Travaux pratiques d'informatique N° 23

Le but principal de cette séance est de vous permettre d'améliorer vos connaissances concernant la programmation générique, en général, et les aspects suivants, en particulier : la définition et l'instanciation des classes génériques, la définition et l'appel des méthodes génériques, les limitations des variables de type, la généricité et l'héritage, la généricité et le polymorphisme, ainsi que l'utilisation de l'interface prédéfinie ***List<E>***, de la classe générique prédéfinie ***ArrayList<E>*** et de la classe prédéfinie ***Collections***.

1. Ecrire un projet contenant deux classes nommées ***Paire*** et ***CP_TP23Exo1***.

La classe publique ***Paire*** :

- est une classe générique avec une seule variable de type notée **T** et qui permet de regrouper deux éléments du même type ;
- a deux champs privés nommée ***premier*** et ***second*** dont le type est générique ;
- définit des méthodes ***getters*** et ***setters*** appropriées ;
- définit trois constructeurs (un premier sans argument, un deuxième avec deux arguments de type générique et un troisième par recopie) ;
- définit une méthode d'instance publique, nommé ***choisir***, sans argument et qui retourne un des deux éléments de la paire choisi au hasard.

La classe publique ***CP_TP23Exo1*** :

- définit une méthode publique et statique nommée ***choisir*** qui :
  - est générique avec une seule variable de type nommée ***T*** ;
  - a deux arguments de type générique ;
  - retourne un résultat de type générique qui correspond à un de ses deux arguments choisi au hasard ;
- définit une méthode standard ***main*** qui permet de tester l'utilisation de la classe générique ***Paire*** (y compris de la méthode ***choisir***) et de la méthode statique locale ***choisir***.

2. Ecrire un projet contenant les classes suivantes :

- la classe générique ***Solo*** (avec une seule variable de type notée ***T*** et limitée par dérivation par la classe prédéfinie ***Number***) ;
- la classe générique ***Paire*** (avec une seule variable de type notée ***T*** et limitée par dérivation par la classe prédéfinie ***Number***) ;
- la classe générique ***Couple*** (avec deux variables de type notées ***T*** et ***U***) ;
- l'interface générique ***ICalculable*** qui :

- utilise une seule variable de type notée ***T*** et limitée par dérivation par la classe prédéfinie ***Number*** ;
- contient une seule méthode appelée ***calculerAire***, sans argument et qui retourne un résultat de type générique ***T*** ;

➢ la classe (ordinaire) **Cercle** qui :

- implémente une instanciation de l'interface générique ad-hoc ***ICaclulable*** pour le type "concret" ***Double*** et une instanciation de l'interface générique prédéfinie ***Comparable*** pour le type "concret" ***Cercle*** ;
- définit un champ privé de type instanciation de la classe générique ***Solo*** pour le type "concret" ***Double*** ;
- redéfinit convenablement la méthode ***toString( )*** ;

➢ la classe (ordinaire) ***Ellipse*** qui :

- hérite d'une instanciation de la classe générique ***Paire*** pour le type "concret" ***Double*** ;
- implémente une instanciation de l'interface générique ad-hoc ***ICaclulable*** pour le type "concret" ***Double*** et une instanciation de l'interface générique prédéfinie ***Comparable*** pour le type "concret" ***Ellipse*** ;
- définit un champ propre et privé de type ***Color*** nommé ***couleur*** ;
- redéfinit convenablement le méthode ***toString( )*** ;

➢ la classe principale ***CP_TP23Exo2*** qui définit plusieurs méthodes publiques, à savoir :

- une méthode statique nommée ***minMax*** qui a comme seul argument une liste de type ***List*** d'ellipses et retourne un résultat de type ***Couple*** d'ellipses qui regroupe, dans cet ordre, la "plus petite" ellipse et la "plus grande" ellipse de la liste ;
- une méthode statique nommée ***coupler*** qui a un seul argument de type ***Ellipse*** et retourne un résultat de type ***Couple*** d'une ***Ellipse*** et d'une référence ***Double*** qui correspondent à l'ellipse argument et à son aire ;
- une méthode statique nommée ***inverser*** qui a un seul argument de type ***Couple*** d'une ***Ellipse*** et d'une référence ***Double*** et qui retourne un résultat de type ***Couple*** d'une référence ***Double*** et d'une ***Ellipse***, obtenu en inversant les deux éléments de l'argument ;
- une méthode statique ***afficher*** de type ***void***, qui a comme seul argument une liste ***List*** de couples (où chaque élément est un ***Couple*** d'une ***Ellipse*** et d'une référence ***Double***) et qui affiche des informations contenues dans la liste argument ;
- la méthode "standard" ***main*** qui permet de tester les autres éléments du projet.