Filière : DUT Info 2 Module 11 : Prog. POO(Java) Année Univ: 2019/2020 Pr. Said BENKIRANE	**Devoir Surveillé Module Partie I-(QCM) Durée 20 mn**	Université Cadi Ayyad École Supérieure de Technologie Essaouira

Nom :……..

Prénom :……..

1. La plateforme **Java ME** signifie :
   - ☐ Java Micro Edition
   - ☐ Java Mobile Edition
   - ☐ Java Maven Edition

2. Soit **b** une variable de type **long**, sa conversion en string est :
   - ☐ String(b)
   - ☐ Long.parseLong(b)
   - ☐ Long.valueOf(b)

3. a*=z+4 est équivalent à:
   - ☐ a=(a*z)+4
   - ☐ a=z+(a*4)
   - ☐ a=a*(z+4)

4. L'écriture suivante : "**DUT**" **instanceof Object**, retourne:
   - ☐ true
   - ☐ false
   - ☐ null

5. Une classe ***Wrapper*** permet de:
   - ☐ Convertir une variable de type primitif à un autre type primitif
   - ☐ Convertir une variable de type objet à un autre type objet
   - ☐ Convertir une variable de type primitif en un objet

6. Le ***foreach*** qui permet de parcourir un tableau tab de type entier s'écrit comme suit :
   - ☐ for(int i : tab)
   - ☐ for(int i=1:tab.length)
   - ☐ for(int i=1:tab.length:i++)

7. Appliquer à une chaine de caractère, la méthode **substring(i,j)** retourne :
   - ☐ Le nombre des caractères contenus entre la position i et la position j
   - ☐ Une sous chaine sans les caractères qui existent entre la position i à la position
   - ☐ Une sous chaine contenant des caractères de la position i à la position j

8. En java la librairie de classes utilisée par **défaut** est :
   - ☐ java.lang
   - ☐ java.util
   - ☐ java.io

9. Le modificateur d'accès **protected**, ne permet pas l'accès à:
   - ☐ Classes de même package
   - ☐ Classe (qui n'est pas une sous classe) dans un package diffèrent
   - ☐ Sous classe dans un package diffèrent

10. Dans la déclaration d'une méthode, le modificateur **final** indique que:
    - ☐ La valeur de retour de la méthode est fixe même si les valeurs d'entrées changent.
    - ☐ La méthode ne peut pas être redéfinie dans une sous-classe
    - ☐ La méthode contenant des constantes.

11. La redéfinition (**Overriede**) d'une méthode de la super-classe est :
    - ☐ La définition de la méthode en utilisant la même signature
    - ☐ La définition de la méthode en modifiant seulement le type de retour
    - ☐ La définition de la méthode en modifiant seulement le type de paramètres

12. Le **polymorphisme** est :
    - ☐ L'adaptation dynamique du comportement selon les objets en présence
    - ☐ L'héritage multiple
    - ☐ L'adaptation dynamique des attributs selon les objets en présence

13. Lorsqu'une classe (non abstraite) **implémente plusieurs interfaces**, elle doit :
    - ☐ Implémenter uniquement toutes les méthodes abstraites de la première interface
    - ☐ Implémenter uniquement toutes les méthodes abstraites de la dernière interface
    - ☐ Implémenter toutes les méthodes abstraites de chacune des interfaces

14. Les exceptions de type ***RuntimeException***, sont :
    - ☐ Des exceptions non-contrôlées et peuvent être ignorées
    - ☐ Des exceptions contrôlées et peuvent être ignorées
    - ☐ Des exceptions contrôlées et doivent être traitées

15. Si aucun bloc **catch** ne correspond au type d'exception qui a été levée :
    - ☐ L'exception est annulée
    - ☐ L'exception est propagée
    - ☐ L'exception est traitée

16. Le ***Layout Manager*** par défaut de conteneur **ContantPane** est :
    - ☐ GridLayout
    - ☐ BorderLayout
    - ☐ FlowLayout

17. Pour **centraliser** une fenêtre à l'écran, on utilise la méthode :
    - ☐ setSize()
    - ☐ setDefaultCloseOperation()
    - ☐ setLocationRelativeTo()

18. Pour utiliser **les boites de dialogues** de Swing, on doit importer le package :
    - ☐ JOptionPane
    - ☐ JOptionDialogue
    - ☐ JOptionBox

19. Pour gérer les événements déclenchés par l'activation d'un bouton, on utilise le **Listener** :
    - ☐ ActionListener
    - ☐ WindowListener
    - ☐ MouseListener

20. Dans le modèle **MVC**, le contrôleur :
    - ☐ Implémente les traitements
    - ☐ Correspond aux apparences
    - ☐ correspond au contenu

Filière : DUT Info 2 Module 11 : Prog. POO(Java) Année Univ: 2018/2019 Pr. Said BENKIRANE	**Devoir Surveillé Module Partie II Durée 1h**	Université Cadi Ayyad École Supérieure de Technologie Essaouira

Nom :……..

Prénom :……..

Exercice 1 : Quels résultats fournit le programme suivant?

```
public class Ex1{
public static void main (String[] args){
int i, n ;
for (i=0, n=0 ; i<5 ; i++) n++ ;
System.out.println ("A : i = " + i + ", n = " + n) ;
for (i=0, n=0 ; i<5 ; i++, n++) {}
System.out.println ("B : i = " + i + ", n = " + n);
for (i=0, n=50 ; n>10 ; i++, n-= i ) {}
System.out.println ("C : i = " + i + ", n = " + n) ;
for (i=0, n=0 ;i<3 ; i++, n+=i, System.out.println ("D : i = " + i + ", n = " +
n)) ;System.out.println ("E : i = " + i + ", n = " + n) ;}}
```

……
……
……
……
……
……
……
……
……

Exercice 2 : Quels résultats fournit le programme suivant?

```
class A{
public void  affiche() { System.out.print ("Je suis un A "); }}
class B extends A {}
class C extends A{
public void  affiche() { System.out.print ("Je suis un C "); }}
class D extends C {
public void  affiche() { System.out.print ("Je suis un D "); }}
class E extends B {}
class F extends C {}
public class Poly{
public static void main (String arg[]){
A a = new A() ; a.affiche() ; System.out.println() ;
B b = new B() ; b.affiche() ; a = b ; a.affiche() ; System.out.println() ;
C c = new C() ; c.affiche() ; a = c ; a.affiche() ; System.out.println() ;
D d = new D() ; d.affiche() ; a = d ; a.affiche() ; c = d ; c.affiche() ;
System.out.println() ;
E e = new E() ; e.affiche() ; a = e ; a.affiche() ;
b = e ; b.affiche() ; System.out.println() ;
F f = new F() ; f.affiche() ; a = f ; a.affiche() ; c = f ; c.affiche() ;}}
```

……
……
……
……
……
……
……
……
……
……
……
……

Exercice 3: **Soit le programme suivant :**

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class Mouse extends JFrame implements MouseListener{
    private JButton b1,b2,b3,b4;
    public Mouse(){
        this.setTitle("Fenetre2");
        this.setBounds(300, 200, 400, 100);
        this.setVisible(true);
        this.addMouseListener(this);
        this.getContentPane().setLayout(new FlowLayout());
        b1=new JButton("Premier");
        b2=new JButton("Suivant");
        b3=new JButton("Precedent");
        b4=new JButton("Dernier");
        this.getContentPane().add(b1);
        this.getContentPane().add(b2);
        this.getContentPane().add(b3);
        //this.getContentPane().add(b4);}
int count=1;
    @Override
    public void mouseClicked(MouseEvent me) {
        int x,y;
        x=me.getX();
        y=me.getY();
        count=me.getClickCount();
        System.out.println("j'ai cliqué:"+count+"Dans la position: "+x+ ",
"+y);count++;}
    public void mousePressed(MouseEvent me) {}
    public void mouseReleased(MouseEvent me) {}
    public void mouseEntered(MouseEvent me) {}
    public void mouseExited(MouseEvent me) {}
public static void main(String[] args){new Mouse();}}
```

Questions:

1- A quoi sert l'interface MouseListener ?
2- Expliquer la linge : `this.getContentPane().setLayout(new FlowLayout());`
3- Expliquer la ligne : this.setBounds(300, 200, 400, 100);
4- Quel résultat sera obtenu sur la console.
5- Dessiner la forme obtenue après exécution du programme.

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....