



## Examen de Langage Java

Nom :

Prénom :

Groupe :

Professeur : BDR – CLR – MBA – MCD – PBT (entourez l'acronyme de votre professeur)

Partie 1	Partie 2	Partie 2	Partie 4	Partie 5	Total	Cote
/10	/24	/8	/4	/4	/50	<b>/20</b>

### DIRECTIVES:

- *L'examen se déroule sans notes*
- *Vous avez 2 heures*
- *Ne détachez pas les feuilles de l'énoncé*
- *Écrivez vos nom, prénom et groupe sur chacune des feuilles qu'on vous distribue*
- *Répondez sur la feuille de papier "ministre" uniquement, toutes les autres feuilles (ainsi que l'énoncé) peuvent servir de brouillon*
- *Numérotez clairement vos réponses, comme elles le sont dans l'énoncé.*  
*Vous pouvez répondre aux questions dans l'ordre que vous voulez*
- *La clarté de vos réponses, le soin et l'attention portée aux conventions d'écriture données en cours seront pris en compte dans la notation.*

### Première partie (10 pts)

- 1.1 (1 pt) Un tableau créé dans une fonction peut être retourné comme valeur de retour de cette fonction. Vrai ou Faux ?
- 1.2 (1 pt) Une fonction dont le type de retour est `void` peut retourner n'importe quel type de valeur. Vrai ou Faux ?
- 1.3 (1 pt) Que va afficher le programme suivant ?

```
public class Interro {
    public static int div2Plus1 (int nb) {
        nb = nb / 2;
        return nb+1;
    }
    public static void main(String[] args) {
        int nb=3;
        div2Plus1(nb);
        System.out.println(nb);
    }
}
```

1.4 (1 pt) Soit la déclaration suivante :  
`int[][] tab = {{1,2},{2}};`  
Quel est le type précis de `tab[1]` ?

1.5 (1 pt) Quelle est la déclaration **correcte** parmi les suivantes ?  
A) `String[][] motss = {"chien"} {"dog"};`  
B) `String[] motss = new String[2][];`  
C) `String[][] motss = new String[][4];`  
D) `String[][] motss = new String[][] {"chien"}, {"dog"};`

1.6 (1 pt) Combien de fois le programme (compilable) suivant va-t-il afficher "Aie" ?

```
public class Interro {
    public static void main(String[] args) {
        for(int i=0; i<5; i=i+1)
            for(int j=i; j>2; j=j-1)
                System.out.println("Aie");
    }
}
```

1.7 (1 pt) Que va afficher le programme (compilable) suivant ?

```
public class Interro {
    public static void main(String[] args) {
        int a = 10, b = 3, c = 0 ;
        System.out.print(1);
        if ( a==b )
            if ( c!=0 )
                System.out.print(2);
        else
            System.out.print(3);
        System.out.print(4);
    }
}
```

1.8 (1 pt) Que va afficher le programme (compilable) suivant ?

```
public class Interro {
    public static void main(String[] args) {
        int i = 0;
        for (i = 5 ; i < 7 ; i = i+1)
            System.out.print (i + " ");
            System.out.print (i + " ");
    }
}
```

1.9 (1 pt) Que vaut l'expression: `3 + 2 + "1" + 2 + 3`

1.10 (1 pt) Que va afficher le programme (compilable) suivant ?

```
public class Interro {
    public static void main(String[] args) {
        double monDouble = 15/4 ;
        System.out.println (monDouble);
    }
}
```

## Deuxième partie (24 pts)

Soit la classe *Animal* suivante.

```
public class Animal {
    private String espèce;
    private boolean femelle;

    public String getEspèce(){
        return espèce;
    }
    public void setEspèce(String e){
        espèce = e;
    }
    public boolean isFemelle(){
        return femelle;
    }
    public void setFemelle(boolean f){
        femelle = f;
    }
    public String toString(){
        String aRetourner = "Espèce : " + espèce;
        if (femelle)
            aRetourner += " femelle";
        else
            aRetourner += " mâle";
        return aRetourner;
    }
}
```

- 2.1 (1 pt) Récrivez la méthode `toString()` en utilisant l'opérateur ternaire ? :
- 2.2 (2 pts) Soit la classe *Test1* suivante. Est-elle correcte ? Si oui, donnez l'affichage son exécution, sinon, expliquez pourquoi.

```
public class Test1{
    public static void main (String [] args) {
        Animal chat = new Animal();
        System.out.println(chat.espèce);
    }
}
```

- 2.3 (1 pt) Écrivez un constructeur de la classe *Animal* possédant un paramètre de type *String* qui initialise l'attribut *espèce* de cette classe.
- 2.4 (1 pt) Écrivez le constructeur suivant, en faisant appel au constructeur de la question 2.2

```
Animal(String espèce, boolean estFemelle)
```

2. 5 (2 pts) Soit la classe Test2 suivante, qui fait appel aux deux constructeurs.  
Est-elle correcte? Si oui, donnez l'affichage de l'exécution de Test2, sinon, expliquez pourquoi.

```
public class Test2{
    public static void main (String [] args) {
        Animal chat = new Animal();
        Animal chien = new Animal("chien");
        System.out.println(chat.toString());
        System.out.println(chien.toString());
    }
}
```

2. 6 (2 pts) Soit la classe Test3 suivant. Est-elle correcte? Si oui, donnez l'affichage de l'exécution de Test3, sinon, expliquez pourquoi.

```
public class Test3{
    public static void main (String [] args) {
        Animal chat = new Animal("chat");
        Animal chien = new Animal("chien", false);
        chat = chien;
        chat.setEstFemelle(true);
        System.out.println(chat.toString());
        System.out.println(chien.toString());
    }
}
```

2. 7 (1 pt) Écrivez une classe appelée Mammifère qui hérite de la classe Animal.
2. 8 (1 pt) Afin que les attributs de Animal soient visibles également dans Mammifère (et pas dans d'autres classes) peuvent-ils rester "private" ? Expliquez et modifiez si nécessaire.
2. 9 (1 pt) Écrivez un constructeur de la classe Mammifère possédant un paramètre de type String qui initialise l'attribut espèce de cette classe en faisant appel au constructeur de la classe Animal.
2. 10 (1 pt) Écrivez la méthode toString() permettant un affichage sous la forme  
*Mammifère :*  
*Espèce : nomDeL'Espèce mâle*
2. 11 (2 pts) Soit la classe Test4 suivante.  
Est-elle correcte? Si oui, donnez l'affichage de l'exécution de Test4, sinon, expliquez pourquoi.

```
public class Test4{
    public static void main (String [] args) {
        Animal oiseau = new Animal("oiseau");
        Mammifère chat = new Mammifère("chat");
        Animal chien = new Mammifère("chien");
        Object femme = new Mammifère ("femme");
        System.out.println(oiseau.toString());
        System.out.println(chat.toString());
        System.out.println(chien.toString());
        System.out.println(homme.toString());
    }
}
```

2. 12 (1 pt) Sachant que le chat est une femelle, complétez, si possible, la classe de test suivante :

```
public class Test5{
    public static void main (String [] args) {
        Mammifère chat = new Mammifère("chat");

        System.out.println(chat.toString());
    }
}
```

2. 13 (2 pts) Ajoutez, dans la classe Mammifère,

- Un attribut de type entier qui est l'identifiant.
- Un attribut *propre à la classe*, de type entier, qui représente le numéro suivant à utiliser pour l'identifiant. Donnez-lui une valeur initiale.

2. 14 (1 pt) Modifiez le constructeur de Mammifère afin qu'il initialise aussi l'identifiant à la valeur numéroSuivant et incrémente la valeur de numéroSuivant de 1

2. 15 (2 pts) Dans la classe Mammifère, écrivez

- Une méthode **non statique** qui retourne la valeur de l'identifiant.
- Une méthode **statique** qui retourne la valeur du numéroSuivant.

Complétez la classe de test suivante afin que Test5 affiche l'identifiant du chat, l'identifiant du chien ainsi que le numéroSuivant.

```
public class Test5{
    public static void main (String [] args) {
        Mammifère chat = new Mammifère("chat");
        Mammifère chien = new Mammifère("chien");

    }
}
```

2. 16 (3 pts) Supposons maintenant que le code source des classes Animal et Mammifère soient dans /home/g30000/Examen/sources et que les .class soient dans /home/g30000/Examen/classes. Donnez une instruction package possible de la classe, ainsi que l'import et le CLASSPATH correspondants nécessaires pour utiliser cette classe à partir d'une classe d'un autre package.

### Troisième partie (8 pts)

3. 1 (1 pt) Écrivez une classe MonException héritant de Exception

3. 2 (2 pts) Écrivez la classe MaClasse. Cette classe possède un constructeur avec un paramètre booléen. Ce constructeur devra lancer une exception de type MonException lorsque le paramètre entré aura la valeur faux.

3. 3 (1 pt) La classe Test suivante est-elle compilable? Si oui, expliquez, si non, justifiez.

```
public class Test {
    public static void main (String [] args) {
        boolean a=true;
        System.out.println(a);
        new MaClasse (a);
    }
}
```

- 3.4 (3 pts) Complétez la classe Test suivante. afin qu'elle attrape les exceptions de type MonException mais aussi toute autre exception avec un message différent.

```
public class Test {
    public static void main (String [] args) {
        boolean a=false;
        System.out.println(a);
        new MaClasse (a);
    }
}
```

- 3.5 (1 pt) La division par 0 provoque une exception de type ArithmeticException héritant de RuntimeException. Pourquoi n'est-on pas obligé de mettre toute division dans un « try-catch » ?

### Quatrième partie (4 pts)

Soit la classe suivante

```
import java.util.*;
public class Test1 {
    public static void main (String [] args) {
        ArrayList<Integer> al = new ArrayList<Integer>();
        for (int i=0; i<10; i++)
            al.add(5);
        for (int i=0; i<al.size(); i++) {
            System.out.println(al.get(i));
        }
    }
}
```

- 4.1 (2 pts) Récrire cette classe en utilisant la notion d'itérateur  
4.2 (2 pts) Récrire cette classe en utilisant le foreach.

### Cinquième partie (4 pts)

- 5.1 (2p) Quels sont les 3 piliers de l'Orienté Objet? Expliquez-en un au choix  
5.2 (2p) Expliquez brièvement la notion d'interface.