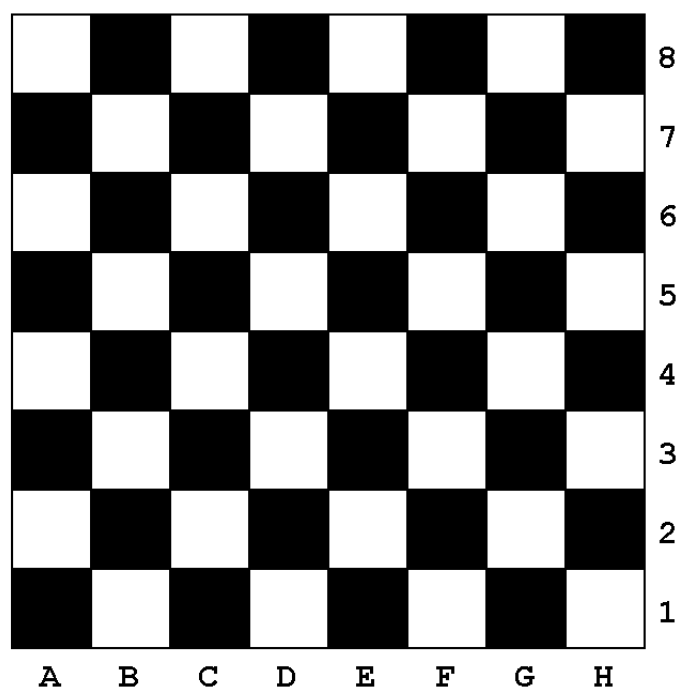


Dessins et graphismes

(Source : Beaucoup d'informations qui suivent relèvent du support officiel de la CNPI.)

Dans un environnement graphique, tout ce que nous voyons à l'écran (fenêtres, boutons, images, ...) doit être dessiné point par point à l'écran. Les éléments graphiques prédéfinis (JLabel, JButton, JSlider, JList, ...) qu'on a utilisés jusqu'à présent possèdent des méthodes internes qui les dessinent à l'écran. Toutes les interfaces graphiques ne peuvent cependant pas être réalisées en n'utilisant que des éléments prédéfinis. L'échiquier de l'image suivante par exemple, ne peut pas être réalisé avec ces éléments. Il faut le dessiner soi-même.

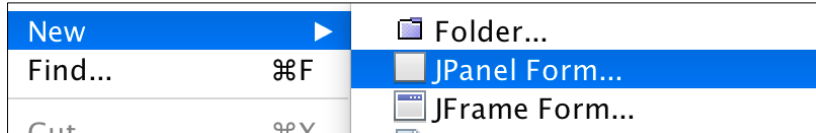


Le présent document décrit comment réaliser ses propres éléments graphiques à l'aide d'un panneau dont la classe Java s'appelle `JPanel`.

Configuration de base

On commence par la configuration de base qui est la même pour tous les éléments graphiques qu'on va créer.

1. Ajoutez un nouveau JPanel à votre projet et appelez le DrawPanel.



2. Cette nouvelle classe possède une vue « Source » et une vue « Design » (comme un « JFrame »). On aura surtout besoin de la vue « Source ». Activez cette vue et ajoutez la méthode suivante que l'on va utiliser pour dessiner notre élément graphique, par exemple l'échiquier d'en-haut.

```

25 public void paintComponent(Graphics g){
26     //ici on peut dessiner ce qu'on veut, par exemple un échiquier
27
28 }

```

3. Le compilateur va indiquer qu'il ne connaît pas la classe Graphics. Comme pour les ArrayList, il faut importer un paquet afin de pouvoir utiliser la classe Graphics avec ses méthodes, attributs et constantes `import java.awt.Graphics;`

On peut l'ajouter automatiquement en cliquant d'abord sur l'ampoule affichée dans le bord près des nombres de ligne, puis choisissant l'option **Add import for java.awt.Graphics**

4. Compilez votre projet en cliquant par exemple sur l'icône :



5. Ajoutez le DrawPanel à votre MainFrame. Activez d'abord la vue « Design » de la MainFrame. Placez alors le DrawPanel par glisser-déposer à partir de l'arbre de la partie gauche de la fenêtre sur la MainFrame.



6. Donnez le nom de variable `drawPanel` à votre élément graphique de la classe DrawPanel.

Le canevas « Graphics »

Un artiste peintre dessine sur un canevas (DE : *die Leinwand*). En informatique l'image de l'écran est aussi appelée canevas et elle consiste en une grille composée de pixels minuscules, des points qui peuvent prendre des millions de couleurs différentes.

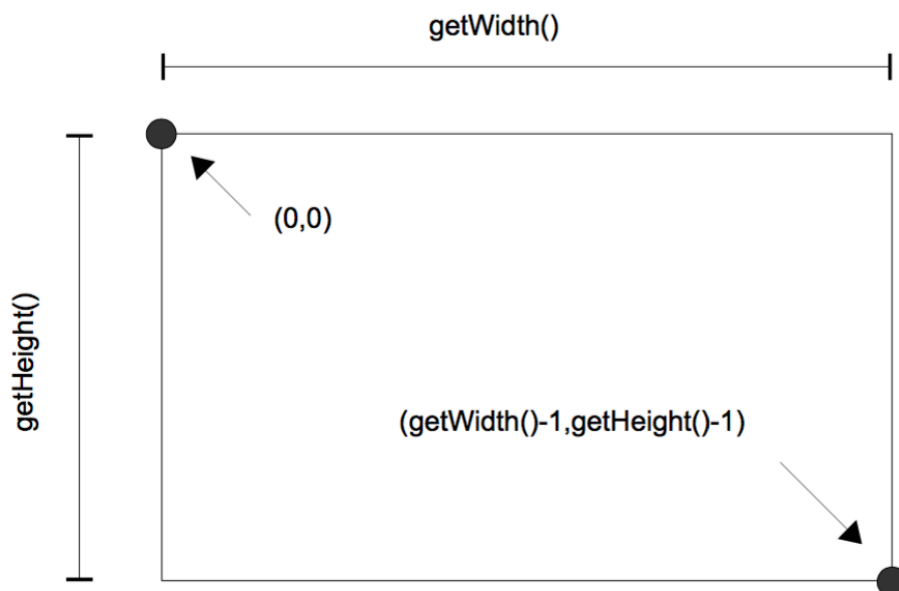
En Java, le canevas n'est rien d'autre qu'une instance de la classe `Graphics`. Celle-ci possède un grand nombre de méthodes relatives aux dessins. Tous les composants graphiques possèdent un tel objet.

La géométrie du canevas

Quant à la géométrie d'un canevas, il faut savoir que l'axe des Y est inversé par rapport au plan mathématique usuel. L'origine, c'est-à-dire le point $(0, 0)$ se trouve en haut à gauche. Le point en bas à droite possède les coordonnées $(\text{getWidth}() - 1, \text{getHeight}() - 1)$.

Attention !

Le canevas lui-même n'a pas d'attribut indiquant sa largeur ou sa hauteur. Voilà pourquoi ces données doivent être prises du panneau `JPanel` sur lequel on dessine.



Les méthodes utiles du canevas pour créer des dessins

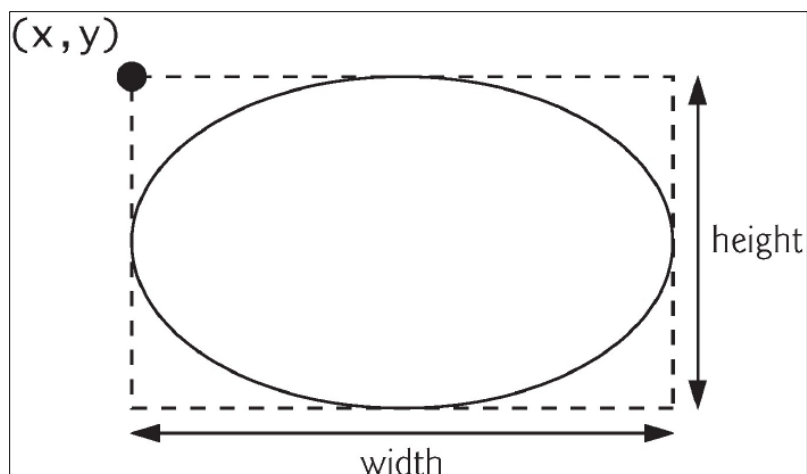
Méthode	Description
<code>Color getColor()</code> <code>void setColor(Color)</code>	Permet de récupérer la couleur actuelle, et d'en choisir une nouvelle.
<code>void drawLine(int x1, int y1, int x2, int y2)</code>	Dessine à l'aide de la couleur actuelle une ligne droite entre les points (x1,y1) et (x2,y2) .
<code>void drawRect(int x, int y, int width, int height)</code> <code>void fillRect(int x, int y, int width, int height)</code>	Dessine à l'aide de la couleur actuelle, un rectangle tel que (x,y) représente le point supérieur gauche, tandis que width et height représentent sa largeur respectivement sa hauteur.
<code>void drawOval(int x, int y, int width, int height)</code> <code>void fillOval(int x, int y, int width, int height)</code>	Dessine à l'aide de la couleur actuelle, une ellipse telle que (x,y) représente le point supérieur gauche, tandis que width et height représentent sa largeur respectivement sa hauteur.
<code>void drawString(String s, int x, int y)</code>	Dessine le texte s à la position (x,y) tel que (x,y) représente le point inférieur de l'alignement de base du texte.

Remarque :

On peut colorer un seul point (pixel) du canevas en traçant une 'ligne' d'un point vers le même point. P.ex. `g.drawLine(50,100,50,100);`

Les paramètres des méthodes `drawOval` et `fillOval`

Le dessin qui suit visualise la sémantique des paramètres des méthodes `drawOval` et `fillOval`



Utiliser des couleurs

Il faut tout d'abord importer le paquet de la classe `Color` : `import java.awt.Color;`
 Pour changer la couleur de ce qu'on dessine on peut alors procéder comme montré à l'image suivante.

```
public void paintComponent(Graphics g){
    Color red = new Color(255, 0, 0);
    g.setColor(red);

    //tout ce qui est dessiné ici est en rouge

    Color green = new Color(0, 255, 0);
    g.setColor(green);

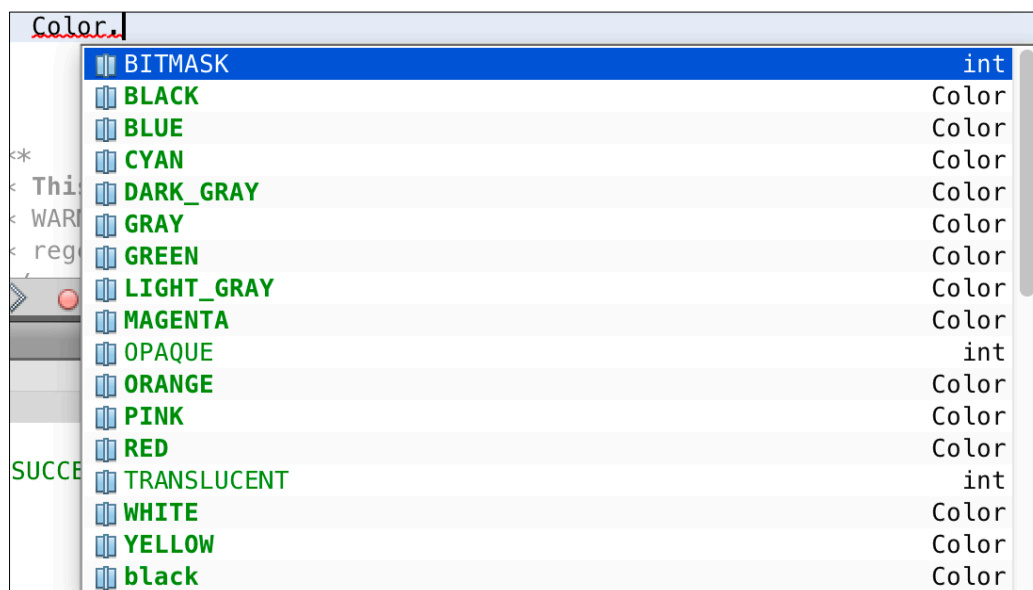
    //tout ce qui est dessiné ici est en vert

    Color black = Color.ORANGE;
    //tout ce qui est dessiné ici est en orange
}
```

Pour représenter des couleurs on utilise trois couleurs de base : rouge, vert, bleu (modèle additif RGB : Red, Green, Blue). Toute autre couleur est créée par différentes quantités de rouge, vert et bleu. Ces quantités sont des valeurs comprises dans l'intervalle [0, 255]. Pour savoir quelles quantités il faut utiliser pour représenter une certaine couleur on a des sites :

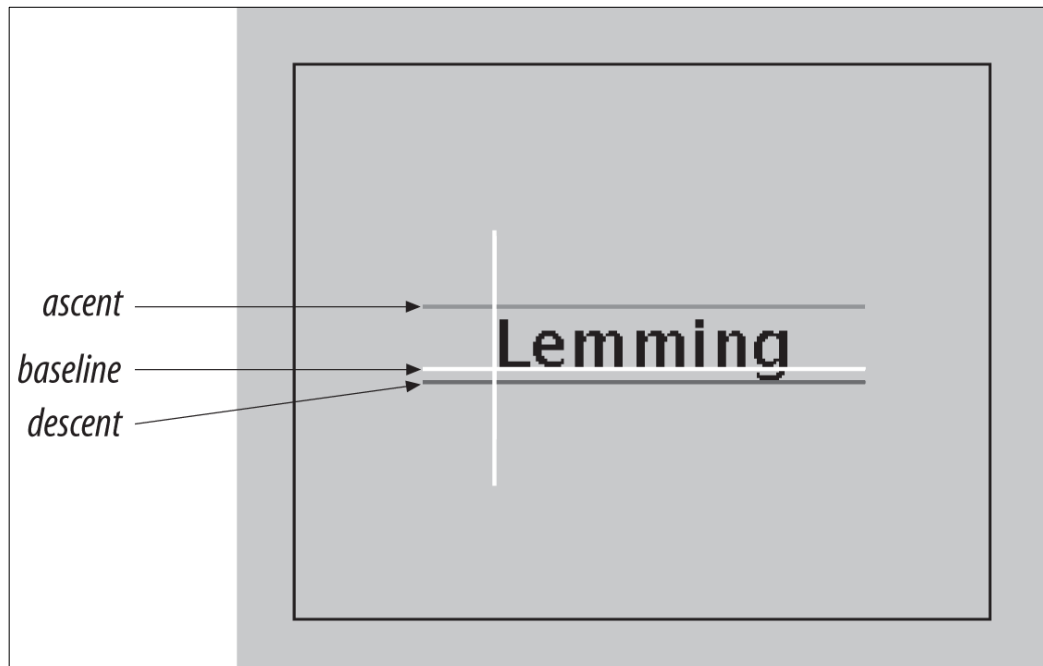
http://www.w3schools.com/colors/colors_picker.asp

Pour certaines couleurs il y a des constantes prédéfinies (3^{ème} exemple de l'image d'en-haut). Vous pouvez voir s'il y a une telle constante en tapant `Color.` puis utilisant le raccourci **CTRL + espace**. Une fenêtre comme à l'image suivante va alors apparaître, vous permettant de choisir une des couleurs affichées.



Les chaînes de caractères et la méthode `drawString`

Lorsqu'on dessine des chaînes de caractères il est utile de savoir que les coordonnées x et y de la méthode `drawString` correspondent à l'intersection des deux lignes blanches de l'image qui suit :



Les différentes polices (EN : font, DE : Schriftarten)

On peut changer la police utilisée pour dessiner une chaîne de caractères en procédant comme dans l'exemple suivant :

```
public void paintComponent(Graphics g){
    int fontSize = 16;
    g.setFont(new Font("Courier New", Font.BOLD, fontSize));

    //Dans cette partie-ci toute chaîne est dessinée avec la police Courier New

    int fontSize2 = 14;
    g.setFont(new Font("Times New Roman", Font.PLAIN, fontSize2));

    //Dans cette partie-ci toute chaîne est dessinée avec la police Times New Roman
}
```

Le premier paramètre constitue le nom de la police. Le deuxième paramètre indique si la police doit être affichée en gras, en italique (`Font.ITALIC`) ou normalement. Finalement le troisième paramètre indique la taille en points (EN : pixels)

Il faut aussi importer le paquet des polices avec l'instruction `import java.awt.Font;`

Déterminer la taille du dessin d'une chaîne de caractères

Pour déterminer les différentes informations relatives à la taille du dessin d'une chaîne de caractères on peut procéder comme à l'image suivante.

```
public void paintComponent(Graphics g){  
    FontMetrics fm = g.getFontMetrics();  
    String s = "test";  
    int stringWidth = fm.stringWidth(s);  
    int stringAscent = fm.getMaxAscent();  
    int stringDescent = fm.getMaxDescent();  
}
```

A nouveau il faut importer le paquet de la classe `FontMetrics` avant de pouvoir utiliser celle-ci (`import java.awt.FontMetrics;`)