IHM en JAVA

Editeur Graphique 2D

`https://bit.ly/2SlMXZa`

- ▷ Solution de l'exercice Calulatrice : `https://bit.ly/2G0nCTm`
- ▷ Utiliser l'EDI (Environnement de Développement Intégré) ECLIPSE.
- ▷ JAVA API documentation : `http://docs.oracle.com/javase/7/docs/api/`
- ▷ A la fin de la séance :
  - A partir de votre compte UNISTRA, envoyez vos fichiers JAVA par email à l'adresse `habed@unistra.fr`.
  - Votre email doit avoir pour sujet "TI2A Editeur Graphique 2D".
  - Renseignez vos nom et prénom dans le mail et en commentaire dans tous les fichiers.

Vous allez développer une interface graphique pour des formes géométriques.

Les classes : Héritage, Polymorhisme, Exceptions, GUI

- ▷ Ecrire les classes `Point2D`, `Forme2D`, `Carre` et `Cercle` - décrivant les formes dont elles portent les noms - qui respectent les spécifications suivantes :
  - Un `Point2D` consiste en une coordonnée `x` et une coordonnée `y` : `(x,y)`. Ces coordonnées doivent être privées et l'accès en lecture et écriture doit se faire via les getters et les setters. Si non renseignées, les coordonnées d'un `Point2D` sont mises à `(0,0)`. Prévoir `toString` pour décrire l'objet par une chaîne `(x,y)`.
  - Une `Forme2D` désigne de façon générale une forme géométrique bidimensionnelle définie par son origine (un `Point2D`) et une couleur (voir la classe `Color` de `java.awt`). La classe `Forme2D` doit fournir les services suivants :
    - la construction (si non renseignée, la couleur par défaut est le bleu) ;
    - un mécanisme pour le comptage du nombre d'objets `Forme2D` créés ;
    - calcul de l'aire de la forme ;
    - translation d'une forme par déplacement de l'origine ;
    - redimensionnement d'une forme ;
    - accès en lecture (getter) à l'origine (tout en protégeant l'intégrité de la forme géométrique) ;
    - getters et setters pour la couleur ;
    - description de l'objet à travers `toString` : `(x,y):couleur` ;
    - dessin de la forme ;
    - comparaison des formes 2D entre elles (interface `Comparable`) sur la base de leurs aires ;
    - vérifier si un point est à l'intérieur, à l'extérieur ou aux frontières de la forme ;
    - lancer une exception "non obligatoire" `WrongOriginException` (à définir) si au moins l'une des coordonnées de son origine est négative ;

  - lancer une exception "obligatoire" `WrongSizeException` (à définir) si largeur ou hauteur est négative ;

  Pourquoi `Forme2D` est abstraite ?
  Vérifier que vous avez bien placé vos exceptions.

- Un `Carre` est une `Forme2D` avec un côté. On souhaite cette classe concrète. Ne pas oublier de redéfinir `toString` :
  `Carre:(x,y):cote:couleur:aire` et mettre en place un mécanisme pour compter les carrés.
  - *Remarque* ; en redimensionnant un carré, son origine (coin supérieur gauche), doit demeurer inchangée. Le dessin se limite (pour l'instant) à afficher (console) la chaîne retournée par `toString`.
- Un `Cercle` est une `Forme2D` avec un rayon. On souhaite cette classe concrète. Ne pas oublier de redéfinir `toString` :
  `Cercle:(x,y):rayon:couleur:aire` et mettre en place un mécanisme pour compter les cercles.
  - *Remarque* : l'origine du cercle correspond à son centre. En redimensionnant un cercle, son centre demeure inchangé. Le dessin se limite (pour l'instant) à afficher (console) la chaîne retournée par `toString`.

L'application

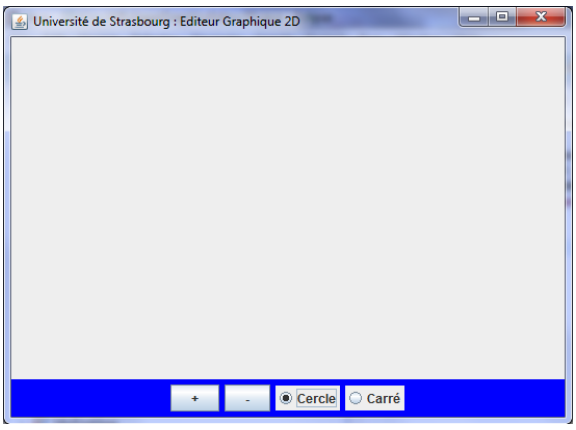
1. Créer la fenêtre :
   - Créer une classe qui hérite de JFrame, y ajouter la méthode "main" et y créer une instance de votre fenêtre.
   - Donner un titre, une taille, et l'affcher à l'écran.
   - Créer un panel, ajouter lui un `BorderLayout`, le définir comme panel par défaut :
     `setContentPane(monPanel)`.
   - Ajouter un Label en haut (`BorderLayout.NORTH`) avec le texte "TP4".
2. Créer une classe `ZoneDessin` qui hérite de Component
   - Déclarer comme attribut une liste de Form2D du TP précédent.
   - Surcharger la méthode `paint`, et selon le type de la forme, dessiner la forme (`fillRect/fillOval` dans `Graphics`).
   - Positionner la forme sur le dessin selon son point d'orgine et de la bonne taille.
   - Ajouter cette zone de dessin au centre (`BorderLayout.CENTER`)
3. Ajouter un bouton "+" (qui ajoute une forme dans la liste). Pour ce faire
   - Utiliser le code suivant qui créé un panel avec le bouton :

```
private JPanel creeBoutons() {
    // créé un panel avec des éléments alignés à gauche
    final JPanel panel =
        new JPanel(new FlowLayout(FlowLayout.CENTER,200,0));
```

```
        // création du bouton ajouter
        final JButton ajouter = new JButton("+");
        // création d'une action pour ajouter dans la liste
        ActionAjouter actionAjout = new ActionAjouter(zoneDessin);
        // affectation de cette action au bouton
        ajouter.addActionListener(actionAjout);
        // ajout du bouton dans la fenetre
        panel.add(ajouter);
        return panel;
    }
```

- Insérer le panel ainsi créé en bas du panel principal (`BorderLayout.SOUTH`).
- Créer une classe ActionAjouter qui implémente ActionListener. Dans la méthode "actionPerformed" (appelée au moment de l'exécution de l'action) écrire le code qui ajoute à votre liste un carré de dimension aléatoire entre 50×50 et 150×150 centré dans la zone de dessin (profiter de la variable "zoneDessin" donnée dans le constructeur). Choisir une couleur RGB aléatoire pour chaque nouvelle forme. Forcer le rafraîchissement de la fenêtre avec "repaint()" depuis votre objet "Component". Vérifier que la forme s'affiche.
- De la même façon (dans "creeBoutons" et les actions) faire le bouton "-" pour supprimer la dernière forme.
- Ajouter un bouton "Tri" pour lancer le tri de la liste de formes dans l'ordre décroissant de leurs tailles. Forcer le rafraîchissement de la fenêtre.

4. Nous souhaitons à présent permettre à l'utilisateur de sélectionner le type de forme géométrique à créer. Ceci sera réalisé à l'aide de boutons radio : voir classes `JButtonRadio` et `ButtonGroup`. L'utilisateur sélectionne "une" forme géométrique avant d'appuyer sur "+".



5. Ajouter un `KeyListener` sur votre dessin
   - Dans le constructeur de votre composant ZoneDessin ajouter la ligne :

     ```
     this.setFocusable(true);
     this.addKeyListener(....);
     ```

   - Ecrire le code dans keyPressed(KeyEvent) de votre objet de type KeyListener de façon à ce que la touche 'r' du clavier ajoute un Rectangle, 'c' un cercle, 'd' supprime la dernière forme.

6. Gestion des évènements de la souris :
   - Un `mouseClicked` avec le bouton gauche permet de sélectionner la première forme géométrique rencontrée dans la liste triée (ordre décroissant) contenant les coordonnées de la souris. La forme sélectionnée devient "rouge". Un `mouseClicked` avec le bouton droit désélectionne la forme : elle reprend sa couleur d'origine.
   - Lorsqu'une forme est sélectionnée, un `mousePressed` à l'intérieur de la forme suivi par un `mouseDragged` permet de déplacer la forme en translation.
   - Lorsqu'une forme est sélectionnée, un `mousePressed` sur le bord de la forme suivi par un `mouseDragged` permet de redimensionner la forme.