NOM & PRENOM : .....

Groupe : .....

Note : ...../20

Exercice 1 : (06pts)

I. C'est quoi une classe abstraite et quel est son but (en détaille)

Une classe abstraite est une classe en programmation orientée objet (POO) qui sert de modèle de base pour d'autres classes, mais qu'on ne peut pas instancier directement.Une classe abstraite représente une idée générale ou un comportement commun

- Éviter la duplication du code
- Forcer une structure commune
- Créer une hiérarchie claire
- Faciliter le polymorphisme

- **Le code suivant ne s'exécute pas. Indiquer l'erreur et sa correction :**

`public class Compute {`
`  int somme(int a, int b);`
`}`
	Corps de méthode manquant, déclarer la classe et la méthode abstraite
	`abstract class Compute {`
	`abstract int somme(int a, int b);`
	`}`

- **Expliquez le concept de polymorphisme en programmation ainsi que ses différentes formes :**

Le polymorphisme est un concept fondamental de la programmation orientée objet (POO).
Le mot vient du grec :

- poly = plusieurs
- morphisme = formes

Donc :
Le polymorphisme signifie :
"un même élément peut prendre plusieurs formes."

Les types de plymorphisme : polymorphisme de compilation(overloading method and operator) et polymorphisme d'execution (methode overriding)

II. Les extraits des codes suivants ne s'exécutent pas. Pour chaque extrait indiquer c'est quoi l'erreur et donner une correction :

1	`class A{`
	`int a=5;`
	`A(int b){a=b; }`
	`}`
	
	`class B extends A{`
	`B(){a++; }`
	`}`
		Appel implicite au constructeur sans paramètres au niveau de la classe B qui n'existe pas dans la classe A
		
		`class B extends A{`
		`B(){super(a++); }`
		`}`

2	`int [] t={1,2,3};` `for (int i=0;i<=t.length;i++)System.out.print(t[i]);`	**Exception ArrayIndexOutOfBoundsException** `for(int i=0;i<t.length;i++)System.out.print(t[i]);`

3	`class C {`
	`private int a;`
	`C(){a++; }`
	`C(int x){ a=x;}`
	`}`
	
	`class E extends C{`
	`  E(int b){a=b++;}`
	`}`
		La variable a n'est pas accessible en dehors de la classe C (déclaré privé dans la classe C)
		• Soit rendre a visible
		• Soit :
		`class E extends C{`
		`  E(int b){super(b++);}`
		`}`

4	`class Animal{`
	`  void speak(){System.out.print("Animal makes sound");}`
	`}`
	`class  Cat extends Animal{`
	`void speak(){System.out.print("Cat makes sound");}`
	`Boolean wakeUp(){ return true;}`
	`}`
	`public static void main(String[] args) {`
	`  Animal Sissi=new Cat(); Sissi.wakeUp();`
	`}`
		La méthode **wakeUp()** n'est pas héritée de la classe Animal et appelé par une instance de la classe Animal (mère)
		
		transtyper (casting) l'instance Sissi en Cat comme suit :
		`((Cat)Sissi).wakeUp();`

5	class f{
	int a;
	static int augmenter(){
	a=a+2;
	}
	}
		Accès à un attribut d'objet,a (**non static**) par une méthode statique
		Soit mettre a **static** ou rendre la méthode augmenter **non static**

Exercice 2 (14 pts):

I. Classe NombreRational :

1. **Créer une classe nommée NombreRational, pour représenter des nombres rationnels (الاعداد الكسرية). La classe NombreRational devrait inclure des attributs pour le numérateur (البسط) et le dénominateur (المقام), une méthode static PGCD, qui renvoie le PGCD (Plus Grand Commun Diviseur) de deux nombres entiers, et des constructeurs pour initialiser des instances qui doivent être simplifiées utilisant le PGCD أ.ق.م,العدد الكسري يجب ان يكون مبسط باستعمال et si le dénominateur est nul, on le corrige en lui affectant la valeur 1.**

- **Le pgcd des deux nombres a et b est calculé comme suit :**

```
while (b != 0) {
        int r = a % b;
        a = b;
        b = r;
      }
int pgcd=a ;
```

```
public class NombreRational {
    int num;

    int denom;
```

.....

```
// PGCD method code here
static  int pgcd(int a, int b) {
       while (b != 0) {
          int r = a % b;
          a = b;
          b = r;
       }
       return a;
   }
//Constructors' code here
```

```
NombreRational(){
    num=1;
    denom=2;
  }
```

```
NombreRational(int a, int b){
    num=a;
    if (b!=0)denom=b;
    else denom=1;
      simplifier();
  }
```

```
private void simplifier(){
    int a=pgcd(num,denom);
   num=num /a;
   denom=denom/a;
  }
```

2. **La classe NombreRational contient des getters et des setters**

```
public int getDenom() {
return denom;
}
```

```
public int getNum() {
return num;
  }
```

```
public void setDenom(int denom) {
        if(denom!=0) this.denom = denom;
        else denom=1;
        simplifier();
    }
```

```
public void setNum(int num) {
     this.num = num;
     simplifier();
 }
```

3. Une méthode qui renvoie l'inverse (المقلوب) du nombre rational courant :

```
// inverse method code here

NombreRational inverse(){
          NombreRational a= new NombreRational();
           a.num=denom;
           a.denom=num;
  return a;                    //ou  return new NombreRational(denom, num);;
}
```

4. Des méthodes pour effectuer des opérations arithmétiques (addition, soustraction, multiplication, division) sur les nombres rationnels. Les méthodes renvoient comme résultats un nombre rational *simplifié*.

```
// addition method code here

NombreRational plus(NombreRational a){
         int newdenom=a.denom*denom;
         int newnum=a.num*denom+ num*a.denom;
        NombreRational b=  new NombreRational(newnum, newdenom);
        b.simplifier();
 return b;}


// subtraction method code here

        NombreRational soustraction(NombreRational a){
                 int M=a.denom*denom;
                 int B=( num*a.denom)-(a.num*denom);
                 NombreRational b=new NombreRational(B,M);
                  b.simplifier();
          return b;
 }

// multiplication method code here
        NombreRational multiplication(NombreRational a){
                 int newdenom=a.denom*denom;
                 int newnum=a.num* num;
                 NombreRational b=new NombreRational(newnum,newdenom);
                 b.simplifier();
          return b;
         }


// division method code here
               NombreRational Division(NombreRational a){
                    return multiplication(a.inverse());
                }
```

5. **display method (afficher le nombre sous sa forme rationnelle)**

```
void display(){
    System.out.println(num+"/"+denom);
}
```

6. **Une méthode decimalVal, qui renvoie la valeur décimale du nombre rationnel, (Sachant que La division de deux *int* donne un résultat entier plutôt qu'un nombre décimal)**

```
double decimalVal(){
      double d= (double) num/denom;
   return d;
 }
```

7. **Une méthose isEqual, qui permet de comparer le nombre courant et un autre nombre et renvoie true si les deux nombres ont la même valeur décimale**

```
boolean isEqual(NombreRational a){

    return a.decimalVal()== (double)num/ denom;

// if a.decimalVal()== (double)num/ denom return true
//else return false
}
```

}

II. **Créer une classe EntierRationnel hérité de la classe NombreRationnel, qui représente un entier sous la forme : $\frac{n}{1}$.**

```
public class EntierRationnel extends NombreRational{

      EntierRationnel (){  super(1,1);}

      EntierRationnel (int a, int b){ super(a,1);  }

      EntierRationnel (int a){ super(a,1);}

    @Override
    public void setDenom(int denom) {
        super.setDenom(1);
    }
}
```

III. **Créer une classe Pourcentage hérité de la classe NombreRationnel, qui représente une valeur sous la forme : $\frac{n}{100}$, en ajoutant les constructeurs nécessaires (NB. Le nombre ne doit pas être simplifié) et une méthode formatPourcentage() qui retourne une chaîne représentant le pourcentage.**

```
public class Pourcentage extends NombreRational{
     public Pourcentage() {
         super(1,100);
     }

  Pourcentage(int a){
        setNum(a);
        setDenom(100);
     }
  String formatPourcentage(){
         return num+"%";
            }}
```

```
Pourcentage(int a, int b){
        setNum(a);
        setDenom(100);
    }
 @Override
  public void setDenom(int denom) {
      super.setDenom(100);
  }
```

IV. Dans une classe de test, créer une liste des nombres rationaux.

```
import java.util.ArrayList;
import java.util.Random;

public class test  {

ArrayList <NombreRational> l=new ArrayList <NombreRational>();
```

1. **Remplir la liste de manière aléatoire (des nombres rationaux, des Pourcentages, des EntierRationnels), et de sorte que chaque élément soit égal à son indice augmenté de 2 et le dénominateur soit l'indice +3.**

```
Random r=new Random();

    for(int i=0;i<6;i++){
        int n=r.nextInt(3); System.out.println("n== "+n);
        if (n==0) l.add(new NombreRational(i+2,i+3));
        if (n==1) l.add(new EntierRationnel(i+2));
        if (n==2) l.add(new Pourcentage(i+2));
      }
```

2. **On considère que la liste représente un polynôme (كثير حدود) : l'indice représente la puissance de x et la valeur à cet indice correspond au coefficient du terme. Calculer la valeur du polynôme pour une valeur donnée du x.**

```
double v=0; int x=3;
      for(int i=0;i<l.size();i++){
            v=v+l.get(i).decimalVal()*Math.pow(x, i);
      }

System.out.println("v= "+v);
```

3. **Calculer le nombre des nombres EntierRationnels, pourcentage et rationnal dans la liste**

```
 int p=0; int e=0;int R=0;
 for(int i=0;i<l.size();i++){
     if (l.get(i) instanceof Pourcentage) p++;
     else if(l.get(i) instanceof EntierRationnel) e++;
     else R++;
 }
System.out.println("P==="+p+"e==="+e+"R==="+R);

}
```