TP n°5 : classes, héritage, composition en C++

En cas de difficulté, demandez un indice ou une explication à l'enseignant. Chaque exercice doit faire l'objet d'une **validation expérimentale poussée**.

Exercice 1 (acquisition)

Vous devez réaliser une classe **PileInt** destinée à implanter une pile d'entiers représentée à l'aide d'un tableau.

La classe **pileInt** doit disposer d'une constructeur prenant en paramètre la capacité de la pile à créer, ainsi que d'une méthode **void push(int x)** permettant d'empiler un entier x et d'une méthode **int pop()** permettant de dépiler un entier.

Les erreurs (dépassement de capacité, dépiler dans une pile vide) seront gérées par affichage d'un message et arrêt du programme.

Compétence : **classes**.

Exercice 2 (consolidation)

Vous devez améliorer la classe **PileInt** de l'exercice 1 en lui ajoutant deux méthodes privées : **void moresize(int d)** et **void lessSize(int d)**, qui augmente et diminue respectivement la capacité de la pile, ainsi qu'un mécanisme d'ajustement automatique de cette capacité en cours d'utilisation.

L'appel de **push** avec une pile pleine doit augmenter de 10 sa capacité, tandis que l'appel de **pop** avec une pile dans laquelle il reste plus de 10 places doit réduire de 10 sa capacité.

Compétence : **classes**.

Exercice 3 (acquisition)

Vous devez réaliser une petite application de gestion d'un carnet de rendez-vous. Chaque rendez-vous a un objet (i.e., une chaîne de caractères indiquant de quoi il s'agit) et occupe une plage horaire caractérisée par une date, une heure de début et une heure de fin. Les heures de début et de fin sont spécifiées à la minute près. Le carnet de rendez-vous, ou agenda, représente une liste de rendez-vous (représentée par un tableau) et doit, dans sa version initiale, permettre d'ajouter de nouveaux rendez-vous (avec une méthode **add**) et d'afficher tous les rendez-vous déjà mémorisés.

Vous devez bien évidemment décomposer autant que possible votre application en classes et utiliser des variables d'instance privées.

Compétence : **héritage et composition**.

Exercice 4 (consolidation)

Vous devez améliorer l'application d'agenda de l'exercice 3 en la dotant de fonctionnalités supplémentaires (à implanter et tester séparément, une à la fois) :

- Vérifier l'absence de chevauchement de rendez-vous dans l'agenda.
- Calculer la durée cumulée des rendez-vous programmés dans une période spécifiées par deux dates données.
- Retirer un rendez-vous de l'agenda.
- Retirer de l'agenda tous les rendez-vous situés avant une date donnée.
- Rechercher, dans une période donnée, la date et l'heure auxquelles un rendez-vous d'une durée donnée peut être insérer au plus tôt sans créer de chevauchement.
- Réaliser un affichage de l'occupation d'une période donnée (spécifiée par une date de début et une date de fin) en représentant par un 'X' chaque heure occupée par au moins une rendez-vous.

Chacune de ces fonctionnalités doit être implantée sous la forme d'une méthode publique acceptant les paramètres appropriés. Vous pouvez en outre ajouter des méthodes privées permettant de réaliser des traitements susceptibles de simplifier les tâches de certaines méthodes publiques, comme par exemple une méthode qui trie les rendez-vous de l'agenda par ordre chronologique.

Compétence : **héritage et composition**.