comme vous avez pu le remarquer dans les TP précédents, les classes Cercle2D, Carre2D, Rectangle2D et Hexagone2D ont des parties de code qui sont très similaires.
Imaginons maintenant que l'on veuille pouvoir déplacer toutes ces figures ou les dessiner graphiquement sur une fenêtre, on voit bien qu'il faudra re-modifier toutes les classes une par une.
La programmation orientée objet à prévu cette situation avec le concept **d'héritage**. On peut établir une relation d'héritage entre deux classes A et B lorsque que l'on peut dire "A est un.e B" ou "B est un.e A".
exemple : si nous devions programmer un jeu de rugby, et que nous ayons les classes suivantes - en POO, un concept unique par classe. Représenter les relations d'héritage possible sur le diagramme ci-dessous :

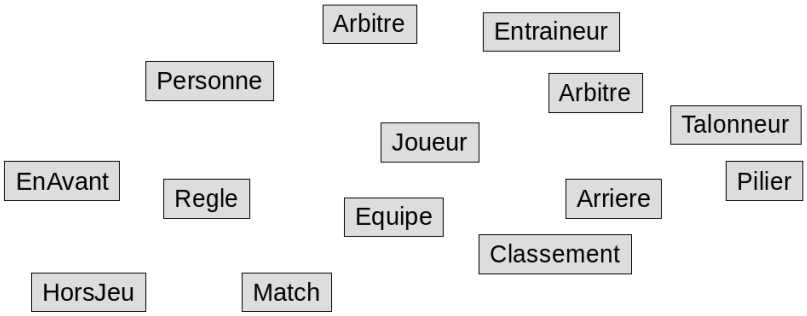

Figure 1 - Relations d'héritage

On propose de reprendre tous les codes en introduisant une classe mère Element2D, et des classes filles comme le montre le diagramme UML suivant (on utilisera un héritage public). On introduit au passage une classe Couleur (un Element2D contient une Couleur).
Notez bien la forme et le sens des flèches de composition ("contient") et d'héritage ("est un.e") et la visibilité des attributs *protected*

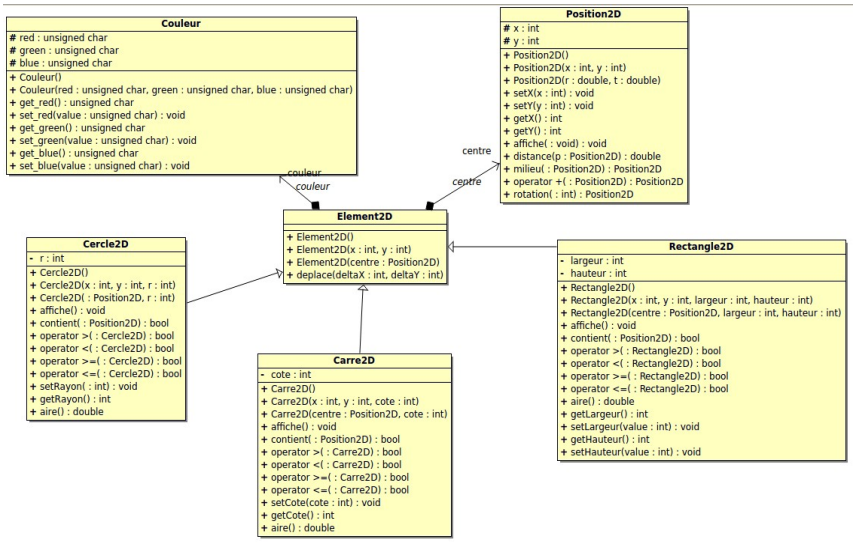

Figure 2 - Représentation UML des classes

Déclarer et implémenter ces classes, en les testant et les validant au fur et à mesure.

Chaque étudiant 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.