**Durée : 1h AUCUN DOCUMENT - AUCUNE CALCULATRICE**

**Lisez attentivement chaque partie du sujet avant de vous lancer dans vos réponses.**

**Partie 1 : 8 QCM ≈ 15 minutes (8 POINTS). Une seule réponse par question :**

**1 point par réponse juste, -0,5 point par réponse fausse, 0 point sans réponse**

**Q1)** Java est un langage

**(a)** Compilé

**(b)** Interprété

**(c)** Compilé et interprété

**(d)** Ni compilé ni interprété

**Q2)** Quelle classe n’a pas de classe mère ?

**(a)** Orpheline

**(b)** String

**(c)** Object

**(d)** aucune

**Q3)** Parmi ces librairies Java (packages) suivantes, laquelle est importée automatiquement ?

**(a)** *javax.swing* permettant de créer des programmes qui utilisent une interface graphique

**(b)** *java.io* permettant de créer des programmes qui accomplissent des opérations d'entrée/sortie

**(c)** *java.util* contenant une série de classes utilitaires (Scanner, Random etc.)

**(d)** *java.lang* contenant les classes de base (String, Math etc.)

**Q4)** Combien de références de type String sont créées par l’instruction ci-dessous?

*String[] strings = new String[10];*

**(a) 0**

**(b) 9**

**(c) 10**

**(d) 11**

**Q5)** Pour la classe définie comme suit :

*public class D {*

*public int x ;*

*public D() {x=3 ; } ;*

*public D( int a) {this() ; x=x+a ;}*

*public D( int a, int b) {this(b) ; x= x-a ;}*

*}*

Qu’affichera le code suivant ?

*D a=new D(5,6) ;*

*System.out.println(a.x) ;*

**(a)** 1

**(b)** 2

**(c)** 3

**(d)** 4

**Q6)** Pour la classe définie comme suit :

*public class Bidon {*

*int x ;*

*}*

Lequel des programmes est faux ?

**(a)** Bidon a=new Bidon(3) ;

**(b)** Bidon a=new Bidon() ; String s=a.toString() ;

**(c)** Bidon a=new Bidon() ; boolean b= a.equals(”bonjour”) ;

**(d)** Bidon a=new Bidon() ; boolean b= a.equals(a) ;

**Q7)** Pour les classes *A* et *D* définies comme suit :

*public class A { public class D extends A {*

*public static int f(int x) {return(x+5) ; } public static int f(int x) {return(x+4) ; }*

*public int g(int x) {return (3) ; } public int g( int x) {return (x+8) ; }*

*} }*

Qu’affichera le code suivant ?

*D d=new D() ; A a =d ;*

*System.out.println(A.f(2)\*a.g(3)) ;*

**(a)** 18

**(b)** 21

**(c)** 66

**(d)** 77

**Q8)** Pour les classes *A* et *B* définies comme suit:

*public class A { public class B extends A {*

*public int x; public B() {x++;}*

*public A() {x=5; } public B(int i) {this(); x=x+i; }*

*} public B(String s) {super(); x- -; }*

*}*

Qu’affichera le code suivant ?

*B b1=new B(); B b2 =new B(2003); B b3= new B(”Bonjour”);*

*System.out.println(b1.x + ” et ” + b2.x + ” et encore ” + b3.x );*

**(a)** 6 et 2009 et encore 4

**(b)** 1 et 2004 et encore 4

**(c)** 1 et 2004 et encore 2003

**(d)** autre chose

**Partie 2 : code Java ≈ 45 minutes (12 POINTS)**

Des extraits de la documentation des API Java sont fournis en **annexe** dans la dernière page.

Pour chacune des classes suivantes, n’oubliez pas d’importer les éventuelles librairies nécessaires. Nous supposons que ces classes font partie du même package.

* **(4 points)** Définir une classe *Compte* avec trois attributs privés: le montant du compte de type **float**, le numéro du compte de type **int**, le nom du propriétaire de type *String*. On y ajoute les éléments suivants :
* Deux constructeurs, l’un par défaut et l’autre avec tous les paramètres nécessaires.
* Tous les getters.
* La méthode surchargée *modifier* dont l’objectif est de modifier le montant du compte : la première méthode cumule le montant du compte avec le montant en paramètre (de type **float**) à condition que cette somme soit positive ; la seconde méthode cumule le montant du compte avec le montant en paramètre (de type **float**) multiplié par la durée en nombre de mois (de type **int**) également en paramètre, à condition que cette somme soit positive.
* La méthode surchargée *toString()* dont l'objectif est de décrire tous les attributs de l'objet considéré avec une chaîne de caractères : cette méthode retourne une expression sous forme de chaîne.

public class Compte {

// attributs privés

private float montant ;

private int num ;

private String nom ;

// constructeurs surchargés

public Compte() {

montant=0 ;

num = 0 ;

nom = null ;

}

public Compte(float montant, int num, String nom) {

this.montant= montant;

this.num = num;

this.nom = nom ;

}

// getters

public float getMontant() { return montant ;}

public int getNum() { return num ;}

public String getNom() { return nom ;}

// Méthode surcharge

public void modifier(float montant) {

float somme = this.montant + montant;

if (somme>0)

this.montant = somme;

}

public void modifier(float montant, int mois) {

float somme = this.montant + montant \* mois;

if (somme>0)

this.montant = somme;

}

@Override

Public String toString() {return "compte numero "+num+" de "+nom+" = "+montant;}

}

* **(4 points)** Définir une classe *TestCompte* avec le **main** qui effectue les traitements suivants :
* Saisir au clavier un montant, un numéro et un nom du propriétaire d’un compte tous du même type que les attributs de la classe Compte. Blinder la saisie du montant et du numéro pour qu’ils soient positifs.
* Créer un objet de la classe *Compte* avec les informations saisies.
* Afficher tous les attributs de l’objet sans la méthode *toString().*
* Saisir au clavier un nouveau montant sans le blinder.
* Modifier le montant de l’objet avec le nouveau montant saisi en utilisant la bonne méthode *modifier.*
* Modifier le montant de l’objet avec le nouveau montant saisi sur une durée de 12 mois en utilisant la bonne méthode *modifier.*
* Afficher les attributs de l’objet avec la méthode *toString().*

import java.util.Scanner ;

public class TestCompte {

public static void main(String args[]) {

// saisie au clavier du montant positif, du num. positif et du nom

Scanner clavier = new Scanner(System.in);

float montant;

int num;

do {

System.out.println("Entrez le montant positif :");

montant = clavier.nextFloat() ;

} while (montant<=0) ;

do {

System.out.println("Entrez le numero positif :");

num = clavier.nextInt() ;

} while (num<=0) ;

System.out.println("Entrez le nom :");

String nom = clavier.next();

// creation (instanciation) de l’objet de Compte avec les informations saisies

Compte c = new Compte(montant, num, nom) ;

// affichage des attributs de cet objet avec les getters

System.out.println("compte numero "+c.getNum()+" de "+c.getNom()+" = "+c.getMontant()) ;

// Saisir un nouveau montant

System.out.println("Entrez un nouveau montant :");

montant = clavier.nextFloat() ;

c.modifier(montant) ; // modifier le montant de cet objet avec le montant saisi

c.modifier(montant,12) ; // modifier le montant de cet objet avec le montant saisi sur 12 mois

// affichage des attributs de cet objet avec la methode toString()

System.out.println(c.toString()) ;

}

}

* **(2 points)** Définir une classe *CompteRemunere* qui hérite de la classe mère *Compte* avec un attribut privé : le taux de rémunération annuelle de type **float**. On y ajoute les éléments suivants :
* Un constructeur avec tous les paramètres nécessaires du *CompteRemunere* en appelant le bon constructeur de la classe mère.
* Une méthode avec le nombre d’années en paramètre qui calcule et retourne le montant du compte, en appliquant le taux de rémunération annuelle du compte après ce nombre d’années.

public class CompteRemunere extends Compte {

private float taux ;

// constructeur par héritage

public CompteRemunere(float montant, int num, String nom, float taux) {

super(montant, num, nom) ; this.taux=taux;

}

// methode de calcul du montant avec taux et nombre d’année

public float getMontant(int nbAnnees) {

// le super dessous pas obligatoire

return super.getMontant() \* (float) Math.pow(1 + taux, nbAnnees); // autre solution : boucle

}

}

* **(2 points)** Dans la suite du **main**, écrire les traitements suivants :
* Créer un objet de la classe *CompteRemunere* avec les informations saisies précédemment et un taux de rémunération annuelle généré aléatoirement entre 1.0 et 3.5.
* Afficher le montant de cet objet en tenant compte du taux de rémunération annuelle généré à appliquer pour une durée de 20 ans.

import java.util.Random // avant la classe TestCompte

// suite du main

// Calcul du taux aléatoire entre 1.0 et 3.5

Random r = new Random() ;

float taux = r.nextFloat() \* (float) (3.5-1.0)+ (float) 1.0; // pas sûr que les cast soient nécessaires

// Créer un objet de la classe Compte Remunere avec les infos saisies et le taux aléatoire

CompteRemunere cr = new CompteRemunere(montant, num,, nom, taux) ;

// Afficher le montant de cet objet pour une durée de 20 ans

System.out.println("montant sur 20 ans pour un taux annuel de "+ taux+" = "+cr.getMontant(20)) ;

**ANNEXE : extraits de la documentation des API Java**

## java.util

## Class Scanner

|  |  |
| --- | --- |
| **Method Summary** | |
| [String](http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) | [**next**](http://docs.oracle.com/javase/6/docs/api/java/util/Scanner.html#next())()  Finds and returns the next complete token from this scanner. |
| **float** | [**nextFloat**](http://docs.oracle.com/javase/6/docs/api/java/util/Scanner.html#nextFloat())()  Scans the next token of the input as a float. |
| **int** | [**nextInt**](http://docs.oracle.com/javase/6/docs/api/java/util/Scanner.html#nextInt())()  Scans the next token of the input as an int. |
| [String](http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) | [**nextLine**](http://docs.oracle.com/javase/6/docs/api/java/util/Scanner.html#nextLine())()  Advances this scanner past the current line and returns the input that was skipped. |

**Class Random**

|  |  |
| --- | --- |
| **Method Summary** | |
| **float** | [**nextFloat**](http://docs.oracle.com/javase/6/docs/api/java/util/Random.html#nextFloat())()  Returns the next pseudorandom, uniformly distributed float value between 0.0 and 1.0 from this random number generator's sequence. |

**java.lang**

**Class Math**

|  |  |
| --- | --- |
| **Method Summary** | |
| **static double** | [**pow**](http://docs.oracle.com/javase/6/docs/api/java/lang/Math.html#pow(double, double))(**double** a, **double** b)  Returns the value of the first argument raised to the power of the second argument. |