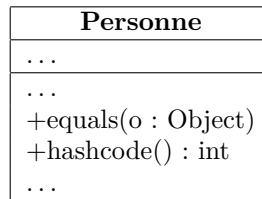


## UE Conception Orientée Objet

### Les données

**Personne** On suppose que l'on dispose d'une classe *Personne*, dont la structure minimale est la suivante :



**Epreuves sportives** On s'intéresse à la modélisation d'*Epreuves* sportives. Une épreuve sportive est caractérisée par des participants dont on peut enregistrer le résultat à l'épreuve. Il est bien évidemment possible :

- d'ajouter un participant à une épreuve, méthode *void addParticipant(Personne p)*,
- d'enregistrer un résultat pour un participant donné, méthode *void ajouteResultat(Personne participant, String res) throws ParticipantInconnuException, BadResultatFormatException*, on peut remarquer que le résultat est donné sous la forme d'une chaîne de caractères et une exception est levée si la chaîne de caractères est mal formée ou si le participant n'est pas enregistré dans l'épreuve.
- de connaître le résultat d'un participant, méthode *Resultat getResultat(Personne p)*,
- de connaître le meilleur résultat connu pour l'épreuve, méthode *Resultat getMeilleurResultat()*,
- d'accéder à l'ensemble des participants, méthode *Iterator<Personne> lesParticipants()*.

**Résultats.** Les résultats sont gérés par les types *Resultat* dont la hiérarchie est donnée à la figure 1. De plus, cette interface étend l'interface *Comparable<Resultat>*.

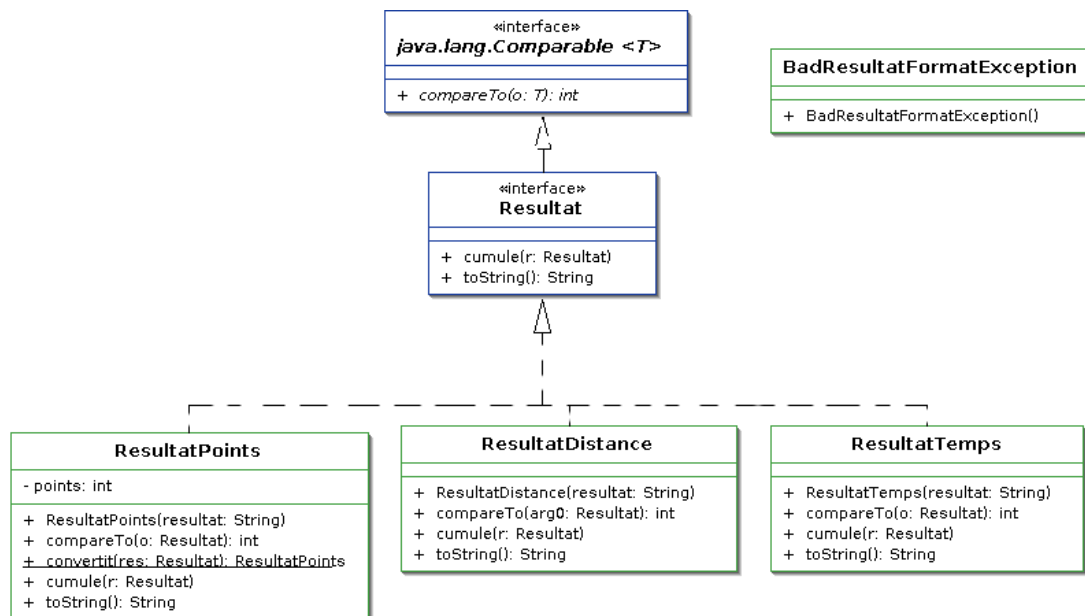


FIG. 1: La hiérarchie *Resultat*

Votre attention est attirée sur les constructeurs qui prennent en paramètre une chaîne de caractères, pour ces constructeurs une exception *BadFormatResultatException* est levée si la chaîne ne respecte pas le format attendu.

On peut également obtenir une instance de *ResultatPoints* à partir d'un objet *Resultat* via la méthode statique *convertit*. L'objectif de cette dernière est notamment de permettre la conversion d'un résultat en temps ou en distance en un nombre de points.

**Les différentes épreuves** On distingue différentes types d'épreuves. Parmi les épreuves certaines sont des épreuves d'*Athlétisme* et d'autres de *Natation* :

- des *Lancer*, comme le disque ou le javelot, tous les lancers sont des épreuves d'athlétisme,
- des *Saut*, comme la perche ou la longueur, tous les sauts sont des épreuves d'athlétisme,
- des *Course*, comme le Marathon ou le 400m 4 nages, on distingue les courses d'athlétisme, des courses de natation. Les courses sont caractérisées par une distance.
- des *Plongeon*, comme le tremplin à 10m, tous les plongeurs sont des épreuves de natation,
- des épreuves multiples appelées *NATHlon*, comme le décathlon ou le duathlon de Lille qui regroupe une course à la nage et une course à pied.

La nature du résultat dépend de l'épreuve : une distance pour les lancer et les sauts, un temps pour les courses et un nombre de points pour les plongeurs ou les n-athlons.

C'est la méthode *ajouteResultat* qui permet d'attribuer un résultat à un participant.

**N-athlons** Les *NATHlon* sont des épreuves composées d'autres épreuves. On peut ajouter une épreuve à un *NATHlon* et avoir un accès à l'ensemble des épreuves.

Pour un objet *NATHlon* la méthode *ajouteResultat* n'a pas d'effet, les résultats sont en fait ajoutés aux épreuves qui le composent. Ajouter un participant à un *NATHlon* revient à l'ajouter à chacune des épreuves qui composent ce *NATHlon*, la cohérence est maintenue à l'ajout d'un nouveau participant mais aussi lors de l'ajout d'une nouvelle épreuve.

Pour un objet *NATHlon* La méthode *ajouteResultat* n'a aucun effet pour un n-athlon. Chaque résultat est en fait ajouté à l'épreuve du n-athlon auquel il correspond. Pour la méthode *getResultat*, les résultats à chacune des épreuves qui composent le n-athlon sont convertis en nombre de points et cumulés pour déterminer le résultat au n-athlon. Une épreuve pour laquelle on n'a pas le résultat vaut 0.

## Les questions

**Q 1 .** Donnez un diagramme UML **clair** de la modélisation des différentes épreuves mentionnées précédemment. On doit voir apparaître **au minimum** dans votre diagramme les différents types suivants : *Epreuve*, *Athletisme*, *Natation*, *Saut*, *Lancer*, *Course*, *Plongeon*, *NATHlon*.

Dans le diagramme vous ferez apparaître : les attributs, les constructeurs, les méthodes avec leurs paramètres et leurs types et l'éventuel type de retour.

**Q 2 .** Donnez le code de la méthode *ajouteResultat* pour le type *Epreuve* ainsi que pour les types *Lancer* et *Course*.

**Q 3 .** Donnez le code JAVA de la classe *NATHlon*.