Niveau 3

TP
S4. Développement logiciel
S4.6.Programmation orientée objet : Définition de classes, Instanciation d'objets

Des Dés

TABLE DES MATIÈRES

1 - PRÉSENTATION

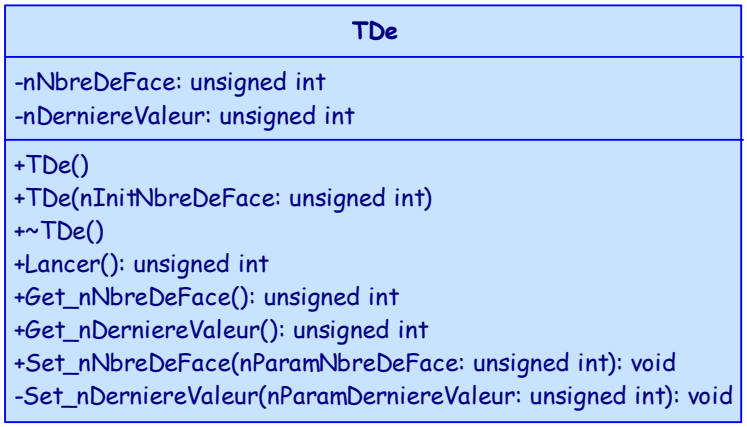
Monsieur Risoire à l'idée géniale de lancer un nouveau jeu, totalement novateur, en ligne : il s'agit de lancer trois dés et de sortir les faces 4, 2 et 1 ! Le chanceux qui tire cette combinaison hyper difficile se voit attribuer le prix fabuleux d'avoir son nom dans la liste des vainqueurs !

Aidons Dédé à programmer un dé de base avant de lui faire son jeu !

Notre travail se concentre sur un simple dé à jouer ; la programmation objet permettant de dupliquer autant d'instances d'objets de la classe de base que l'on souhaite.

Le but de ce TP est l'apprentissage et la compréhension de la programmation objet d'une classe.

L'analyse de l'objet réel a permis de rédiger le diagramme de classe SysML suivant :



Il ne reste plus qu'à coder cette classe !

2 - TRAVAIL À RÉALISER

2.1 - PRÉPARATION DES FICHIERS

On ne s'attachera pas pour ce TP à la documentation automatique dOxygen.

1) Après avoir créé un nouveau projet de type C++ console, ajoutez une nouvelle classe C++ nommée `TDe`. Vous constaterez que deux fichiers ont été créés : un fichier d'**en-tête** `.H` et un dit d'**application** `.CPP`. Normalement, ces fichiers sont pré-remplis par la **déclaration** de la classe et du constructeur par défaut (dans le `.H`) et par la **définition** du constructeur (dans le `.CPP`).

2) Déplacez les fichiers générés dans leur dossier respectif et adaptez le projet en conséquence.

2.2 - LES ATTRIBUTS

3) Ajoutez les déclarations des attributs de la classe `TDe` dans le fichier en-tête.

2.3 - CONSTRUCTEURS ET DESTRUCTEUR

4) Dans le fichier d'en-tête, ajoutez les déclarations des **constructeurs** (par défaut et paramétré) et du **destructeur**, conformément au diagramme UML.

5) Complétez le fichier application associé pour ajouter les définitions des constructeurs et du destructeur.

> Ajoutez dans chacune de ces méthodes un affichage du style "Je suis le constructeur par défaut (ou paramétré)", qui vous indiquera qui s'exécute et à quel moment.

2.4 - LE PROGRAMME PRINCIPAL

6) Modifiez le fichier du programme principal de façon à créer un objet `oMonDe6Faces` **instance** de la classe `TDe`, utilisant le **constructeur par défaut**.

7) Ajoutez un second dé dans le `main()`, ayant **20 faces** !

8) Testez ces instanciations dans le `main()`.

2.5 - LES ACCESSEURS ET MUTATEURS

9) Ajoutez dans le `.H` la déclaration des **accesseurs** du dé.

10) Ajoutez dans le `.CPP` la définition de ces méthodes.

11) Codez **l'accesseur** qui permet de retourner le nombre de face du dé.

12) Programmez le code du **mutateur** du nombre de face. Il faut toujours tester si la valeur passée en paramètre est valide, c'est-à-dire qu'elle est comprise dans la plage des valeurs possibles pour l'attribut en question. Pour le cas d'un dé, le nombre de faces ne peut pas être négatif, ni inférieur à 2. Validez la valeur du paramètre passé. S'il n'est pas valide, imposez-lui la valeur 6. Initialisez l'attribut du nombre de faces avec ce paramètre.

13) Codez **l'accesseur** qui permet de retourner la dernière valeur sortie.

14) Programmez le code du **mutateur** de la dernière valeur tirée.

15) Testez ces méthodes dans le `main()`.

2.6 - MÉTHODES MÉTIER

Ces méthodes ont été définies lors de l'analyse et définissent les actions dont les objets de la classe sont capables.

Dans le cas du dé, la seule **méthode métier** est celle qui permet de le `lancer()`.

16) Ajoutez la déclaration de la méthode dans le fichier d'en-tête.

17) Ajoutez la définition de la méthode `Lancer()`, dans le fichier `.CPP`.

18) Codez, dans la méthode `Lancer()`, le nécessaire pour obtenir un numéro au hasard en fonction du nombre de face du dé.

19) Testez cette méthode dans le programme principal sur les deux dés que vous avez créés. Affichez les résultats obtenus.

3 - MÉTHODES DE DÉVELOPPEMENT

Ces méthodes ne font pas partie des méthodes métier (ce n'est pas le rôle d'un dé de valider ses probabilités !) mais elles sont utiles pour le développeur, afin de simplifier les algorithmes ou de l'aider dans la mise au point. Généralement elles n'apparaissent pas initialement dans le diagramme de classe.

Dans le cas du dé, il faut être sûr qu'il ne soit pas pipé (chaque face a autant de chance d'être tirée que les autres : `nNbreDeTirage/nNbreDeFace`). Pour ça on doit réaliser une petite analyse statistique sur un assez grand nombre de tirages, par exemple 100 000 !

20) Incorporez la validation dans la classe `TDe`. Une nouvelle méthode d'implémentation dont le prototype sera `bool ValiderProba(unsigned int nNbreTirage)` devra alors être créée. Elle prend en paramètre le nombre de tirages pour le test et retourne `true` ou `false` en fonction du résultat.

21) Proposez le code permettant de valider que chaque face a bien le même nombre de chance d'être tirée. Pour ça, dans la méthode `ValiderProba()`, on créera **dynamiquement** un tableau `nSortie` ayant le même nombre de cases que de faces du dé. Chaque case stockera le nombre de fois que la face correspondante est sortie (`nSortie[0]` est associée à la face `1`, `nSortie[1]` à la face `2`...).

22) Ce tableau est rempli dans une boucle permettant les `nNbreTirage` tirages. Le dé est lancé, on ajoute `1` dans la case du tableau correspondante à la valeur tirée.

23) Créez une nouvelle variable (`fProba`) initialisée avec la probabilité de tirage de chaque face. Puis une seconde (`fErreur`) avec le taux d'erreur acceptable (ici 5 % de la `fProba`).

24) Validez, dans une seconde boucle que la valeur de chaque case du tableau `nSortie`, a bien une valeur comprise entre `fProba-fErreur` et `fProba+fErreur`.
Si c'est le cas, votre dé n'est pas pipé ! Bravo !
Sinon, proposez un correctif efficace.

25) Dans le `main()`, appelez cette méthode pour les deux dés déjà fonctionnels et affichez si chacun est ou n'est pas pipé !
Créez un nouveau dé ayant 100 faces et validez ses probabilités. Si elles ne sont pas valides, trouvez-en la raison et adaptez votre programme.

4 - BONUS

26) Programmez le jeu de Dédé, utilisant trois dés à 6 faces et déclarant gagnantes toutes les combinaisons `4`, `2` et `1` simultanées.