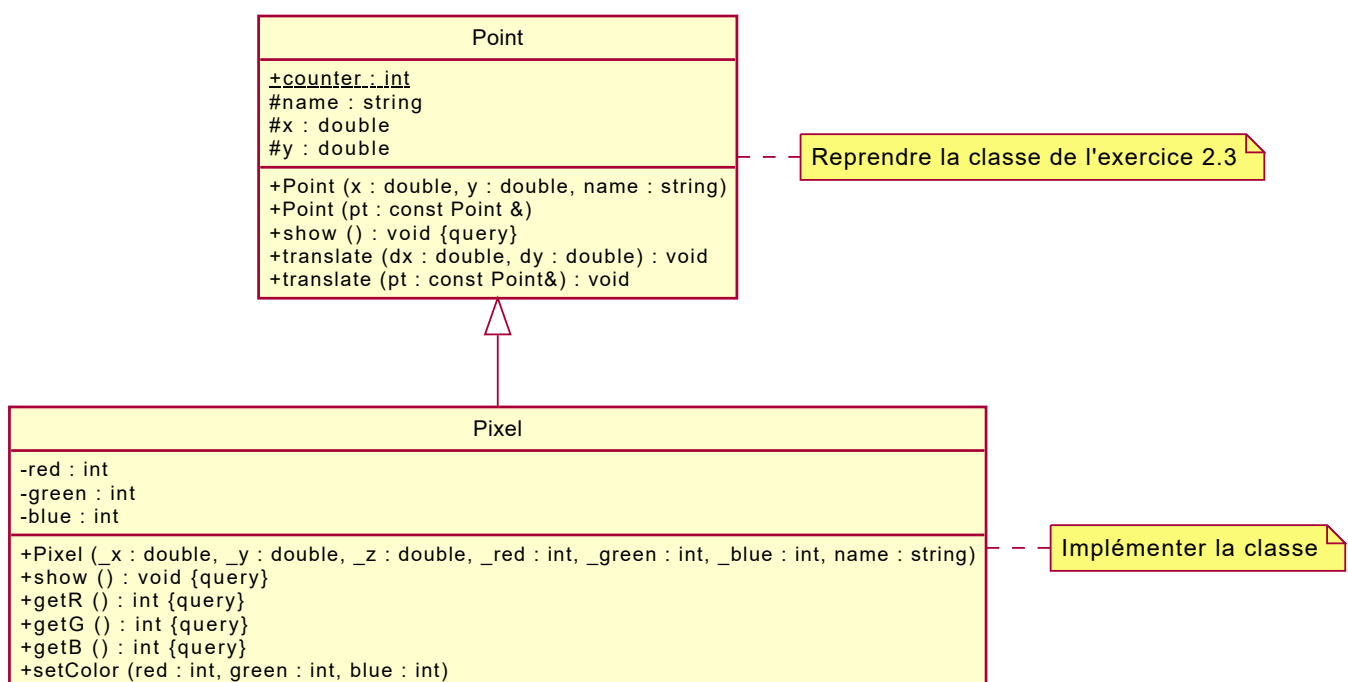


Serie 4.1

Exercice 1 : Héritage simple

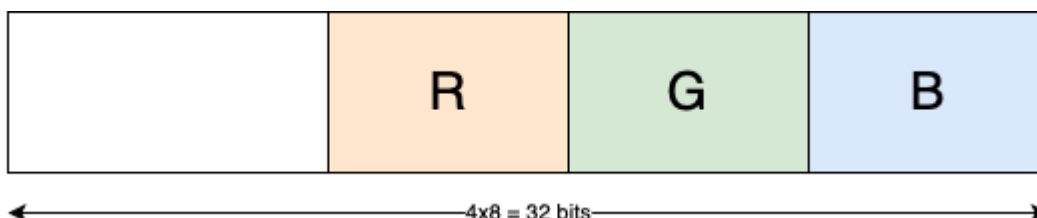
Ecrire une classe **Pixel** permettant de manipuler des points colorés. Une couleur sera représentée par 3 entiers (voir schéma UML), chacun compris entre 0 et 255. Cette classe disposera de la méthode `show()` qui affichera les coordonnées et la couleur d'un point. Cette classe devra mettre en œuvre la notion d'héritage, en utilisant la classe `Point` définie dans la série 2-3.

On donne aussi le diagramme de classe des classes *Point* et *Pixel* :



Option

La couleur pourra être codée sur un seul int. Les huit bits de poids faible contenant le bleu, les huit bits suivants le vert et les huit suivants le rouge.



Exercice 2 : Héritage - méthode virtuelle

Quel sera l'affichage lors de l'exécution du programme ci-dessous ?

```
class Point
{
protected:
    double x = 0., y = 0.;
public :
    Point (double abs=0, double ord=0) { x=abs ; y=ord ; }

    virtual void show ()
    {
        cout << "I'm a point. \n" ;
        cout << " my coordinates are : " << x << ", " << y << "\n" ;
    }
} ;

class Pixel : public Point
{
    short color = 0;
public :
    Pixel (double _x=0., double _y=0., short cl=1) : Point (_x, _y)
    { color = cl ; }

    virtual void show () override
    {
        cout << "I'm a pixel. \n" ;
        cout << "    my coordinates are : " << x << ", " << y ;
        cout << "    and my color is :    " << color << "\n" ;
    }
};
```

```
int main()
{
    Point p(3., 5.);
    Point * adp = &p;
    Pixel pc(8., 6., 2);
    Pixel * adpc = &pc;

    adp->show();
    adpc->show();
    cout << "-----" << endl;
    adp = adpc;
    adp->show();
    adpc->show();
    return 0;
}
```

Exercice 3 - Méthode virtuelle 2

Quel sera l'affichage lors de l'exécution de du programme suivant :

```
class Point
{
protected:
    double x = 0., y = 0;
public :
    Point (double abs=0, double ord=0) { x=abs ; y=ord ; }

    void show ()
    {
        identify();
        cout << "    my coordinates are : " << x << ", " << y ;
    }
    virtual void identify ()
    {
        cout << "I'm a point. \n" ;
    }
};

class Pixel : public Point
{
    short color = 0;
public :
    Pixel (double _x=0., double _y=0., short cl=1) : Point (_x, _y)
    { color = cl ; }

    virtual void identify () override
    {
        cout << "I'm a pixel of color: \n" << color << "\n";
    }
};
```

```
int main()
{
    Point p(3,4);
    Pixel pc(5,9,5);

    p.show();
    pc.show();
    cout << "-----\n";
    Point * adp = &p;
    Pixel * adpc = &pc;

    adp->show();

    adpc->show();
    cout << "-----\n";

    adp = adpc;

    adp->show();
    adpc->show();

    cout << "\nPress any key to continue... ";
    cin.get();
    return 0;
}
```