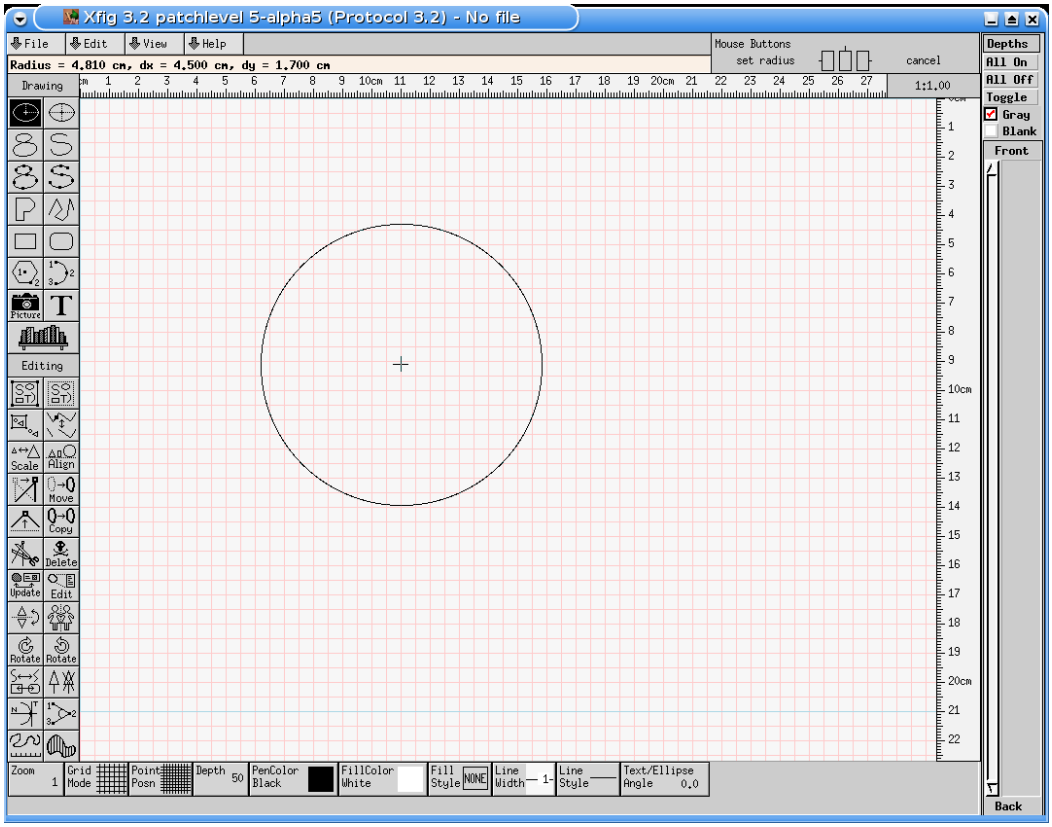
TD Ingénierie de l'Interaction Homme Machine

RICM2, année 2007/2008

1. But des TD

Les TD, d'une durée de 12h par groupe (4 séances de 3h) portent sur la réalisation d'un éditeur de dessin vectoriel de type Xfig (Figure 1) simplifié. Ils ne portent pas sur l'ergonomie des logiciels. L'IHM sera implémentée en Java à l'aide de la librairie Swing.

Figure 1 : Exemple d'interface (Xfig)



2. Spécifications

a. Spécifications fonctionnelles

Vous devez réaliser une IHM de **dessin vectoriel**. Il s'agit de pouvoir placer dans un cadre un ensemble de formes 2D basiques (lignes, cercles, rectangles, ...).
Chaque forme possède différents **attributs** (position, taille, orientation, épaisseur de trait, couleur, ...) que l'on peut assigner à la création ou modifier une fois la forme créée.
Chaque objet pourra être **sélectionné** de façon interactive à la souris et ses paramètres modifiés (déplacement de la forme par manipulation directe, modification de la couleur, ...).

b. Spécifications externes

En suivant l'exemple fourni par Xfig, vous devez réaliser une IHM contenant :

- Une **fenêtre de dessin** (cadre blanc) où l'on placera les formes souhaitées. Sa taille pourra être redimensionnée.
- Un ensemble d'**icônes de choix de tracés** du coté gauche de la scène (lignes, cercles, rectangles, ...) ainsi que l'**icône de sélection des formes**.
- Eventuellement un ensemble d'**icônes de caractéristiques** (couleur, épaisseur de traits, ...) en bas de la scène.
- Eventuellement un ensemble de **menus de préférence/import-export** (chargement, sauvegarde, ...) en haut de la scène.

Suivant la sélection des différents icônes, et caractéristiques, vous pourrez alors dessiner un ensemble de formes à l'écran.

c. Spécifications techniques

Les **conteneurs Java** de la libraire **Swing** seront utilisés afin de pouvoir positionner les éléments interactifs tels que les **boutons** de formes et d'attributs et de pouvoir gérer les **événements** associés (clics et mouvements de la souris).
Les formes dessinées (lignes, cercles, ...) proviendront de la librairie **awt.Graphics** de java permettant de tracer directement sous forme vectorielle.
La mise en place de la gestion des événements et leur lien avec l'affichage des formes se fera directement en codant "à la main" (ex. coder l'ensemble avec Emacs ou n'importe quel éditeur de texte, mais ne pas utiliser de GUI).

3. Consignes de codage

a. Structuration

Le code devra refléter une séparation des préoccupations entre la présentation **graphique**, la **gestion des événements** et le noyau fonctionnel de **gestion des formes**.

Voici les classes **minimales** devant apparaître de façon obligatoire dans votre code :

L'IHM sera contenue dans une classe nommée *VectorialDrawing*.

La fenêtre de dessin et l'ensemble des icônes seront séparés en **deux classes distinctes**.

Une classe sera spécialisée dans la **gestion des événements** (fenêtre de dessin et icônes).

Les formes peuvent être de différentes natures (lignes, rectangle, cercle, ...). Cependant, l'ensemble de ces formes devront **dériver d'une classe de base** *ElementGraphic* possédant des **attributs communs** à l'ensemble des formes : (centre, largeur, hauteur, couleur, taille du trait).

Les formes plus spécialisées seront **dérivées** de cette forme de base, typiquement:

```
class Circle extends ElementGraphic
{
  private double Radius;

  private double setRadius() {this.Radius = Radius;}
  private double getRadius() {return Radius;}
}
```

Enfin, l'ensemble des formes affichées sera stocké sous forme d'une **liste chaînée** afin de simplifier leur ajout/suppression. Une classe *GraphicList* sera spécialisée dans la gestion de cette liste. Elle permettra notamment de **retrouver un élément** de la liste en fonction de **coordonnées passées en paramètres** (afin de sélectionner un objet). La liste chaînée contiendra l'ensemble des objets sous forme d'*ElementGraphic*.
La classe d'origine de ces objets pourra être retrouvée à l'aide de la commande *instanceof*.

ex.

```
ElementGraphic tempElement = List.getElement(k);
if(tempElement instanceof Line)
{
  Line L = (Line)tempElement;
  ...
}
if(tempElement instanceof Disc)
{...
```

b. Ecriture

Vos noms d'attributs, méthodes et classes devront **respecter les normes** de Java.
ex. CeciEstUneClasse
ceciEstUneMethode()
int ceciEstUnAttribut;

Chaque classe sera codée dans un **fichier séparé**.
Votre code sera largement **commenté** afin de définir le rôle de vos méthodes et classes. Les passages complexes feront également l'objet de commentaires.
Il conviendra de tenir une attention particulière à ce que vos commentaires puissent être **réutilisés par javadoc** pour une création automatique de fichier d'aide.
ex.

```
/**
 * @author votre nom
 * @version finale du 11/2007
 */
```

(@param parametre description,@return description,@exception classname description)

A l'intérieur de vos classes, veuillez bien à **encapsuler** votre implémentation au maximum (pas d'attributs publics, travail par itérateurs, ...).

Votre implémentation devra permettre d'augmenter le nombre de formes de bases facilement en

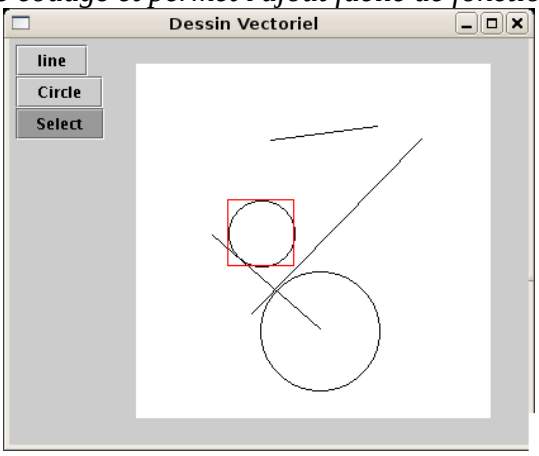
suivant une **hiérarchie claire** dans vos dérivations de classes.
(Ellipse dérive de Circle qui dérive de ElementGraphic, ...)

c. Consignes d'avancement

Votre travail sera jugé sur la **robustesse** de votre interface face à un utilisateur quelconque.
Il est important que celui-ci ne 'plante' pas à la première mauvaise manipulation. Il est donc fortement conseillé d'**avancer étape par étape** en respectant les points détaillés ci-après (Travail à faire, programme minimal). N'essayez pas de mettre en place dès le début quinze formes, une palette de couleur et l'historique de toutes vos actions.

Essayez de mettre en place initialement une fenêtre d'affichage et une barre d'outils avec les boutons de formes (2 formes sont suffisantes au début) et de sélection de celles-ci (Figure 2).
Une fois cette structure minimale mise en place de façon suffisamment générique (penser à l'aspect objet de votre code), il sera aisé d'y rajouter des couches de fonctionnalités par la suite.
Un programme avec plein d'options ne fonctionnant pas ou ayant de nombreux 'bugs' sera plus mal vu qu'un programme avec peu d'options mais fonctionnant très bien.

Figure 2 : IHM minimaliste permettant d'afficher lignes et cercles ainsi que de les sélectionner. Ne nécessite que peu de temps de codage et permet l'ajout facile de fonctionnalités par la suite.



3. Résultats attendus et barème

a. Programme minimal

Votre programme devra remplir ces fonctionnalités **au minimum**:

- Dessin au choix (sélectionnable par des icônes en partie gauche la fenêtre) de trois types de formes: Ligne, Cercle, Rectangle.

- Visualisation de la forme en cours lors du déplacement de la souris après le premier clic (ex. pour le cercle, le premier clic désigne le centre. Le rayon évolue alors en suivant le

mouvement de la souris. Au second clic, le cercle reste figé).

- Possibilité de modifier la couleur (au moins 3 couleurs au choix).
- Possibilité de sélection d'une forme une fois dessinée par clic de souris à coté de celle-ci dans l'écran d'affichage.
- Affichage de l'objet sélectionné par des rectangles pleins sur les quatre côtés extrémaux de la forme.
- Déplacement de la forme sélectionnée par "drag and drop".
- Suppression d'une forme sélectionnée (icône suppression doit apparaître).
- Sauvegarde et chargement de votre scène sous forme de fichier ascii simple
  (ex. pour une ligne puis un cercle:
  L (x0,y0) (x1,y1) (r,g,b)
  C (x0,y0) R (r,g,b)
)
- L'accès au chargement et à la sauvegarde se fera via un menu situé en haut de la fenêtre de visualisation.

b. Notation

Différents critères entreront en compte pour la notation de votre travail. Gardez à l'esprit qu'il s'agit d'une IHM. La notation tiendra donc compte de l'aspect visuel du projet ainsi que de son interaction avec l'utilisateur. Il sera donc plus important de vérifier la robustesse et la facilité de prise en main de votre IHM que de l'optimisation de votre code interne.

/4 points *Respect des consignes de codage*

- Respect de la structure des classes
- Encapsulation du code
- Commentaires corrects dans le code (utilisation de javadoc)
- Règles typographiques de java

/5 points *Respect des consignes fonctionnelles*

- Choix des formes de base (Ligne, Disque, Rectangle)
- Affichage et sélection corrects de celles-ci
- Possibilité de déplacement des formes par "drag and drop"
- Mise en place des caractéristiques de base
- Sauvegarde et chargement des figures dans un fichier ascii

/5 points *Respect des consignes IHM*

- Simplicité d'utilisation (sigles explicites, choix aisé des formes)

- Effets de bords et détails (mieux vaut peu de choses qui fonctionnent bien que beaucoup de choses ne fonctionnant qu'à moitié) : code robuste face aux situations exceptionnelles (changement d'icônes avant fin du tracé de la forme, clic en dehors de la zone de dessin, ...). Programme qui ne ‘crash’ pas même en cas de fausse manipulation.

- Aspect classique de l'IHM (menus et icônes classique, ne doit pas désorienter l'utilisateur final). Doit être compréhensible dans ses fonctionnalités standard sans lecture d'une documentation extérieure.

- Sélection et manipulation aisées des formes (doit pouvoir sélectionner facilement une forme souhaitée proche de la zone de clic)

/2 points *Avancement régulier du projet*

- Montrer un avancement régulier au cours des séances

- Réaliser le projet petit à petit en s'assurant de son fonctionnement correct.

- Partage des tâches au sein du binôme

/4 points *Amélioration du projet minimal*

- Voir section suivante

c. Améliorations possibles

Si l'ensemble des points cités devant être réalisés dans le programme minimal sont réalisés, et que l'ensemble des consignes de codage ont étés respectés. Il vous sera alors possible d’enrichir / d’améliorer votre projet.
Vous pourrez alors mettre en place par difficulté (listes non exhaustives) :

Facile:

- Ajout de caractéristiques: palette de couleur, épaisseur du trait, ...
- Ajout de fonctionnalités: Rotation, homothétie, symétrie, ...
- Ajout de formes de bases: Ellipses, Formes pleines, Texte, Triangle, ...
- Soigner l'aspect de l’IHM (icônes descriptifs, menu de sauvegarde et chargement classique, ...)

Moins aisé :

- Possibilité de zoom et de-zoom sur la figure.
- Mise en place d'annulation des dernières modifications (enregistrement des actions dans une pile).
- Ajout de formes plus complexes: Splines, courbes interpolantes avec affichage et contrôle des tangentes.

5. Consignes de rendu

A la fin des 4 séances, chaque binôme livrera son code sous forme d'archive **Nom1_Nom2.zip** ou **Nom1_Nom2.tar.gz**.

L'archive contiendra:

- L'ensemble du code source dans un répertoire **/source**
- Les classes compilées permettant une exécution par la machine virtuelle java dans un répertoire **/bin**
- Un fichier d'exemple créé à l'aide de votre programme que l'on peut charger dans un répertoire **/save**
- Un mini compte-rendu (moins de 5 pages) au format pdf contenant:
  - *Une copie d'écran valorisante de votre IHM*
  - *Ses fonctionnalités (fonctions de bases et améliorations faites le cas échéant en expliquant pourquoi vous avez choisi telle ou telle amélioration)*
  - *La structure de vos classes (graphe ou explication)*
  - *Les différents bugs subsistant toujours (mieux vaux le déclarer ouvertement que de tenter de les cacher ou de passer à coté)*
  - *Les améliorations que vous souhaiteriez implémenter si vous aviez eu plus de temps (en justifiant pourquoi).*
  - *Une conclusion reprenant ce qui vous a plu et là où vous avez éprouvé des difficultés.*

L’archive devra être envoyé par mail à l’adresse suivante :
damien.rohmer@imag.fr