Ecole Nationale des Sciences Appliquées - Al Hoceima

Département Mathématiques et Informatique

Travaux dirigés : Les bases du langage Java

Module : Programmation Orientée Objet en Java

Première Année Ingénierie des Données

Année universitaire 2022/2023

Répondre aux questions suivantes :

1. Pourquoi Java est appelé un langage Independent de la plateforme (*Platform-Independent*) ?
2. Est-ce que Java est un langage 100 % orienté objet ? Pourquoi ?
3. Qu'elle est la différence entre Double et double?
4. Donner un code qui permet de convertir un "int" en "char" ?
5. Donner un code qui permet de convertir un caractère majuscule en un caractère minuscule.
6. Quels sont les types primitifs et quelles sont leurs valeurs par défaut?
7. Expliquer comment un programme Java libère la mémoire non utilisée. comment est appelé ce mécanisme ?
8. Quel est le résultat de l'opération suivante en JAVA : 9/2 + 9.0/2 ?
9. Donner la valeur des expressions Java ci-dessous :
   a) *5.0 / 4 - 4 / 5*
   b) *7 < 9 - 5 && 3 % 0 == 3*
   c) *"B" + 8 + 4*
10. Est-ce qu'il est possible d'affecter null à une variable de type double ?
11. Est-ce qu'il est possible d'affecter null à une variable de type Double ?
12. Quelle est l'instruction incorrecte dans le code suivant :

```
int i = 32;
float f = 45.0;
double d = 45.0;
```

13. Soit c une variable de type char, L'expression *c++* est-elle équivalente à *c = c + 1* ? sinon pourquoi ?
14. Quelles sont les erreurs contenues dans le code ci-dessous ?

```
int n ;
short p ;
        char c ='1';
byte b1, b2 ;
n = b1 * b2+ c ;
p = b1 * b2 ;
```

15. Qu'affiche le programme suivant :

```
public static void main(String[] args) {
        int i = 10, j = 10, ii = 10, jj = 10;
        boolean bool1, bool2;
        bool1 = true | (i++ < 0);
        bool2 = true || (j++ < 0);
        System.out.println("i:" + i + " j:" + j);
        bool1 = false & (ii++ < 0);
        bool2 = false && (j++ < 0);
        System.out.println("ii:" + ii + " jj:" + jj);
        bool1 = bool1 ^ true;
        System.out.println(bool1);
        bool1 = bool1 ^ true;
        System.out.println(bool1);
}
```

16. Qu'affiche le programme suivant :

```
public class GiTest1 {
   public static void main(String[] args) {
byte choix =0;
```

```
choix= choix+1;
switch(choix){
case 1 : System.out.println("ok");
case 2 : System.out.println("ok");break;
case 3 : System.out.println("NO");
default : System.out.println("KO");
}
   }
}
```

17. Soient ces déclarations :

```
byte b ; short p ; char c ; int n ; float x ;
```

Parmi les expressions suivantes, lesquelles sont incorrectes et pourquoi ?

```
c = c + 2;
c--;
c += 1;
b -= c;
p -= b;
p = p – 2* b;
n -= x;
n = n + (x++);
x++;
```

18. Soient ces déclarations :

```
byte b ; short p ; int n ; long q ;
final int N=10 ;
float x ; double y ;
```

Parmi les expressions suivantes, lesquelles sont incorrectes et pourquoi ?

```
b = n
b = 25
b = 500
x = 2*q
y = b*b
p = b*b
b = b+5
p = 5*N-3
```

19. Soit le programme :

```
public class FirstProgram {
        public static void main(String args[]) {
                byte lbyte1 = 50, lbyte2 = 100;
                int n;
                n = lbyte1 * lbyte2;
                System.out.println(lbyte1 + "*" + lbyte2 + " = " + n);
                int n1 = 100000, n2 = 200000;
                long p;
                p = n1 * n2;
                System.out.println(n1 + "*" + n2 + " = " + p);
        }
}
```

Le résultat de l'exécution de ce programme est :
50*100 = 5000
100000*200000 = -1474836480
Expliquer pourquoi ?

20. Soient ces déclarations :

```
char c = 60, ce = 'e', cg = 'g' ;
byte b = 10 ;
```

Donner le type et la valeur des expressions suivantes :

- c + 1
- 2 * c
- cg - ce
- b * c

21. Soit le code suivant :

```
class MyProg {
public static void main(String[] args) {
        if (args.length == 1 | args[1].equals("test")) {
                System.out.println("test case");
        } else {
                System.out.println("production " + args[0]);
        }
}
}
```

Si nous exécutions la commande : **java MyProg live2** que sera le résultat d'affichage ?

22. Soit le code suivant :

```
class MyProg {
public static void main(String[] args) {
        if (args.length == 1 | | args[1].equals("test")) {
                System.out.println("test case");
        } else {
                System.out.println("production " + args[0]);
        }
}
}
```

Si nous exécutions la commande : **java MyProg live2** que sera le résultat d'affichage ?

23. Qu'affiche le code ci-dessous ?

```
class ProgENSAH {
public static void main(String[] args) {
Integer i = 42;
String s = (i < 40) ? "al-hoceima" : (i > 50) ? "Imzouren" : "Beni Bouayach";
System.out.println(s);
}
}
```

24. Qu'affiche le code ci-dessous ?

```
class Ensah {
        int nbrStudents = 15;
        public static void main(String[] args) {
                final Ensah ensah1 = new Ensah();
                Ensah ensah2 = new Ensah();
                Ensah ensah3 = EnsahMethod(ensah1, ensah2);
                System.out.println((ensah1 == ensah3) + " " + (ensah1.nbrStudents ==
ensah3.nbrStudents));
        }
```

```
        static Ensah EnsahMethod(Ensah pEnsah1, Ensah pEnsah2) {
                final Ensah ensah = pEnsah1;
                ensah.nbrStudents = 16;
                return ensah;
        }
}
```

25. Que pouvez-vous dire de l'affectation de la valeur null lors de la déclaration de maVariable dans la classe ci-dessous :

```
public Class A{
  private String maVariable = null;
….
}
```

26. Un attribut, n'est ni public, ni protected, ni private, quelle est donc sa portée ?
27. Y'a-t-il une erreur dans la classe suivante ? Si oui laquelle ?

```
public class Toto {
 private String monAttrribut;
 public static String maMethode(){
  if ( monAttribut == null ){
  monAttribut = "Mon attribut";
  }
  return monAttrribut;
 }
```

28. Soit maChaine une variable String. De manière générale, vaut-il mieux utiliser : maChaine.equals("abc") ou bien "abc".equals(maChaine) ? pourquoi ?
29. Quel est le résultat de l'expression suivante : (obj1==obj2) où obj1 et obj2 sont deux objets.
30. A quoi sert le mot clé final en java ? donner une réponse détaillée en prenant en compte toutes ses possibilités d'utilisation.
31. Qu'est-ce qu'un Design Pattern ?
32. Qu'est-ce qu'un Design Pattern Singleton ? Comment peut-on le coder en Java ?
33. On considère la suite d'instructions suivantes :

```
A x,u,v ;
x=new A();
A y=x;
A f = x;
A z=new A();
```

Combien d'instances de la classe A sont créés ? Pourquoi ?

34. Pour la classe C définie comme suit:

```
class C {
public static int i;
public int j;
public C() {i++; j=i; }
}
```

Qu'affichera le code suivant ?
C x=new C(); C y=new C(); C z= x;

System.out.println(z.i + "et "+ z.j);

35. Quelle erreur a été commise dans cette définition de classe ?

```
class Math {
        private final float pi;
        private final int v = 10;
        private final int x;
        private final int z = v;
        public ClassA(float pPi) {
                pi = pPi;}}
```

36. Quel est le résultat de l'exécution du programme ci-dessous :

```
public class MyMath {
   public static int  add(int i, int j){
           System.out.println(1);
           return i+j;
   }
   public static double  add(double i, int j){
           System.out.println(2);
           return i+j;
   }
   public static double  add(double i, double j){
           System.out.println(3);
           return i+j;
   }
   public static float  add(float i, float j){
           System.out.println(4);
           return i+j;
   }
   public static double  add(double i, float j){
           System.out.println(5);
           return i+j;
   }
   public static double  add(float i, double j){
           System.out.println(6);
           return i+j;
   }
   public  static void main(String[] args) {
           float f =0.1f;long lLong =1;byte b =1;
           MyMath.add(1, 2);
           MyMath.add(1, 1.2);
           MyMath.add(1.1, b);
           MyMath.add(lLong, 1.2);
           MyMath.add(1*0.1, 1/2);
           MyMath.add(1.2, 1.2);
           MyMath.add(b, 1.2);
           MyMath.add(f, 1.2);
           MyMath.add(0.1*2/2,(float) 1);
           MyMath.add(1, f);
           MyMath.add(lLong, f);
   }
}
```

37. Quelles erreurs figurent dans la définition de la classe suivante ?

```
class ExempleSurdef {
   public void afficheInt(int n) {
System.out.println(n);
   }
   public int afficheInt(int p) {
System.out.println(p);
   }
   public void afficheFloat(final float x) {
System.out.println(x);
   }
   public void afficheFloat(double x) {
System.out.println(x);
   }
   public void afficheLong(long n) {
System.out.println(n);
   }
   public int afficheLong(final long p) {
System.out.println(p);
   }
}
```

38. Quel est le résultat de l'exécution du programme ci-dessous :

```
public class MyMath {
   public static double  add(double i, int j){
      System.out.println(2);
      return i+j;
   }
   public static double  add(double i, double j){
      System.out.println(3);
      return i+j;
   }
   public static float  add(float i, float j){
      System.out.println(4);
      return i+j;
   }
   public static double  add(byte i, double j){
      System.out.println(6);
      return i+j;
   }
   public  static void main(String[] args) {
      float f =0.1f;byte b =1;
      MyMath.add(b, f);
   }
}
```

39. Détecter les appelles incorrectes de la méthode add dans la méthode main de la classe suivante :

```
public class Calcul {
   public float add(int n, float x) {
      return n + x;
   }
   public static void main(String[] args) {
      Calcul lCalcul =  new MyMath();
      int n = 1,m = 1;
      byte b = 1;
      float x= 1.0f;
```

```
        double y= 1.2;
        lCalcul.add(n, x);
        lCalcul.add(b + 1, x);
        lCalcul.add(b, x);
        lCalcul.add(n, y);
        lCalcul.add(n, (float) y);
        lCalcul.add(n, 21 * x);
        lCalcul.add(n + 9, x + 0.21);
        lCalcul.add(0.2, x);
        lCalcul.add(m+n, x);
        lCalcul.add(n, x*(m+n));
    }
}
```

40. Détecter les appelles incorrectes de la méthode add dans la méthode main de la classe suivante :

```
public class MyCalcul {
    public float add(byte b, float x) {
        return b + x;
    }
    public static void main(String[] args) {
        MyCalcul lCalcul;
        int n;
        byte b;
        float x;
        lCalcul.add(b, x);
        lCalcul.add(b, 0.1f);
        lCalcul.add(b, 0.1 * n);
        b = 12;
        lCalcul.add(b, x);
        lCalcul.add(12, x);
    }
}
```

41. Si on transmet un paramètre int à une méthode et on modifie la valeur de ce paramètre dans la méthode, la variable int d'origine qui a été transmise reste inchangée. Expliquer pourquoi.
42. Est-il possible de forcer le Garbage collector en Java ? Si oui comment ?
43. Qu'affiche le programme :

```
public class Prog {
    private int i;
    public void test() {
            System.out.println(i);
    }
    public static void main(String[] args) {
            Prog p = new Prog();
            p.test();
    }
}
```

44. Qu'affiche le programme

```
public class Prog {

    public void test() {
            int i;
```

```
                System.out.println(i);
        }
        public static void main(String[] args) {
                Prog p = new Prog();
                p.test();
        }
}
```

45. Qu'affiche le programme ci-dessous ?

```
class Exemple2{
  public Exemple2() {
System.out.print("Bonjour ");
  }
  public Exemple2(int i) {
this();
System.out.println("ENSAH " + i);
  }
  public Exemple2(double i) {
this();
System.out.println("Imzouren " + i + 2 + 1);
 }
 public static void main(String[] args) {
Exemple2 lObj=new Exemple2(2011.);
 }
}
```

46. Lesquelles des méthodes ci-dessous respectent la norme JavaBeans ?
    a. addStudent
    b. getSize
    c. deleteUser
    d. isGreen
    e. putStudents

47. Soit le code ci-dessous :

```
public class Prog {
        public static void main(String[] args) {
                doWork(1);
                doWork(1, 2);
        }
        // ligne 6
}
```

Lesquelles des méthodes ci-dessous peuvent être insérées à la ligne 6 pour que le code compile ?

a. static void doWork (int... doArgs) { }
b. static void doWork (int[] doArgs) { }
c. static void doWork (int doArgs...) { }
d. static void doWork (int... doArgs, int y) { }
e. static void doWork (int x, int... doArgs) { }

48. Soit deux fichiers :

Fichier 1:

```
package ensah;
 public class TestEnsah {
```

```
 int a = 6;
 protected int b = 7;
 public int c = 8;
}
```

Fichier 2:

```
package fsth;
 import ensah.*;
 public class TestFSTH {
 public static void main(String[] args) {
   TestEnsah t = new TestEnsah ();
   System.out.print(" " + t.a);
   System.out.print(" " + t.b);
   System.out.print(" " + t.c);
 }
}
```

Qu'il est le résultat d'exécution ?

a. 6 7 8
b. 6 puis une exception
c. Erreur de compilation à la ligne 7
d. Erreur de compilation à la ligne 8
e. Erreur de compilation à la ligne 9
f. Erreur de compilation à la ligne 10

49. Soit le code ci-dessous :

```
class Prog {
        Prog tester(Prog cb) {
                cb = null;
                return cb;
        }
        public static void main(String[] args) {
                Prog c1 = new Prog();
                Prog c2 = new Prog();
                Prog c3 = c1.tester(c2);
                c1 = null;
                // do Work
        }
        }
```

Lorsque l'exécution arrive à //do Work, combien d'objets sont éligibles pour GC?

a. 0
b. 1
c. 2
d. Erreur de compilation
e. Impossible à savoir
f. Une erreur sera produite à l'exécution

50. Soit le code ci-dessous :

```
class Prog {
        Short test = 200;
        Prog tester(Prog cb) {
                cb = null;
                return cb;
```

```
        }
        public static void main(String[] args) {
                Prog c1 = new Prog();
                Prog c2 = new Prog();
                Prog c3 = c1.tester(c2);
                c1 = null;
                // do Work
        }
}
```

Lorsque l'exécution arrive à //do work, combien d'objets sont éligibles pour GC?

a. 0
b. 1
c. 2
d. Erreur de compilation
e. Impossible à savoir
f. Une erreur sera produite à l'exécution

51. Soit le code ci-dessous :

```
class Beta { }
class Alpha {
static Beta b1;
Beta b2;
 }
 public class Tester {
 public static void main(String[] args) {
 Beta b1 = new Beta(); Beta b2 = new Beta();
 Alpha a1 = new Alpha(); Alpha a2 = new Alpha();
 a1.b1 = b1;
 a1.b2 = b1;
 a2.b2 = b2;
 a1 = null; b1 = null; b2 = null;

 // do work
 }
 }
```

Lorsque l'exécution arrive à la ligne // do work , Combien d'objets seront éligible pour GC? Expliquer pourquoi ?

52. Soit le code ci-dessous :

```
class ProgENSAH {

        ProgENSAH test;
        ProgENSAH() {
        }
        ProgENSAH(ProgENSAH pTest) {
                test = pTest;
        }
        public static void main(String[] args) {
                ProgENSAH p2 = new ProgENSAH();
                ProgENSAH p3 = new ProgENSAH(p2);
                p3.doWork();
```

```
            ProgENSAH p4 = p3.test;
            p4.doWork();
            ProgENSAH p5 = p2.test;
            p5.doWork();
        }
        void doWork() {
            System.out.print("ENSAH ");
        }
}
```

Quel est le résultat de l'exécution ?

a. ENSAH
b. ENSAH ENSAH
c. ENSAH ENSAH ENSAH
d. Erreur de compilation
e. ENSAH, puis exception
f. ENSAH ENSAH, puis exception

53. Quelles erreurs ont été commises dans le début de programme suivant ?

```
public static void main (String args[]){
int n=10 ;
final int p=5 ;
int t1[] = {1, 3, 5} ;
int t2[] = {n-1, n, n+1} ;
int t3[] = {p-1, p, p+1} ;
int t4[] ;
t4 = {1, 3, 5} ;
float x1[] = {1, 2, p, p+1} ;
float x2[] = {1.25, 2.5, 5} ;
double x3[] = {1, 2.5, 5.25, 2*p} ;
```

54. On considère la méthode ci-dessous et le tableau int[] list = { 10, 20, 30, 40, 50 } :

```
public void mystery(int[] array) {
        int tmp = array[array.length - 1];
        for (int i = 1; i < array.length; i++) {
                array[i] = array[i - 1];
        }
        array[0] = tmp;
}
```

Qu'il est l'effet de l'appel mystery(list);

55. La classe ci-dessous est écrite pour implémenter une pile, mais elle contient une erreur qui pourra causer une fuite de mémoire, détecter cette anomalie et proposer la correction à faire.

```
/**
 * Classe qui modélise une pile
 * @author  Beginner Programmer
 *
 */
public class BeginnerStack {
   private Object[] elements;
   private int size = 0;

   public BeginnerStack(int initialCapacity){
          elements = new Object[initialCapacity];
   }

   public void push(Object e){
```

```
            guaranteeCapacity();
            elements[size++] = e;
        }

        public Object pop() throws EmptyStackException{
            if(size == 0) throw new EmptyStackException();
            Object result = elements[--size];
            return result;
        }

        /**
        * Permet de s'assurer qu'il y a pas de palce pour au moins un
    élément
        * supplémentaire en doublant la capacité à chaque accroissement de
    la                  taille du tableau
        */
    private void guaranteeCapacity(){
            if(elements.length == size){
                Object[] elementsAnciens = elements;
                elements = new Object[2* elements.length+1];
                System.arraycopy(elementsAnciens,0, elements, 0,size);
            }
        }
}
```

a- Mettre à jour le programme principal pour tester les nouvelles modifications.