TEK-UP
Ecole Supérieure Privée
Technologie & Ingénierie

Ecole Supérieure Privée Technologies & Ingénierie

Type d'épreuve	: ☐ Devoir	■ Examen Session Principale	
Enseignants	: Mr Mohamed RJAB, Mme Basma KHIL, Mr Slah BOUHARI		
Matière	: JAVA		
Année Universitaire	: 2022-2023	**Semestre**	: 2
Classe	: TIC-1		
Documents	: ☐ Autorisés	■ Non autorisés	
Date	: 01/06/2023	**Durée :**	1h30mn
Nombre de pages	: 5		
Barème	: 8, 12		

Exercice1 : (8 points)

1. Quelle est la sortie de code suivant ?

```
public class MyClass {
    public static void main (String[]
args) {
String str1 = "Hello";
String str2 = new String("Hello");
 String str3 = "Hello";
 System.out.println(str1 == str2);

System.out.println(str1.equals(str2));
System.out.println(str1 == str3);
    }
}
```

A) true, true, true
B) false, true, true
C) true, false, true
D) false, false, true
E) true, false, false

2. Quelle est la sortie de code suivant ?

```
1. public class Person {
2. String name = "No name";
3. public Person(String nm) {
4. name = nm; }
5. }
6. public class Employee extends
Person {
7. String empID = "0000";
8. public Employee(String id) {
empID = id; }
9. }
10.
11. public class EmployeeTest {
12. public static void main(String[]
args){
13. Employee e = new
Employee("4321");
14. System.out.println(e.empID);
15. }
16. }
```

A) 4321
B) 0000
C) An exception is thrown at runtime.
D) Compilation fails because of an error in line 8.
E) Compilation fails because of an error in line 14.

3. Quelle est la sortie du code ci-dessous ?

```
abstract class Animal {
    private String name;

    public Animal(String name) {
        this.name = name;     }

 public abstract void makeSound();

  public void printName() {
     System.out.println("Name: " +
name); }}

class Dog extends Animal {
    public Dog(String name) {
        super(name);    }

    @Override
    public void makeSound() {

System.out.println("Woof!"); }}

public class Main {
    public static void
main(String[] args) {
Animal animal = new Dog("Buddy");
        animal.printName();
        animal.makeSound();}}
```

A) Name: Buddy
   Woof!
B) Woof!
   Name: Buddy
C) Compilation Error
D) Runtime Error

4. Quelle est la sortie du code ci-dessous ?

```
interface Printable {
    void print();
}
class Document implements
Printable {
    private String content;

 public Document(String content) {
        this.content = content;
}

 @Override
 public void print() {

System.out.println("Printing
document: " + content);    }}

class Image implements Printable {
    private String fileName;

public Image(String fileName) {
        this.fileName = fileName;
}

 @Override
 public void print() {

System.out.println("Printing
image: " + fileName); }}

public class Main {
    public static void
main(String[] args) {
     Printable printable1 = new
Document("Sample Document");
     Printable printable2 = new
Image("image.jpg");

     printable1.print();
     printable2.print();}}
```

A) Printing document: Sample Document
   Printing image: image.jpg
B) Printing image: image.jpg
   Printing document: Sample Document
C) Compilation Error
D) Runtime Error

5. Quelle est la sortie du code ci-dessus ?

```
public class ExceptionExample {
    public static void
main(String[] args) {
 try {
       int result = divide(10, 0);

System.out.println("Result: " +
result);
    }
catch (ArithmeticException e) {

System.out.println("ArithmeticExce
ption caught");}
catch (Exception e) {

System.out.println("Exception
caught");
}
finally {

System.out.println("Finally block
executed");
}
}
public static int divide(int
dividend, int divisor) {
    if (divisor == 0) {
         throw new
ArithmeticException("Divisor
cannot be zero");
   }
   return dividend / divisor;
}
}
```

A) ArithmeticException caught
   Finally block executed
B) Exception caught
   Finally block executed
C) ArithmeticException caught
D) Compilation Error

6. Quelle est la sortie du code ci-dessus ?

```
class X{
    String titre="X";
    void affiche(String s){
    System.out.println(s+" "+titre);
    }
}

class XY extends Y{
     String titre="XY";
     void affiche(String s){
```

```
class Y extends X{

      String titre="Y";
      void affiche(String s){
       super.affiche(s+" "+titre);
      }
}
     super.affiche(titre+" "+s);
 }
 }
```

Que réalise le code suivant : **`X unx = new XY(); unx.affiche("fils de");`**

A) Provoque une erreur à la compilation
B) XY fils de X Y
C) XY fils de Y X
D) XY fils de

7. Choisir la ou les bonnes réponses.
*La classe **A** hérite de **B** qui hérite de **C**. **C** est une classe abstraite qui implémente une interface **I**. **A** et **B** ne sont pas des classes abstraites.*
   A) C peut implémenter une partie des méthodes de l'interface I
   B) C doit avoir une méthode abstraite
   C) A et B doivent à elles deux implémenter toutes les méthodes de I qui n'ont pas été implémentées par C
   D) A ne peut pas implémenter des méthodes de l'interface I

8. Choisir la ou les bonnes réponses.
En Java, on déclare un tableau de la manière suivante T[] tab = new T[100].
   A) T peut être une classe abstraite
   B) T peut être une interface
   C) T ne peut pas être de type primitif
   D) Aucune proposition n'est correcte

Problème : (12 points)

N.B. : lire tout l'énoncé du problème avant de répondre

Dans le cadre de l'informatisation d'une mairie, on veut automatiser le calcul des impôts locaux. On distingue deux catégories d'habitations : les habitations à usage professionnel et les maisons individuelles, l'impôt se calculant différemment selon le type d'habitation considérée. Pour cela, on définit les classes **HabitationProfessionnelle** et **HabitationIndividuelle** et les caractéristiques communes à ces deux classes sont regroupées dans une classe nommée **Habitation**.

La classe Habitation comprend les attributs suivants : proprietaire (String), adresse(String), surface(double).

Les méthodes exigées :

- double impot() : qui permet de calculer le montant de l'impôt que doit payer le propriétaire de l'habitation à raison de 2 € (euro) par m^2.
- void affiche() : qui permet d'afficher les trois attributs de la classe Habitation.
- On a également besoin d'un constructeur à trois paramètres permettant d'initialiser une instance de la classe Habitation.

Les classes HabitationIndividuelle et HabitationProfessionnelle héritent de la classe Habitation.

- La classe HabitationIndividuelle est caractérisée par les attributs suivants : nbPieces (int) et existe_piscine(boolean).
- Le calcul de l'impôt d'une maison individuelle se calcule en fonction de la surface habitable, du nombre de pièces et de la présence ou non d'une piscine. On compte 100

€/ pièce et 500 € supplémentaire en cas de présence d'une piscine (la formule : 2*surface + 100*nbPieces + 500 si piscine).

- La classe HabitationProfessionnelle est caractérisée par l'attribut suivant : nbEmployes (int)
- Le calcul de l'impôt d'une habitation à usage professionnel se calcule en fonction de la surface occupée, comme dans le cas de la classe Habitation, et du nombre d'employés travaillant dans l'entreprise. On compte 1000 € supplémentaire par tranche de 10 employés.
- Les classes HabitationIndividuelle et HabitationProfessionnelle doivent redéfinir les deux méthodes de la classe Habitation (en considérant les changements nécessaires). **Il est obligatoire d'utiliser les méthodes de la super-classe**. Chaque classe doit avoir un constructeur adéquat respectant les règles d'héritage.

Quelques situations exceptionnelles peuvent être générées lors du calcul de l'impôt pour les habitations individuelles et professionnelles. Si des valeurs négatives sont fournies pour la surface habitable ou le nombre de pièces d'une habitation individuelle, cela peut conduire à des résultats incorrects lors du calcul de l'impôt.

N'oublier pas de vérifier les valeurs lors du calcul de l'impôt et en lançant une exception appropriée si nécessaire.

1. Définir la classe Habitation.
2. Définir la classe d'exception ImpotException pour gérer les problèmes de type « valeurs négatives » lors du calcul de l'impôt.
3. Implémenter la classe HabitationIndividuelle
4. Définir la classe HabitationProfessionnelle

On désire à présent calculer l'impôt local des habitations (individuelles ou professionnelles) d'une commune. Pour cela, on utilise une collection d'objets représentée par un tableau (vous pouvez utiliser un tableau dynamique) où chaque élément désigne une habitation individuelle ou professionnelle.

5. Définissez ainsi une classe Commune comportant les attributs : lesHabitations : un tableau (ou ArrayList : tableau dynamique) d'habitations et un nom (le nom de la commune)

Ajouter les méthodes :

- Un constructeur (avec paramètre(s))
- AjoutAbitation permettant d'ajouter un objet Habitation à l'ensemble des habitations de la commune.
- calculImpôtLocal permettant de calculer l'impôt local payé par l'ensemble des habitations de la commune.