Nom : **Prénom :** **Groupe :**

Exo 1 : (QCM): Choisir **les** bonnes réponses (un ou plus). (08 pt)

1- Un constructeur est une méthode qui se caractérise par :
a- Exécuté lors de la création de l'objet. **b-** Porte le même nom que la classe.
c- N'a pas un type de retour.

2- L'instanciation d'un objet consiste à :
a – Réserver un espace mémoire pour cet objet.
b- Appeler le constructeur de cet objet. **c**- on doit utiliser le mot clé **this** pour le faire.

3- Pour un objet:
a- Deux variables peuvent pointer sur le même objet.
b- Un objet ne peut être pointé que par une seule variable **c**- un objet a une seule référence.

4- Une interface:
a- Peut avoir des méthodes abstraites et des méthodes normales.
b- Peut être implémenté par n'importe quelle classe.
c- Ne Peut être implémenté **que** par des classes de la même hiérarchie d'héritage.

5- On programmation orienté objet :
a- On ne peut pas créer un objet d'une classe qui est abstraite.
b- On ne peut pas hériter d'une classe qui est déclarée finale.
c- Une classe qui hérite d'une classe abstraite doit implémenter toutes les méthodes abstraites de la classe mère **Sinon** elle reste abstraite.

6- Un attribut qui ne présente aucun modificateur d'accès :
a- Est un attribut public. **b**- Est attribut accessible que par les sous classes de sa classe.
c- Est un attribut accessible que par les classes du même package.

7- Dans une classe :
a- On ne peut avoir qu'un seul constructeur (pas plus).
b- On peut avoir des méthodes publiques et des méthodes privées.
c- On peut avoir des attributs et des méthodes statiques.

8- par convention :
a- le nom de la classe commence par une majuscule.
b- le nom d'une méthode commence par une minuscule.
c- l e nom d'un attribut commence par une majuscule.

9- Si la classe mère possède deux constructeurs le premier sans paramètre et autre avec paramètre :
a- Le constructeur de sa classe fille peut faire un appelle explicite à un de ses deux constructeurs.
b- On Peut ne définir aucun constructeur pour la classe fille.
c- On obtient une erreur Si aucun appelle explicite est effectué au supère constructeur depuis le constructeur de la classe fille.

10- Une méthode d'un objet sert à représenter :
a- Ses composants. **b-** Ses états**.** **c**- Ses actions.

11- Les interfaces qui gèrent l'événement : click sur un bouton dans une interface graphique sont :
a- BortherLayout. **b-** ActionListener. **c-** JFrame

12- La machine virtuelle java:
a- Permet d'exécuter le code source java (fichier .java).
b- Eclipse a besoin d'installer une machine virtuelle java ou plus pour exécuter les programme.
c- Permet d'exécuter les fichiers bytecode (fichier .classe).

13- Dans une classe ; on peut utiliser une méthode d'une classe mère même si elle est redéfinie on utilisant :

a- Le mot clé super. **b**- le mot clé this.
c- Directement en utilisant le nom de la méthode.

14- Si **Etudiant** est une sous classe de **Personne** et on a : **Personne p = new Etudiant("mohamed")**;

a- L'instruction **(Etudiant)p).setNote(15);** est juste (pas d'erreur). (***setNote()*** est une méthode de la classe Etudiant).

b- L'instruction **p.identifier();** est juste (pas d'erreur). (***identifier()*** est une méthode de la classe Personne).

c- L'instruction **p.setNote(15);** est juste (pas d'erreur). (***setNote()*** est une méthode de la classe Etudiant).

Exo2

```
//I
public class Point {           // (01 pt)
	protected double x;
	protected double y;
	public Point(double x, double y) {
		this.x = x;
		this.y = y;
	}
	//2     (01.5 pt)
	public double getX() {
		return x;
	}
	public void setX(double x) {
		this.x = x;
	}
	public double getY() {
		return y;
	}
	public void setY(double y) {
		this.y = y; }
}
//II-
//3 (01 pt)
public class Carre {
	protected double cote;
	protected Point centre ;
	public Carre(double cote, Point centre) {
		this.cote = cote;
		this.centre = centre;
		// on peut aussi faire
		//this.centre = new Point(centre.getX(),centre.getY());
	}
	//4 (0.75 pt)

	public double surface(){
		return (cote*cote);
	}
	//5(0.75 pt)
	public boolean memeTaille(Carre c){
		return (cote == c.cote);
	}
	//6 (0.75 pt)

	public boolean memeTaille(Carre c1,Carre c2){
		return (c1.cote == c2.cote);
	}
	//7  (01 pt)

	public void deplacerVers(Point nouveauCentre){
		this.centre = nouveauCentre ;
		//on peut faire : centre.setX(P.getX()) ;    centre.setY(p.getY()) ;
	}
}
//III-
//8-        (.05 pt)
public class Cube extends Carre {
```

```java
	public Cube(double cote, Point centre) {
		super(cote, centre);
		}
	//9	(0.75 pt)
	public double volume(){
		return (cote*cote*cote);
	}
	//10	(0.75 pt)
	public double surface(){
		return (cote*cote*6);
	}
	}
// VI-
//11-
public class Prog {

	public static void main(String[] args) {
		//11 (01 pt)

		Point a = new Point(2, 3);
		Point b = new Point(5, 6);
		//12 (0.5 pt)

		Carre c1 = new Carre(4,a);
		//13 (0.75 pt)

		System.out.println("la surface de ce carré est : " + c1.surface());

	}
}

//V- (1.5 pt)

public class Carre {
	protected double cote;
	protected Point centre ;
	protected static double maxCote = 0;
	public CarreMax(double cote, Point centre) {
		this.cote = cote;
		this.centre = centre;
		if (this.cote > maxCote)
			maxCote = this.cote;
			}
	// Les autres méthodes ne se changent pas
}
```