Classes d'objets graphiques en C++

On dispose de 4 classes d'objets permettant d'utiliser les capacités graphiques des machines :

1. La classe `Fenetre` qui permet de faire apparaître des fenêtres à l'écran puis d'y dessiner et d'y écrire du texte. Les points (*pixels*) dans une fenêtre sont repérés par leur coordonnées qui sont 2 entiers dont le premier augmente lorsqu'on se déplace vers la droite et le second lorsqu'on se déplace vers le bas. Le coin supérieur gauche de la fenêtre a donc les coordonnées 0 et 0. De plus l'unité de mesure dans une fenêtre est le *pixel*.
2. La classe `Souris` qui permet d'utiliser la souris dans une fenêtre. Chaque objet souris doit être associé à une fenêtre, il permettra alors de connaître les actions faites sur la souris lorsqu'elle se trouve dans cette fenêtre.
3. La classe `Couleur` qui permet de manipuler des couleurs.
4. La classe `Image` qui permet de manipuler des images crées à partir de fichiers.

1 La classe `Fenetre`

Méthodes relatives à l'apparition et les disparition d'une fenêtre

```
void apparait(const char* ="Sans nom", int=400, int=400, int=0, int=0, int=255, int=255, int=255)
```

Cette méthode fait apparaître la fenêtre à l'écran.

— Le 1er paramètre est le titre de la fenêtre (une chaîne de caractères). Par défaut la fenêtre est intitulée "Sans nom".

— Les 2 paramètres suivants définissent respectivement la largeur et la hauteur de la fenêtre. Par défaut la taille de la fenêtre est 400 pixels par 400 pixels.

— Les 2 paramètres suivants définissent les coordonnées du coin supérieur gauche de la fenêtre à l'intérieur de l'écran. Par défaut la fenêtre sera placée en position $(0, 0)$ c'est à dire en haut à gauche de l'écran.

— Les 3 derniers paramètres définissent la couleur de fond de la fenêtre par ses composantes rouge, verte et bleue à l'aide d'entiers compris entre 0 et 255. La couleur par défaut est le blanc.

```
void disparait()
```

Cette méthode fait disparaître la fenêtre de l'écran.

Méthodes relatives aux dimensions et position d'une fenêtre

```
void deplacerEn(int, int)
```

Cette méthode déplace le coin supérieur gauche de la fenêtre aux coordonnées passés en paramètre.

```
int hauteur() const
```

Cette méthode retourne la hauteur de la fenêtre.

```
int largeur() const
```

Cette méthode retourne la largeur de la fenêtre.

```
int coordEnX() const
```

Cette méthode retourne la coordonnées en x du coin supérieur gauche de la fenêtre.

```
int coordEnY() const
```

Cette méthode retourne la coordonnées en y du coin supérieur gauche de la fenêtre.

Méthodes relatives à la gestion des couleurs

```
Couleur couleurFond() const
```

Cette méthode retourne la couleur du fond de la fenêtre. La valeur retournée est un objet de la classe `Couleur`.

```
void choixCouleurTrace(int, int, int)
```

Cette méthode permet de définir la couleur qui sera utiliser lors des prochain tracés (dessins et textes) à l'intérieur de la fenêtre. La couleur est décrite par ses composantes rouge, verte et bleue.

```
void choixCouleurTrace(Couleur)
```

Idem méthode précédente mais cette fois-ci la couleur est décrite par un objet de la classe `Couleur`.

Méthodes relatives aux dessins

```
void traceLigne(int, int, int, int, int=1)
```

Cette méthode permet de dessiner une ligne dans la fenêtre.
- Les 2 1ers paramètres sont les coordonnées d'une extrémité de la ligne.
- Les 2 paramètres suivants sont les coordonnées de l'autre extrémité.
- Le dernier paramètre définit l'épaisseur de la ligne. Par défaut une ligne fait 1 pixel d'épaisseur.

```
void traceArc(int, int, int, int, int=0, int = 360, int=1)
```

Cette méthode permet de tracer un arc dans la fenêtre.
- Les 2 1ers paramètres sont les coordonnées du coin supérieur gauche.
- Les 2 suivants sont respectivement la largeur et la hauteur de l'arc.
- Le paramètre suivant indique l'angle de début de tracé exprimé en degré.
- Le paramètre suivant indique l'angle d'arc tracé exprimé en degré
- Le dernier paramètre définit l'épaisseur de la ligne. Par défaut une ligne fait 1 pixel d'épaisseur.

```
void remplitRectangle(int, int, int, int)
```

Cette méthode trace un rectangle plein.
- Les 2 1ers paramètres sont les coordonnées du coin supérieur gauche du rectangle.
- Les 2 suivants sont respectivement la largeur et la hauteur du rectangle.

```
void remplitEllipse(int, int, int, int)
```

Cette méthode trace une ellipse pleine contenue dans un rectangle
- Les 2 1ers paramètres sont les coordonnées du coin supérieur gauche du rectangle contenant l'ellipse.
- Les 2 suivants sont respectivement la largeur et la hauteur du rectangle contenant l'ellipse.

Méthodes relatives à l'écriture de textes

```
void choixFonte(const char*, int=12, bool=false, bool=false)
```

Cette méthode permet de choisir la police de caractères.
- Le 1er paramètre est le nom de la police comme par exemple "times", "helvetica", "courier", etc.
- Le 2eme paramètre est la taille de la police (12 par défaut).
- Le 3eme paramètre (TRUE ou FALSE) indique si l'écriture est en gras ou non.
- Le dernier paramètre indique si l'écriture est en italiques ou non.

```
void ecrit(int, int, const char*)
```

Cette méthode écrit une chaîne de caractères.
- Le 2 1ers paramètres sont les coordonnées auxquelles sera écrite la chaîne de caractères.
- Le paramètre suivant est la chaîne de caractères qui doit être écrite.

```
void ecrit(int, int, int)
```

Idem méthode précédente mais cette fois-ci le 3eme paramètre, celui qui sera écrit, est un entier.

Méthodes relatives à l'affichage d'images

```
void afficheImage(const char*, int=0, int=0)
```

Cette méthode affiche permet d'afficher une image.
- Le 1er paramètre est une chaîne de caractères contenant le nom du fichier image.
- Les 2 autres paramètres contiennent les coordonnées où sera affichée l'image (le coin supérieur gauche)

```
void afficheImage(Image, int=0, int=0)
```

Idem méthode précédente mais cette fois-ci l'image à afficher est contenu dans un objet de la classe `Image`

Autres méthodes

```
void effacer()
```

Cette méthode efface tout le contenu de la fenêtre.

```
void delai(int) const
```

Cette méthode permet d'attendre un délai exprimé en ms par le paramètre.

Les constructeurs

```
Fenetre()
```

Ce constructeur par défaut crée la fenêtre mais ne l'affiche pas.

```
Fenetre(const char*, int, int, int=0, int=0, int=255, int=255, int=255)
```

Ce constructeur crée une fenêtre et l'affiche, ses paramètres sont identiques à ceux de la méthode `apparait`.

2 La classe `Souris`

```
void associerA(Fenetre&)
```

Cette méthode permet d'associer la souris à une fenêtre

```
void position(int&, int&) const
```

Cette méthode permet de connaître à tout moment la position de la souris. Les deux paramètres reçoivent les coordonnées du point de la fenêtre où se trouve la souris.

```
bool testeBoutons(int&, int&, int&) const
```

Cette méthode permet les actions faites sur les boutons de la souris. Elle renvoie :
- `false` si aucun bouton n'a été appuyé.
- `true` sinon. Dans ce cas les 2 premiers paramètres reçoivent les coordonnées du point où à eu lieu l'action, le dernier paramètre reçoit le numéro du bouton qui a été appuyé (1 pour le bouton gauche, 2 pour celui du centre, 3 pour celui de droite).

Les constructeurs

```
Souris(Fenetre&)
```

Ce constructeur permet d'associer directement la souris à une fenêtre.

3 La classe `Couleur`

Méthodes de manipulation

```
void definir(int, int, int)
```

Cette méthode permet de définir les composantes *rouge, verte* et *bleue* à l'aide d'entiers compris entre 0 et 255.

```
int rouge() const;     int vert() const;     int bleu() const;
```

Ces méthodes retournent l'une des composantes de la couleur.

Constructeur

```
Couleur(int=0, int=0, int=0)
```

Ce constructeur permet de construire une couleur en précisant directement ses composantes rouge, verte et bleue.

4 La classe `Image`

```
Image(const char*)
```

Une image sera nécessairement construite à l'aide d'un paramètre : le nom du fichier contenant l'image.

```
int hauteur() const
```

Cette méthode retourne la hauteur de l'image.

```
int largeur() const
```

Cette méthode retourne la largeur de l'image.