de la methode 'smart'
        smart(smartR,smartB, "jaune");
        System.out.println("POINT_2");
        System.out.println(smartR == smartB);
        System.out.println(smartR.equals(smartB));

        // Affichage des smarties
        smartR.affiche();
        smartB.affiche();

        // POINT_2 draw memory scheme at this point

        // Appel de la methode 'caramb'
        caramb(caramCitron, caramCaramel, "fraise");

        System.out.println("POINT_3");
        System.out.println(caramCitron == caramCaramel);
        System.out.println(caramCitron.equals(caramCaramel));
        // Affichage des carambars
        caramCaramel.affiche();
        caramCitron.affiche();

        // POINT_3 draw memory scheme at this point
    }
}
```

Figure 7 : Classe Main

Étude d'une application imaginaire : Système d'Information (SI) de la société « MyDesignFactory » (7 points)

Dans cette partie, nous vous demandons de vous intéresser à la création du Système d'information minimaliste pour la société « MyDesignFactory » qui fabrique des meubles et produits de décoration d'intérieur design.

NB : cette application et le descriptif qui suit est parfaitement fictif et ne reflète pas ce que l'on vous demanderait pour vraiment créer un système d'information même minimaliste. Nous vous demandons de ne pas en tenir compte et de répondre aux questions demandées sans relever les imprécisions ou les invraisemblances.

Votre client la société « MyDesignFactory » vous a contacté pour développer une nouvelle version minimaliste de leur SI. Ils vous fournissent seulement une description textuelle de leurs attentes et ce sera à vous de proposer des solutions et une implémentation du SI minimaliste.

Le descriptif de la première version fournie par la société « MyDesignFactory » est représenté dans la Figure 8. Assurez-vous bien de lire attentivement ce descriptif avant de répondre aux questions.

Question 7.1 : Diagramme de classes UML (4 points)

Proposez un diagramme de classes et une description brève de vos choix lors de la réalisation de ce diagramme et des relations entre les classes et/ou leurs attributs.

Ne faites apparaître que les attributs que vous jugez utiles et les méthodes les plus importantes (**pas de getters/setters ni tous les constructeurs**).

Si vous voulez représenter des classes ou méthodes abstraites, veuillez adopter une notation facilement lisible (p. ex. en couleur, souligné, etc.) et **précisez quelle notation vous avez choisi !**

Ajoutez à votre diagramme une justification de vos choix (par ex. si vous n'avez pas représenté dans votre diagramme une partie des informations fournies par la société).

Question 7.2 : Magasins (2 points)

La société « MyDesignFactory » est satisfaite de votre proposition, elle vous demande maintenant de pouvoir rajouter à votre SI leurs magasins qui existant un peu partout en France.
Un magasin possède un certain nombre d'articles à vendre qui peuvent ou non être expédiés aux clients. Quand ils ne peuvent pas être expédiés, les clients doivent venir les retirer en magasin.

Modifiez votre diagramme et/ou ajoutez des classes permettant de prendre cette modification en compte. Implémentez le constructeur par défaut de la classe Magasin.

Nous souhaitons une nouvelle version de notre SI, minimale dans un premier temps. Dans cette première version nous souhaitons pouvoir modéliser les différents articles de design (et uniquement les articles) que nous produisons et vendons.

Tous nos articles ont en commun d'avoir une référence (sous forme de chaine de caractères), un prix hors-taxe (HT, nombre réel), un prix toutes taxes comprises (TTC, nombre réel) et un nom/description succincte (chaine de caractères).

Nos articles sont de différents types : meubles, luminaires ou accessoires. Chaque type a plusieurs sous-types :

- Canapé, fauteuil, table, rangement et lit pour les meubles.
- Lampadaires, lampes, abat-jour ou suspension pour les luminaires.
- Tapis, Salle de bains, décoration et textile pour les accessoires.

Enfin chaque sous-type a plusieurs modèles, par exemple il existe un modèle de canapé « Rochester » qui peut avoir plusieurs couleurs différentes et existe en deux tailles différentes : 3 ou 5 personnes. Tous les modèles de tous les sous-types ont les mêmes caractéristiques, par exemple tous les modèles de canapés sont caractérisés par une couleur et une taille (décrite comme un nombre de personnes pouvant s'assoir sur le canapé).

Certains articles peuvent être expédiés et d'autres non car ils sont trop volumineux. Pour les articles ne pouvant pas être expédiés les clients doivent venir les chercher en magasin. Les articles ne pouvant être expédiés sont les suivants :

- Canapés,
- Tables,
- Tapis,
- Lit,
- Lampadaires.

Les articles expédiés ont des frais de livraison qui seront ajoutés au prix TTC. Les articles les plus chers (prix TTC > 1000€) peuvent être payés en 3 fois sans frais.

Figure 8 : Description fournie par "MyDesignFactory" sur leur mode de fonctionnement.

Question 7.3 : Limites ? (1 point)

Quelles sont les limites au schéma que vous proposez, et comment pourriez-vous les résoudre.