classe Cercle2D

déclarer une classe Cercle2D représentant un cercle dans le plan dont le centre est une Position2D, fiche précédente, le rayon un entier >0, avec:

1. un constructeur par défaut qui initialise le centre à (0, 0) et le rayon à 1.
2. un constructeur permettant de spécifier les deux coordonnées du centre et le rayon (si le rayon n'est pas spécifié, il sera égale à 1 - argument par défaut).
3. un constructeur permettant de spécifier la position du centre et le rayon (si le rayon n'est pas spécifié, il sera égale à 1 - argument par défaut).
4. une méthode affiche qui écrit sur le terminal le cercle sous la forme (x, y, r); ex: **`(120, 230, 30)`**
5. une méthode qui indique si le cercle contient, ou pas, une position passée en argument.
   prototype: **`bool Cercle2D::contient(Position2D);`**
6. la surcharge des opérateurs relationnels (> < ≥ ≤) qui comparent le cercle à un autre, passé en paramètre - la comparaison se fera sur leurs rayons respectifs.
   prototype: **`bool Cercle2D::operator >(Cercle2D);`**
7. une méthode aire qui calcule l'aire du cercle, prototype: **`double Cercle2D::aire();`**
8. une méthode touche qui indique si un cercle touche un autre cercle passé en argument.
   prototype: **`bool Cercle2D::touche(Cercle2D);`**

Marche à suivre:

comme précédemment, déclarer les méthodes de la classe les unes après les autres, en vérifiant au fur et à mesure leur bon fonctionnement dans le programme principal.

Après avoir dupliqué votre programme (enregistrer sous…) dans un autre fichier, remplacer votre programme principal par le suivant :

```
int main(int argc, char **argv)
{
Position2D p1(100, 100), p2;
Cercle2D c1, c2(200, 100, 10), c3(-200, 100), c4(p1, 30), c5(p1, -20);

// affichage des deux positions et des cinq cercles initiales
cout<<"p1:";p1.affiche();
cout<<"p2:";p2.affiche();
cout<<"c1:";c1.affiche();
cout<<"c2:";c2.affiche();
cout<<"c3:";c3.affiche();
cout<<"c4:";c4.affiche();
cout<<"c5:";c5.affiche();

cout<<endl;
if (c2.contient(p1))cout<<"c2 contient p1";
else cout<<"c2 ne contient pas p1";
cout<<endl;
if (c1<=c3 && c4>c2) cout<<"la condition est vérifiée !"<<endl;
cout<<endl;

cout<<"après traitement"<<endl;
cout<<"p1:";p1.affiche();
cout<<"p2:";p2.affiche();
cout<<"c1:";c1.affiche();
cout<<"c2:";c2.affiche();
cout<<"c3:";c3.affiche();
cout<<"c4:";c4.affiche();
cout<<"c5:";c5.affiche();

cout<<"aire de c5 = "<<c5.aire()<<endl;
return 0;
```

```
}
```

Si vous avez bien respecté le cahier des charges, vous devez impérativement obtenir :

```
1:(100,100)
p2:(0,0)
c1:(0,0,1)
c2:(200,100,10)
c3:(0,100,1)
c4:(100,100,30)
c5:(100,100,1)

c2 ne contient pas p1
la condition est vérifiée !

après traitement
p1:(100,100)
p2:(0,0)
c1:(0,0,1)
c2:(200,100,10)
c3:(0,100,1)
c4:(100,100,30)
c5:(100,100,1)
aire de c5 = 3.14159
```

Représentation UML

En UML, ces deux classes seraient représentées sous la forme suivante. La relation de composition traduit le fait qu'un Cercel2D contient une Position2D. (notez bien la forme et le sens de la flèche!)

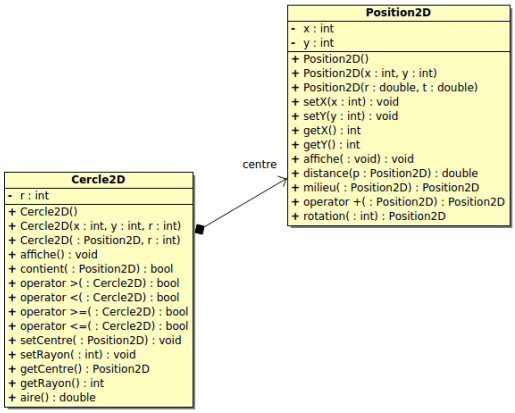

Figure 1 - représentation UML des classes

Chaque binôme devra envoyer par email une archive (.zip) contenant le ou les fichiers constituant le programme complet du TP, et ce, impérativement avant le jour de la séance suivante.

Si le programme qui ne se compile pas (présence d'erreurs de syntaxe) la note de compte rendue ne sera donnée que sur la moitié des points prévus.

Si vous n'avez pas pu terminer le programme durant la séance, charge à vous de le terminer durant la semaine.

La présentation et les commentaires dans les programmes seront très appréciés.