Objectifs :

- Détecter un contexte de polymorphisme
- Maitriser le comportement polymorphique d'une méthode. Savoir faire usage des méthodes virtuelles.
- Manipuler une collection d'objets polymorphiques
- Maitriser le concept de classe abstraite et des méthodes virtuelles pures relatives. Reconnaître le contexte de leur mise en œuvre
- Redéfinir l'opérateur d'affectation

Exercice 1 Gestion de comptes bancaires

En se basant sur l'exercice relatif aux comptes bancaires du TD Héritage, on désire maintenant mettre en place une application C++ capable de gérer tous les comptes bancaires au sein d'une banque.

Rappelons que :

- Un compte bancaire est caractérisé par : un numéro de compte (RIB), un solde.

- Un compte épargne est un compte caractérisé par une valeur minimale du solde de égale par défaut à 5DT, un taux d'intérêt annuel et une fonction permettant de calculer l'intérêt annuel.

- Un compte courant est aussi un compte caractérisé en plus par une valeur minimale du solde égale par défaut à -500 DT.

- On peut déposer de l'argent dans un compte, retirer de l'argent à partir d'un compte et afficher un compte.

Par ailleurs, une banque est caractérisée par un nom et un lieu, et permet de gérer les différents types de comptes en effectuant les opérations suivantes :

- ajouter un compte, un compte courant ou un compte épargne.
- afficher tous les comptes d'une banque
- rechercher un compte donné à partir de son RIB.

Travail demandé :

1. Implémenter les classes relatives aux comptes, comptes bancaires et comptes épargnes
2. Implémenter la classe Banque
3. Implémenter un programme de gestion de comptes bancaires permettant de créer différents types de comptes et de mettre en œuvre tous les traitements déjà mentionnés

Exercice 2 Gestion d'une bibliothèque

On désire implémenter un système de gestion des emprunts d'ouvrages dans une bibliothèque. Un ouvrage doit être nécessairement soit un livre, soit une vidéo.

Une analyse a permis d'identifier les classes suivantes :

- **Ouvrage** : Un ouvrage est caractérisé par un titre (String), une date de création (String) et un indicateur d'existence (booléen : vrai si l'ouvrage est disponible dans la bibliothèque et faux s'il est emprunté).

- **Livre** : Un livre est un ouvrage caractérisé par son auteur (String).

- **Vidéo** : Une vidéo est un ouvrage caractérisé par son éditeur (String) et sa durée (Réel)

- **Abonne** : Un abonné est caractérisé par son numéro d'identité (entier), son nom (String), le numéro de son abonnement (entier) et le titre de l'ouvrage emprunté de la bibliothèque (String) (un abonné peut prendre un seul ouvrage à la fois).

- **Bibliothèque** : La bibliothèque est caractérisée par :

Un ensemble d'ouvrages, où chaque ouvrage est présent en un seul exemplaire (collection d'ouvrages) et un ensemble d'abonnés (collection d'abonnés). Cette classe doit offrir les méthodes suivantes :

1. getOuvrage : qui prend comme paramètre un titre et retourne l'ouvrage correspondant.

2. ajoutOuvrage : qui permet d'ajouter un ouvrage dans le tableau, s'il n'existe pas déjà (la comparaison doit se faire sur le titre)
3. supprimerOuvrage : qui permet de supprimer un ouvrage donné par son titre.

4. getAbonne : qui prend comme paramètre un numéro d'identité et retourne l'abonné correspondant.

5. ajoutAbonne : qui permet d'ajouter un abonné, s'il n'existe pas.

6. emprunter : qui prend comme paramètres un titre d'ouvrage et un numéro d'identité et permet d'affecter l'ouvrage correspondant s'il existe à l'abonné correspondant s'il existe ; sachant que : Un ouvrage ne peut être emprunté que s'il est disponible. Un ouvrage emprunté doit devenir non disponible.

7. rendre : qui prend comme paramètre un numéro d'identité et supprime l'ouvrage qui lui a été affecté. Cet ouvrage devient alors disponible.

8. info : qui permet d'afficher tous les ouvrages présents dans la bibliothèque, ainsi que tous les abonnés ayant un emprunt en cours.

9. livreAuteur : qui affiche les livres d'un auteur passé en paramètre.

Exercice 3

Une institution d'enseignement désire réaliser une application afin de gérer les durées des épreuves pour toutes les matières enseignées, via une application C++.
Chaque épreuve peut être soit une **épreuve pratique**, soit un **quiz**.
Une **épreuve** est caractérisée par :

- Un code unique
- Une durée (en minutes)

Une **épreuve pratique** est une **épreuve** caractérisée en plus par :

- Un indicateur (0 indique un « test », et 1 indique un « examen »)
- Le nombre d'exercices

Un **quiz** est une **épreuve** qui est caractérisée en plus par :

- Un thème
- Un nombre de questions

L'application est caractérisée par le nom de l'institution et elle doit gérer l'ensemble des épreuves.
Implémenter toutes les classes avec les méthodes que vous jugez nécessaires.
Définir les getters et les setters nécessaires.
Implémenter le constructeur de copie, l'opérateur d'affectation et le destructeur dans la bonne classe.

L'application doit répondre aux besoins suivants :

1- Ajouter une épreuve (qui peut être soit pratique, soit un quiz).
2- Afficher les quiz liés à un thème donné.
3- Calculer la durée de chaque épreuve, sachant que :

- Les durées des tests sont calculées en fonction du nombre d'exercices (25 minutes par exercices)
- Les durées des examens sont fixées à 90 minutes indépendamment du nombre d'exercices
- Les durées des quiz sont calculées en fonction du nombre des questions (une minute pour chaque question)

4- Supprimer tous les Quiz liés à un certain thème.
5- Modifier le nombre d'exercice d'une épreuve pratique donnée par son code.

On vous demande d'implémenter un programme principal (main) qui permet de tester l'ensemble des fonctionnalités demandées.