

## Exercice 1

(dessiner l'arbre d'héritage pour bien comprendre !)

### Question 1:

hum !

Mes cher amis, bonjour!

[Erreur de compilation : La classe AdulteDistingue n'existe pas]

J'chui pas un bebe!

[Erreur de compilation : Ado n'est pas un Jeune]

Agheu, agheu!

Agheu, agheu!

J'chui pas un bebe!

### Question 2:

1. P1 = P2; //ok
2. P4 = P1; //ok
3. P3 = P4; //Erreur de compilation: une Personne n'est pas un Adulte
4. P3 = P1; //Erreur de compilation: une Personne n'est pas un Adulte
5. P4 = P5; //ok, si affectation de P5 ok
6. P7 = P6; //Erreur de compilation: un Enfant n'est pas un BebeCadum
7. P7 = (BebeCadum) P4; //Erreur d'execution: Bebe n'est pas un BebeCadum
8. P6 = (Bebe) P4; //ok
9. P3 = (AdulteDistingue) P2; //Erreur de compilation : La classe AdulteDistingue n'existe pas
10. P8 = (Bebe) P5; //Erreur d'execution: Ado n'est pas un Bebe

### Question 3:

a)

e[0] = (**Jeune**) P4;

e[1] = P5; //ok, si affectation de P5 ok.

e[2] = P6; //ok

e[3] = P7; //ok

e[4] = P8; //Erreur: e est de dimension 4

b)

J'chui pas un bebe!

hum! //si affectation de P5 ok

Agheu, agheu!

Agheu, agheu!

c)

P5 = e[0]; //ok

P4 = e[1]; //ok

P7 = (**BebeCadum**) e[2];

P8 = (**Enfant**) e[3];

P6 = e[4]; //Erreur: e est de dimension 4

## Exercice 2

### Question A:

```
public void dessineToi(){  
    for(int i=0; i<dessins.size(); i++)  
        ((Dessin) dessins.elementAt(i)).dessineToi();  
}
```

### Question B:

```
Rectangle r1 = new Rectangle(10,10,20,40);  
Rectangle r2 = new Rectangle(5,25,50,60);  
Ligne L1 = new Ligne(5,35,45,10);  
  
DessinComplexe dc1 = new DessinComplexe();  
dc1.ajoute(r1);  
dc1.ajoute(r2);  
dc1.ajoute(L1);
```

### Question C:

```
public class DoubleLigne extends Dessin {  
    Ligne L1;  
    Ligne L2;  
  
    public DoubleLigne(int x1,int y1, int x2, int y2, int x3, int y3){  
        super(x1,y1,x3,y3);  
        L1 = nex Ligne(x1,y1, x2,y2);  
        L2 = nex Ligne(x2,y2, x3,y3);  
    }  
  
    public void dessineToi(){  
        L1.dessineToi();  
        L2.dessineToi();  
    }  
}
```

**Question D:**

1)

Dans la classe DessinElementaire :

```
public int largeur() {
    int l= x1-x2;
    if(l<0) l=-l;
    return(l);
}

public int hauteur() {
    int v = y1-y2;
    if(v<0) v=-v;
    return(v);
}
```

Dans la classe DessinComplexe :

NB: un objet vide aura une taille infiniment petite (-1000000).

```
public int largeur() {
    int x_min= 1000 * 1000, x_max = 0;
    Dessin cur;

    for(int i=0; i<dessins.size(); i++){
        cur = (Dessin) dessins.elementAt(i);
        if(cur.x1<x_min) x_min=cur.x1;
        if(cur.x1>x_max) x_max=cur.x1;
        if(cur.x2<x_min) x_min=cur.x2;
        if(cur.x2>x_max) x_max=cur.x2;
    }

    return(x_max-x_min);
}

public int hauteur() {
    int y_min= 1000 * 1000, y_max = 0;
    Dessin cur;

    for(int i=0; i<dessins.size(); i++){
        cur = (Dessin) dessins.elementAt(i);
        if(cur.y1<y_min) y_min=cur.y1;
        if(cur.y1>y_max) y_max=cur.y1;
        if(cur.y2<y_min) y_min=cur.y2;
        if(cur.y2>y_max) y_max=cur.y2;
    }

    return(y_max-y_min);
}
```

2)

Dans la classe Dessin:

```
public void calculCentre(){
    cx = (x1 + x2) / 2;
    cy = (y1 + y2) / 2;
}
```

=> Cette méthode n'a pas d'effet dans la classe DessinComplexe car x1,x2,y1 et y2 n'y sont pas initialisé !, Ceci démontre qu'il y a une erreur dans l'énoncé, il faut en fait réécrire la méthode **ajoute** dans la classe DessinComplexe :

```
public void ajoute(Dessin d){
    dessins.addElement(d);
    //x1 sera l' x minimum et x2 l' x maximum
    if(d.x1<x1) x1=d.x1;
    if(d.x2<x1) x1=d.x2;
    if(d.x1>x2) x2=d.x1;
    if(d.x2>x2) x2=d.x2;
    //y1 sera l' y minimum et y2 l' y maximum
    if(d.y1<y1) y1=d.y1;
    if(d.y2<y1) y1=d.y2;
    if(d.y1>y2) y2=d.y1;
    if(d.y2>y2) y2=d.y2;
}
```

Ceci implique de définir un constructeur pour DessinComplexe, c'est d'ailleurs obligatoire car il n'existe pas de constructeur vide dans la classe Dessin :

```
public DessinComplexe() {
    super(1000*1000,1000*1000,0,0);
}
```

Il faut aussi que les attributs x1,x2,y1 et y2 de la classe Dessin soient déclarés **protected** et non pas **private**, pour permettre leur exploitation dans la nouvelle méthode **ajoute**, (ou alors il faut rajouter des accesseurs en lecture sur ces attributs).

```
public class Dessin {
    protected int x1,x2,y1,y2;

    //...//
}
```