Cours de C++, 1ère année, HE-Arc

Serie 5.1

Exercice 1 : RTTI

On aimerait pouvoir comparer des objets afin de savoir s'ils sont de la même classe et si leur contenu est identique.

1. Reprendre le projet de la série 4.2 concernant les figures.
2. Implémentez la méthode suivante `bool compareShapes(Figure *fig1, Figure *fig2)` qui retoune vrai si:
   - Les objets sont des instances de la même classe (utilisez typeid)
   - Le point pos définit dans `Figure` est le même pour les deux objets.
3. Le main() a l'aspect suivant:

```
int main()
{

    Rectangle r1(Point( 1,  2),  4.0, 10.0);
    Rectangle r2(Point( 1,  2),  4.0, 10.0);
    Rectangle r3(Point(10, 20), 10.0, 20.0);
    Circle    c1(Point(1.1, 5.3), 5.0);

    cout << endl << "Comparison of r1 and r2" << endl;
    areTheSame = compareShapes(&r1, &r2);
    cout << endl << "r1 and r2 are the same : " << boolalpha << areTheSame <<endl;

    cout << endl << "Comparison of r1 and r3" << endl;
    areTheSame = compareShapes(&r1, &r3);
    cout << endl << "r1 and r3 are the same : "<< boolalpha << areTheSame <<endl;

    cout << endl << "Comparison of r1 and c1" << endl;
    areTheSame = compareShapes(&r1, &c1);
    cout << endl << "r1 and c1 are the same : "<< boolalpha << areTheSame <<endl;
}
```

Le résultat :

```
Comparison of r1 and r2
     Comparison of an objet 9Rectangle with an objet 9Rectangle
     -  typeid are the same
     -  positions are the same
r1 and r2 are the same: true

Comparison of r1 and r3
     Comparison of an objet 9Rectangle with an objet 9Rectangle
```

```
     - typeid are the same
     - positions are different
r1 and r3 are the same: false

Comparison of r1 and c1
     Comparison of an objet 9Rectangle with an objet 6Circle
     - typeid are different
r1 and c1 are the same: false
```

Exercice 2 : Transtypage dynamique

On aimerait pouvoir accéder à une méthode spécifique à une des classe dérivée, sans connaitre le pointer sur cette classe, mais uniquement le pointeur sur la classe mère.

1. Reprendre le projet de la série 4.2 concernant les figures.
2. Ajouter la méthode `getRadius()` à la classe `Circle`
3. Le code ci-dessous ne fonctionne pas car la méthode n'existe pas dans les classes `Figure`, `Rectangle` et `Triangle`.

```
int main()
 Figure *myShapes[3];

 myShapes[0] = new Circle(Point(1.1, 5.3), 5.0);
 myShapes[1] = new Triangle(Point(2, 2), Point(10, 3), Point(-1, -1));
 myShapes[2] = new Rectangle(Point(4, 2), 4.0, 10.0);

 cout << "------- List of shapes -----------" << endl;
 for (auto shape : myShapes)
 {
     shape->show();
     //shape->getRadius();
 }
 return 0;
}
```

A vous d'améliorer ce code afin qu'il offre la sortie suivante:

```
Cercle:
   Point : (1.1, 5.3), radius=5
>>>>>>>>>>>>>>> The radius is: 5    (obtenu au moyen de getRadius()

Triangle:
  Point : (2, 2)
  Point : (10, 3)
  Point : (-1, -1)

Rectangle:
  Point : (4, 2), l = 10, h = 4
```