COLLÈGE MONTMORENCY

Travail pratique #1

Pondération : 17%
Travail en équipe de 2

Toutes les spécifications du projet doivent être respectées

Dates des remises

Livrable	***Contenu***	***Date de remise***	**Pénalité de retard**
Livrable#1	L'interface graphique complète, sans la gestion des évènements.	Vendredi 22 février 2019 avant 23 h 55 mn.	3% par jour de retard.
Livrable#2	Fonctionnalité de dessin de toutes les formes implantée à 100% (sans la sauvegarde des formes).	Vendredi 1er mars 2019 avant 23 h 55 min.	3% par jour de retard.
Livrable#3	L'application complète testée et documentée.	Vendredi 15 mars 2019 avant 23 h 55 min.	4% par jour de retard.
Le « workSpace » du projet doit être nommé : TP1_VosNoms_livrable_N. Celui-ci doit être compressé en format compressé (.ZIP ou .RAR), puis déposé dans Moodle dans le dossier attitré selon le livrable. – N représente le numéro du livrable (1,2 ou 3)			

Programmation d'une application de dessin vectoriel

But :

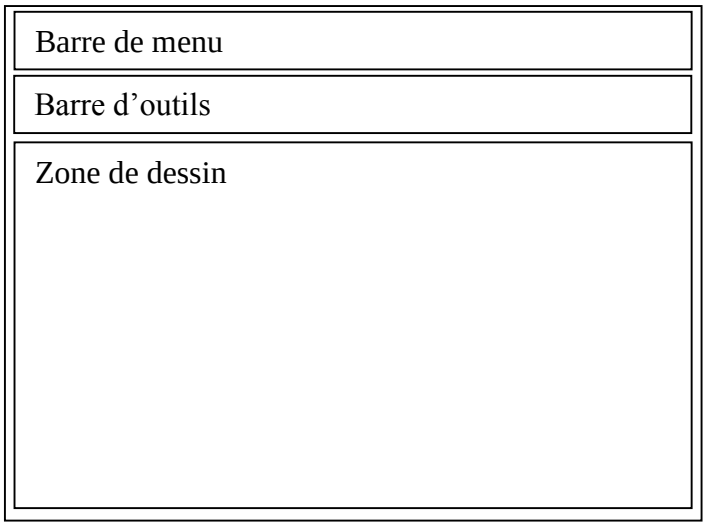
- Consolider les acquis de la programmation orientée objet.
- Traiter les évènements.
- Utiliser le graphisme en Java.
- Produire une interface graphique munie d'une barre d'outils et d'un menu déroulant.
- Utiliser les boites de dialogue pour communiquer avec l'utilisateur et le système.

Il s'agit de réaliser une application du style de **Paint** de Windows, mais en plus simplifiée.
Notre application devrait nous permettre de réaliser les fonctionnalités suivantes :

1. Créer une nouvelle feuille de dessin;
2. Choisir une couleur de contour et de fond d'une forme dans la palette de couleur;
3. Dessiner des formes (ellipse, rectangle, trait droit) sur une feuille de dessin en respectant les couleurs de fond et de contour sélectionnées;
4. Sauvegarder la feuille de dessin sur le disque (en utilisant la sérialisation);
5. Ouvrir une feuille de dessin préalablement enregistrée;
6. Quitter l'application.

Note : Tout changement dans les spécifications énoncées ici-bas doit être justifié par l'étudiant et validé par l'enseignant.

1. Description de l'interface graphique et des spécifications fonctionnelles



La fenêtre de l'application doit s'afficher au centre de la fenêtre et ses dimensions ne doivent pas dépasser celles de l'écran. Vous devez choisir un logo pour votre application (ce sera l'icône de la fenêtre) **et le titre de la fenêtre doit être le nom du fichier ouvert suivi de celui de l'application.** (Prendre comme exemple l'application bloc-notes de Windows)

1.1 La zone de dessin

La zone de dessin est la feuille blanche du dessinateur. Cette feuille est un panneau qui s'grandit en même temps que la fenêtre. Une seule feuille de dessin sera affichée à la fois dans la fenêtre principale. Elle affiche toutes les formes dessinées et permet d'en rajouter d'autres formes. On doit pouvoir voir la progression d'une forme tout au long de son tracé avec la souris.

1.2 La barre d'outils

- La barre d'outils doit comporter des icônes permettant de sélectionner les formes à dessiner (ovale, rectangle, et trait), les couleurs de contour et les couleurs de fond des formes. Il y a 3 groupes d'icônes dans cette boite à outils :
  - Un groupe représentant les icônes des couleurs des traits et des contours des formes
  - Un groupe représentant les icônes des couleurs de remplissage des formes.
  - Un groupe représentant les icônes des formes (ovale, carré, trait droit)
- Vous devez choisir **au moins** 6 couleurs de votre choix (Exemple : blanc, rouge, vert, jaune, bleu, noire). On doit trouver les mêmes couleurs dans les deux premiers groupes.
- Quand une icône est sélectionnée, elle doit être **mise en évidence** (par exemple, elle peut rester enfoncée ou le fond de l'icône pour changer de couleur)**.**
- Une « **infobulle »** est affichée au survol de l'icône par la souris pour dire ce que fait l'outil.
- Cette barre d'outils doit **être insérée initialement en haut de votre fenêtre**.
- On doit pouvoir distinguer facilement les icônes représentant la couleur de remplissage de celles représentant la couleur de contour. Vous devez créer vous-mêmes vos images dans Paint ou dans tout autre logiciel de conception d'images.

1.3 La barre de menu

- La barre de menu doit comporter un menu **Fichier** et un menu **À propos**
- Le menu **Fichier** doit avoir 5 options :
  - **Nouveau** qui permet de créer une nouvelle feuille de dessin. Si la feuille de dessin en cours n'a pas encore été sauvegardée, un message doit s'afficher demandant à l'utilisateur s'il veut d'abord enregistrer la feuille en cours avant d'ouvrir une nouvelle.
  - **Enregistrer** doit pouvoir ouvrir une boite de dialogue de type JFileChooser pour enregistrer le fichier, si jamais la feuille de dessin en cours n'a pas encore de nom. Sinon l'ancienne version du fichier sera écrasée par la nouvelle.
  - **Enregistrer sous** : pour enregistrer la zone de dessin sous un autre nom.
  - **Ouvrir :** permet d'ouvrir une feuille de dessin préalablement enregistrée à l'aide d'une boite de dialogue de type JFileChooser. S'il existe une feuille de dessin non encore enregistrée, alors un message doit être affiché pour demander à l'utilisateur de confirmer la sauvegarde de l'ancienne feuille de dessin.
  - **Quitter** : pour quitter l'application après confirmation de l'utilisateur.
- Le menu **«À propos »**
  - **A**ffiche une boite d'informations spécifiant : le nom de logiciel, les auteurs, la version du logiciel
- Vous devez ajouter les raccourcis clavier nécessaires permettant d'utiliser les menus.

2. Spécifications détaillées

2.1 Nous utiliserons les méthodes suivantes pour dessiner nos formes vides ou pleines

```
drawLine(int x1, int y1, int x2, int y2)
drawOval(int x, int y, int width, int height)
drawRect(int x, int y, int width, int height)
fillRect(int x, int y, int width, int height)
fillOval(int x, int y, int width, int height)
```

2.2 Les classes métier permettant de réaliser le travail.

Classes **Forme**, **Trait**, **Rectangle** et **Ovale**. Ces classes doivent être documentées selon la javadoc.

Description des classes

a. La classe Forme :

C'est une classe abstraite qui représente n'importe quelle forme (un trait, un rectangle ou un ovale). Elle regroupe tous les éléments communs aux figures :

- Les coordonnées x1 et y1 du premier point qui débute la forme. Ces coordonnées sont récupérées quand la souris est appuyée pour débuter le dessin de n'importe quelle forme.
  - **Note** : x1 et y1 doivent représenter le **point supérieur gauche** pour pouvoir tracer les formes correctement.
- La couleur de contour et la couleur de remplissage de la forme. Ces deux attributs sont de type **Color**. L'utilisateur peut choisir une couleur de contour et une autre couleur pour le remplissage de sa forme.
- Les méthodes, en plus des accesseurs, nécessaires sont :
  - `Public abstract setParametres (int x1, int y1, int x2, int y2);`
    - Les coordonnées x1 et y1 sont ceux décrits ci-haut. x2 et y2 représentent les coordonnées du dernier point de la figure. Ces dernières sont obtenues quand la souris est relâchée. Pour obtenir la largeur et la hauteur du rectangle et de l'ovale, il suffit de trouver la relation mathématique entre ces coordonnées.
  - `Public abstract tracer (Graphics g);`

- Le contexte Graphique g passé en paramètre sera celui de la feuille de dessin. C'est cette méthode qui s'occupera de dessiner la forme en appelant les méthodes spécifiées au point 2.1

b. La classe Trait :
Cette classe hérite de **Forme**. Elle possède en plus, les coordonnées **x2** et **y2** qui représentent l'extrémité du segment dessiné. Elle redéfinit les méthodes **setParametres**(int x1, int y1, int x2, int y2) et tracer(Graphics g);

c. La classe Rectangle :
Cette classe hérite de **Forme**. Elle possède en plus la **largeur** et la **hauteur** du rectangle (de type **int** car on travaille en pixels). Ajoutez les accesseurs reliés s'il y a lieu.

Cette classe redéfinit la méthode **setParametres**(int x1, int y1,int x2, int y2) qui doit calculer le champ largeur en fonction des abscisses x1 et x2 et le champ hauteur en fonction des ordonnées y1 et y2. Pensez à garder comme coordonnées x1 et y1 celles du **point supérieur gauche** pour pouvoir tracer le rectangle correctement.

Cette classe redéfinit la méthode **tracer**(Graphics g). Elle trace un rectangle en utilisant les méthodes spécifiées au point 2.1.

d. La classe Ovale : <u>à vous de la définir. L'implémentation de cette classe doit être optimale; utilisez l'héritage.</u>

e. La classe PanDessin :
- C'est une classe <u>dérivant de JPanel</u>. Elle représente le panneau pour dessiner.
- Les données de cette classe sont :
  - **Color Fg** et **Color Bg** : la couleur de fond et la couleur du tracé courantes choisies dans la boite à outils. Ces données sont modifiées lors du clic sur les icônes de couleurs de la boite à outils.
  - **MouseEvent pemierClic** : Cet objet nous permet de récupérer les coordonnées x1 et y1. (`premierClic.getX() et premierClic.getY()`)
  - Un attribut **int typeFigure** : détermine le type de forme choisie dans la boite à outils.
  - **Forme formeCourante** : la forme qui est en cours de dessin. Elle sera initialisée en fonction du type de la forme sélectionnée. Exemple dans le cas d'un rectangle, on écrira : *`formeCourante = new Rectangle(…).`*
  - **ArrayList <Forme> liste** : la liste de toutes les formes de la feuille de dessin en cours. Elle contiendra tous les objets dessinés. C'est cette liste qui sera <u>stockée dans un fichier sur le disque</u> (en utilisant la sérialisation).
- Cette classe doit redéfinir la méthode **paintComponent(Graphics g).**

```
paintComponent(Graphics g){
       // Appel de la méthode parent
      …
      //Parcourir la liste des formes  et pour chaque élément de cette
      liste, appeler la méthode tracer(g)
…}
```

- Cette classe doit implémenter les interfaces **MouseListener** et **MouseMotionListener**.
  - Les coordonnées x1 et y1 sont récupérées par la méthode
- Cette classe doit aussi implémenter les événements suivants :
  - **MousePressed** (MouseEvent e) qui correspond au début du clic.
    - Les coordonnées x2 et y2 **intermédiaires** sont récupérées par la méthode

- **mouseDragged**(MouseEvent e) qui correspond à la phase pendant le clic. Ces coordonnées sont toujours mises à jour (écrasées). Cela permet de tracer la forme progressivement pendant que la souris est glissée (draguée).

- N'oubliez pas d'appeler la méthode **repaint**() du panneau pour actualiser l'affichage.

3. Structuration du projet

Le projet doit être organisé en sous-dossiers (packages) :

- un pour les images
- un pour les classes métiers (Forme, Carre,…)
- un pour les classes graphiques

Vous pouvez créer une interface pour déclarer l'ensemble des constantes liées à votre application. Voir l'annexe en fin du document.

4. Demande d'amélioration de l'application

Une fois l'application terminée et approuvée par la professeure, apportez les modifications décrites ici-bas.

- Ajoutez un triangle isocèle.
- Ajoutez la possibilité de choisir la grosseur du pinceau.
- Ajoutez la possibilité de choisir la couleur dans une boite de dialogue de choix de couleurs.

Annexe

```
public interface AffichageConstantes {

        // Font-----------------------------------
        final Font F_MENUBAR = new Font("Segoe UI", Font.BOLD, 20);
        …
        // Messages-------------------------------------
        final String MSG_ERROR_OPEN_FILE = "Erreur lors de l'ouverture d'un fichier.";
        …
        // ToolTips -------------------------------------
        final String[] BTN_PAL_COLORS = { "Couleur noire", "Couleur grise",…}
        …
}
```

On peut alors utiliser ces constantes dans une classe si elle implémente cette interface

```
public class Fenetre extends JFrame implements AffichageConstantes {

        // Font-----------------------------------
        menuColorBar.setFont(F_MENUBAR);
       …
       // Messages-------------------------------------
       JOptionPane.showMessageDialog(this, MSG_ERROR_OPEN_FILE);
       …
        // ToolTips -------------------------------------
       btNoir.setToolTipText(BTN_PAL_COLORS[0]);
       …
```