1. ***C’est quoi le problème de référence communes ?***

**a. Deux objets de même type ou de type différents ayant une propriété dynamique, qui pointe sur un même objet.**

b. Deux objets de même type ayant une propriété statique de même valeur dans les deux objets.

c. L’affectation entre deux objets dynamiques de même type.

1. ***Dans quelle situation le problème des références communes entre objets d’une classe A peut-il avoir lieu.***

**a. Affectation entre objets statique de type A**

b. Affectation entre objets dynamiques de type A

c. Appel á une méthode ayant un paramètre de type : (A)

d. Appel á une méthode ayant un paramètre de type : (A &)

1. ***Dans quel endroit le problème des références communes peut-il être résolu ?***

**a. Constructeur de copie**

b. Constructeur par défaut

**c. surcharge de l’opérateur =**

d. Destructeur

1. ***En comparant l’affectation et l’initialisation, on peut dire :***

a. Les deux utilisent l’opérateur =

b. Les deux utilisent le constructeur de copie

**c. L’affectation utilise l’opérateur =, et l’initialisation utilise le constructeur de copie.**

d. L’affectation utilise le constructeur de copie, et l’initialisation utilise l'opérateur =

1. ***Une variante du constructeur de copie est la notion de constructeur de conversion, dans quel cas faut-il créer un constructeur de conversion ?***

**a. Possibilité d’initialiser un objet statique d’une classe A avec un objet d’une classe B.**

b. Convertir un objet dynamique de type A en un objet dynamique de type B.

c. Possibilité d’accepter le casting entre une classe A et une classe B

1. ***La surcharge de l’opérateur = doit être réaliser pour les raisons suivantes :***

**a. Résoudre des problèmes de gestion mémoire lors de l’affectation entre 2 objets statiques de mêmes types.**

**b. Permettre l’affectation entre deux objets de types différents (possible que vrai)**

c. Eviter le casting entre deux objets de types compatibles

d. Permettre l’affectation entre deux objets dynamiques de même type

1. ***Le quelle des définitions suivantes est la plus correcte et la plus recommandée concernant la surcharge de l'opérateur pour une classe A ?***

a. A operator=(A a) ;

b. void operator=(A &a) ;

**c. A& operator=(A &a) ;**

d. void operator=(A a) ;

1. ***Quelle est la définition correcte concernant la surcharge de l’operator<< pour une classe A ?***

a. friend void operator << (A &p1) ;

**b. friend ostream& operator<<(ostream &os, A &a);**

c. friend ostream &operator<<(A &p1);

d. friend ostream operator<<(ostream os, A p1);

1. ***Sélectionner la ou les instructions équivalent á l’instruction suivante : cout << ‘’Poo en C++’’ ;***

a. cout.operator<<(‘’ Poo en C++) ;

**b. operator<<(cout, ‘’POO en C++’’);**

c. << (cout, ‘’ Poo en C++’’);

**d. std::cout << “POO en C++’’;**

Remarque : int et les float en utilise cette syntaxe 🡪 cout.operator<<(3432.24) ;

1. ***Qu’est-ce que peut dire sur ‘’ cout’’ :***

a. Il s’agit d’un opérateur d’affichage des objets combiné avec l’opérateur ‘’ << ‘’

**b. C’est un objet de type ostream déclaré dans la bibliothèque iostream représentant un flux de sortie vers l’écran.**

c. C’est une instruction d’affichage qui remplace printf dans le cas de la POO en C++

1. ***Quel est l’intérêt de définir une méthode friend ?***

**a. Fournir á la méthode la possibilité d’accéder aux membres privés de ses paramètres ,s’ils sont des objets de la classe dans laquelle elle est déclarée.**

b. Créer une fonction qui sera considérée comme une méthode de la classe dans laquelle elle est définie.

c. Créer une fonction amie d’une classe qui peut construire ou détruire les objets de la classe dans laquelle elle est définie.

1. ***Soit une classe A. durant l’exécution de code suivant combien d'instances seront créées.***

A createObjecte(){

A object ;

return object

}

A object1, object2 ;

object2 = createObjecte() ;

a. 3 instances.

**b. 4 instances**

c. 5 instances.

1. ***Soit une classe A avec un constructeur par défaut un constructeur de copie est surcharge de l’opérateur =. Durant l’exécution du code précédent, quel sont les membres appelés*** ?

**a. Constructeur par défaut.**

**b. Constructeur de copie.**

**c. Operateur =**

1. ***Soit une classe A durant l’exécution du code suivant combien d’instance seront créer***

A createObjecte(){

A object;

return object

}

A object1 ;

A object2 = createObjecte () ;

**a. 3 instances**

b. 4 instances

c. 5 instances

1. ***Soit une classe A avec un constructeur par défaut de copie, surcharge de l’operator= durant l’exécution de code suivant, quels sont les membres appelés ?***

void printObject(A param){

Cout << param ;

}

A arg ; printObject(arg) ;

**a. Constructeur par défaut.**

**b. Constructeur de copie.**

c. Operateur =

1. ***Soit une classe A avec un constructeur par défaut de copie, surcharge de l’operator= durant l’exécution de code suivant, quels sont les membres appelés ?***

void printObject(A &param){

Cout << param ;

}

A arg ; printOjbect(arg) ;

**a. Constructeur par défaut**

b. Constructeur de copie.

c. Operateur =.

1. ***Sélectionner la bonne définition du constructeur de copie d’une classe A données ?***

a. A(A \*object) ;

b. A(A object) ;

**c. A(A &object) ;**

d. copy A(A &object) ;

1. ***Indique les positions d’exécution des destructeurs sur les différents objets créés ?***

void function2(){

A object ;

cout << object ;

}$ici$

Void function1(int counter){

A \* object1 ;

if(counter > 1) {

A\* object = new A();

object1 = object2;

}

Fuction2();

delete object 1;$ici$

}

Function1();

1. ***Nous disposons d’une classe mère qui a un seule constructeur avec un paramètre de type chaîne de caractère permettant de communiquer sa valeur a une propriété ‘’ nom’’ , nous disposons aussi d’une classe Fille fille de la classe mère avec un seul constructeur sans paramètre qui devrez fixer le nom á la chaîne fille***.
   1. ***Ecrire ce qui manque dans la déclaration de la classe mere.h***

class Mere{

protected :

char\* nom ;

public :

Mere(char\* nom){

this->nom = (char\*)malloc(strlen(nom) + 1);

strcpy(this->nom, nom);

}

};

* 1. ***Ecrire ce qui manque dans la déclaration dans la classe fille.h***

class Fille : public Mere{

public :

Fille() : Mere(‘’’’) ;

} ;

1. ***Nous disposons d’une classe M et d’une classe F, fille de la class M avec les objets suivante :***

***M m1, \*m2 ; F f1, \*f2 ;***

***Cocher les affectations possibles syntaxiquement :***

**a. m1 = f1 ;**

**b. m2= f2 ;**

c. f1 = m1 ;

d. f1 = (F)m1 ;

e. f2 = m2 ;

**f. f2 = (F\*)m2 ;**

Partie Java:

***1. Cocher les éléments déclare incorrectement ou ne respectant pas les conventions de nommage :***

1. package com.company.app ;
2. package java.lang ;
3. **package com.SMI.app ;**
4. **class entreprise {…} ;**
5. class Traitement{…} ;
6. **class gestionDesFichier{…} ;**
7. **void GetData{…} ;**
8. **public static final int maxInt = 20 ;**
9. **public static final int MIN\_VALUE ;**
10. public static final double PI = 3.14 ;

***2. Comment déclarer un compteur d’instances :***

a. public static final int counter = 0 ;

b. public static int counter = 0 ;

c. private int counter = 0 ;

**d. private static int counter = 0 ;**

***3. On*** ***voudrait gérer des employés d’une entreprise. Pour cela on a identifié 3 types d’employés : Permanent, contractuel et stagiaire. La méthode du calcule de salaire prévu dans la classe <<Employée>> devrait donc avoir une implémentation différente pour chaque cas, En plus et afin de prévoir différentes façons d’afficher les employés, on a prévu un affichage séparé de la classe <<Employée>> á l’aide d’un afficheur disposant de 2 concrétisations possibles : afficheur console et afficheur graphique.***

***On demande alors de dégager les différentes structures décrites par l’énoncé (dans la colonne <<Nom>> tout en précisant leur type (classe, classe abstraite ou interface), la classe mère de chacune ainsi que la classe qui la contient si elle existe.***

| **Nom** | **Classe** | **Classe abstraite** | **Interface** | **Classe Fille de** | **Agrégée dans** |
| --- | --- | --- | --- | --- | --- |
| **Employee** |  | **Oui** |  | **Object** |  |
| **Permanent** | **Oui** |  |  | **Employee** |  |
| **Contractuel** | **Oui** |  |  | **Employee** |  |
| **Stagiaire** | **Oui** |  |  | **Employee** |  |
| **Afficheur** |  |  | **Oui** |  | **Employee** |
| **AfficheurConsole** | **Oui** |  |  | **Afficheur** |  |
| **AfficheurGraphique** | **Oui** |  |  | **Afficheur** |  |

***4.Soit la classe suivante :***

package p1 ;

class A{

int p1 ;

private int p2 ;

public int p3 ;

protected int p4 ;

}

***Cocher les instructions erronées :***

package p1 ;

class B {

public void m1(){

A a1 = new A() ;

1. a1.p1  = 0 ;
2. **a1.p2 = 0 ; erronée**
3. a1.p3 = 0 ;
4. a1.p4 = 0 ;

}

}

package p2 ;

class C extends A{

public void m1(){

1. **p1++ ; erronée**
2. A a1 = new A() ;
3. **p2++ ; erronée**
4. **a1.p1 = 0 ; erronée**
5. **a1.p2 = 0 ; erronée**
6. p3++ ;
7. p4++ ;
8. a1.p3 = 0 ;
9. a1.p4 = 0 ;

}

}

***5. Soit la classe ci-après. Cocher les instructions incorrectes :***

1. class A{

2. private int p1 ;

3. private int static p2 ;

4. public void m1(){

5. p1 = 0 ;

6. p2 = 0 ;

7. }

8. public void static main (){

**9. m1() ; erronée**

**10. p1++ ;**

11. p2++ ;

12. }

13. }

***6. Quelles sont les instructions incorrectes en Java :***

**a. float pi = 3.14 ;**

b. char c = 65 ;

c. Object s = ‘’ABC’’ ;

**d. Double x = 10 ;** erronée, il faut faire Double x = 10. ;

***7. Soient deux classes différentes A et B, et deux objets (A p1) et (B p2). Cocher les instructions incorrectes :***

**a. p1 = p2 ;**

**b. p1 = (A)p2 ;**

c. Object p3 = p1 ;

**d. p2 = new Object() ;**

***8. L’intérêt et la raison d’être des interfaces java :***

a. Réaliser des types énumères.

**b. Faire une spécification et une abstraction de services.**

c. Réaliser l’héritage multiple.

d. Réaliser des interfaces graphiques.

***9. Sélectionner la ou les assertions correctes á propos des interfaces :***

a. Une interface peut implémenter une autre interface.

b. Une interface peut contenir des propriétés privées.

**c. Une interface peut étendre une autre interface.**

**d. Une classe peut implémenter une ou plusieurs interfaces.**

***10. On considère la création d’un <<Vector>> comme suit :***

Vector<Object> v1 = new Vector<>() ;

***Quelle sont les assertions correctes á propos de l’instruction suivante :*** v1.add(24) ;

1. Instruction erronée.
2. **Instruction correcte**
3. **Utilisation de l’autoboxing**

**d. Affectation fille -> mère**

***11. Quelles sont les assertions correctes á propos des classes : Vector et LinkedList :***

**a. La classe Vector est basée sur un tableau et pas la classe LinkedList.**

**b. La classe Vector est plus rapide que la classe LinkedList.**

c. Les deux classes sont basée exactement sur la même structure.

**d. La classe Vector utilise un accès direct alors que la classe LinkedList utilise un accès séquentiel.**

***12. Qu’est-ce qu’on peut dire á propos des méthodes virtuelle en Java :***

a. On ne peut pas avoir des méthodes virtuelles en Java.

b. On peut définir une méthode virtuelle á l’aide du modificateur <<virtual>>.

**c. Les méthodes en Java sont automatiquement virtuelles.**

***13. Pour ajouter un élément á un tableau Java t[] :***

a. Il suffit de changer la valeur de la propriété length : t.length = t.length + 1 ;

**b. Il faudra créer un nouveau tableau avec la taille t.length + 1, copier le contenu du premier dans le nouveau , et référencer la nouvelle adresse du tableau.**

c. Il faudra appeler la méthode add sur le tableau : t.add(…)

***14. L’autoboxing en Java permet d’affecter :***

**a. Une donnée de type primitif á un objet de type wrapper**

b. Un objet á une donnée de type primitif

c. Une donnée de type primitif á un objet quelconque

***15. Nous disposons d’une classe <<M>> et d’une classe <<F>>, fille de la classe <<M>> avec les objets suivants des deux classes M et F :***

F f1 ;

M m1 ;

Cocher les affectations possibles syntaxiquement :

**a. m1 = f1 ;**

b. f1 = m1 ;

**c. f1 = (F)m1 ;**

***16. Nous disposons d’une classe <<M>> et d’une classe <<F>> fille de la classe <<M>> dont le code est le suivant :***

public class M{

public void print(){

System.out.println(‘’Mére’’) ;

}

public class F extends M{

public void print(){

System.out.println(‘’Fille’’) ;

}

}

***Soit le code suivant :***

M m1 = new F() ;

m1.print() ;

***L’exécution du code affichera :***

**a. Fille**

b. Mère

c. Une exception

***17. Soient les objets String s1 et StringBuffer sb1, pour affecter s1 á sb1 :***

a. sb1 = (StringBuffer)s1 ;

b. sb1 = s1 ;

**c. sb1 = new StringBuffer(s1) ;**

d. Affectation impossible car ce sont 2 objets de différents types

***18. Quelles relation entre String et StringBuffer :***

a. String est fille de StringBuffer

b. StringBuffer est fille de String

**c. Elle ont une mère commune**

d. Pas de relation

***19. Quelle sont les solutions optimales pour parcourir une LinkedList<String> L :***

**a. for(String item : L) {…}**

b. for(int i= 0 ;i < L.size() ;i++) { String item = L.get(i) ; …}

**c. Iterator<String> it = L.iterator() ; while(it.hasNext()) {String item = it.next() ; …}**

***20. Dans quel package trouver les classes de gestion des structures de données en Java :***

a. java.lang

**b. java.util**

c. java.struct