COURS – TD 9: DESSINER EN JAVA

Où dessiner ?

- Uniquement sur le composant Canvas en AWT
- Sur tous les composants héritant de JComponent en SWING
- Dans tous les cas, on utilise le contexte graphique
  - Graphics
  - Graphics2D (depuis JDK 1.2)

Contexte Graphique

- Utilise la classe Graphics
- Fonctionne sous la forme d'états graphique
- Comporte
  - Le composant sur lequel on dessine
  - La couleur
  - La fonte
  - Le rectangle de découpe (clipping)
  - Les transformations (translation, rotation)

Changement d'états du contexte

- **setClip**(int x, int y, int width, int height)
- **setClip**(Shape clip)
- **setColor**(Color c)
- **setFont**(Font font)
- **translate**(int x, int y)

Couleurs

- Couleurs
  - Color(float r, float g, float b)
  - Color(float r, float g, float b, float a)
  - Color(int r, int g, int b)
  - Color(int r, int g, int b, int a)

- Des couleurs prédéfinies :
  - RED,
  - YELLOW,
  - GREEN,
  - BLUE,
  - …

Fontes

- Font(String name, int style, int size)
  - Name :
    - Constantes de la classe Fonte
      - DIALOG,
      - DIALOG_INPUT,
      - MONOSPACED,
      - SANS_SERIF,
      - SERIF
    - Font[] GraphicsEnvironment.getAllFonts()
    - String[] GraphicsEnvironment.getAvailableFontFamilyNames()
  - Style : Constantes de la classe Fonte
    - PLAIN,
    - ITALIC,
    - BOLD

Primitive de dessin

- **Dessiner les contours**
  - **draw3DRect**(int x, int y, int width, int height, boolean raised)
  - **drawArc**(int x, int y, int width, int height, int startAngle, int arcAngle)
  - **drawLine**(int x1, int y1, int x2, int y2)
  - **drawOval**(int x, int y, int width, int height)
  - **drawPolygon**(int[] xPoints, int[] yPoints, int nPoints)
  - **drawPolygon**(Polygon p)
  - **drawPolyline**(int[] xPoints, int[] yPoints, int nPoints)
  - **drawRect**(int x, int y, int width, int height)
  - **drawRoundRect**(int x, int y, int width, int height, int arcWidth, int arcHeight)

- Remplir la forme : Fill à la place de draw

Affichage de texte ou d'image

- Texte
  - drawBytes(byte[] data, int offset, int length, int x, int y)
  - drawChars(char[] data, int offset, int length, int x, int y)
  - drawString(String str, int x, int y)

- Image
  - drawImage(Image img, int x, int y, Color bgcolor, ImageObserver observer)
  - drawImage(Image img, int x, int y, ImageObserver observer)
  - drawImage(Image img, int x, int y, int width, int height, Color bgcolor, ImageObserver observer)
  - drawImage(Image img, int x, int y, int width, int height, ImageObserver observer)
  - drawImage(Image img, int dx1, int dy1, int dx2, int dy2, int sx1, int sy1, int sx2, int sy2, Color bgcolor, ImageObserver observer)
  - drawImage(Image img, int dx1, int dy1, int dx2, int dy2, int sx1, int sy1, int sx2, int sy2, ImageObserver observer)

Création du contexte graphique

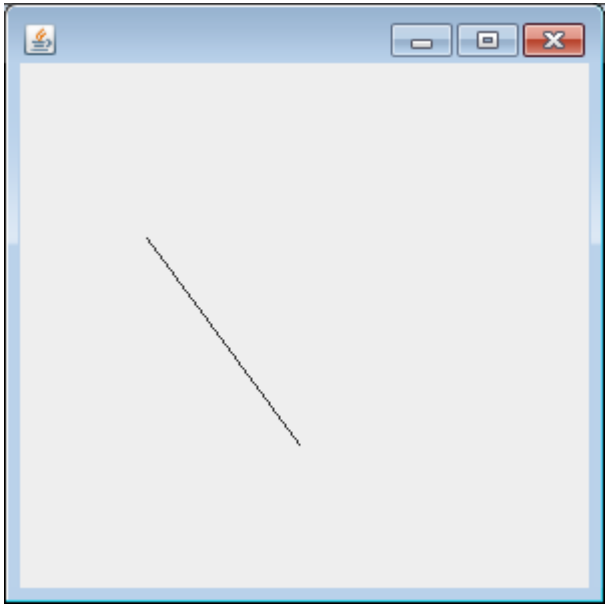
- On ne peut pas instancier un objet de type Graphics

- Pour l'utiliser :
  - Redéfinir la méthode
    - void paint(Graphics g), void update(Graphics g) de la classe Canvas (AWT)
    - protected void paintComponent(Graphics g) de la classe Jcomponent
  - En appelant la méthode Graphics getGraphics() d'un composant ou d'une image
  - En appelant la méthode createGraphics() à partir d'un Graphics existant.

- Les Graphics obtenus explicitement doivent être libérés avec la méthode dispose().

Exercice

- Dessiner une ligne
  - créée sur le MousePressed
  - Mise à jour sur le MouseDragged
  - Figée sur le MouseReleased



Contexte Graphique étendu

- Fonctionnalités supplémentaires avec Graphics2D
  - Propriété d'affichage : RenderingHints
  - Style de trait : Stroke
  - Outil de remplissage : Paint
  - Transformation géométrique : AffineTransform
  - Forme à afficher : Shape

Propriétés d'affichage

- abstract void setRenderingHint(RenderingHints.Key hintKey, Object hintValue)
  - RenderingHints.Key
    - KEY_ALPHA_INTERPOLATION
    - KEY_ANTIALIASING
    - KEY_COLOR_RENDERING
    - KEY_DITHERING
    - KEY_FRACTIONALMETRICS
    - KEY_INTERPOLATION
    - KEY_RENDERING
    - KEY_STROKE_CONTROL
    - KEY_TEXT_ANTIALIASING
    - KEY_TEXT_LCD_CONTRAST
  - Value
    - VALUE_ANTIALIAS_OFF
    - VALUE_ANTIALIAS_ON

Caractéristiques des traits

- abstract void setStroke(Stroke s)

- Classe BasicStroke
  - BasicStroke()
  - BasicStroke(float width)
  - BasicStroke(float width, int cap, int join)
  - BasicStroke(float width, int cap, int join, float miterlimit)
  - BasicStroke(float width, int cap, int join, float miterlimit, float[] dash, float dash_phase)

CAP_BUTT
JOIN_BEVEL

CAP_SQUARE
JOIN_MITER

CAP_ROUND
JOIN_ROUND

Couleurs / Dégradés / Textures

- abstract void setPaint(Paint paint)

- Interface Paint
  - Color,
  - GradientPaint,
  - LinearGradientPaint,
  - MultipleGradientPaint,
  - RadialGradientPaint,
  - TexturePaint

Transformations géométriques

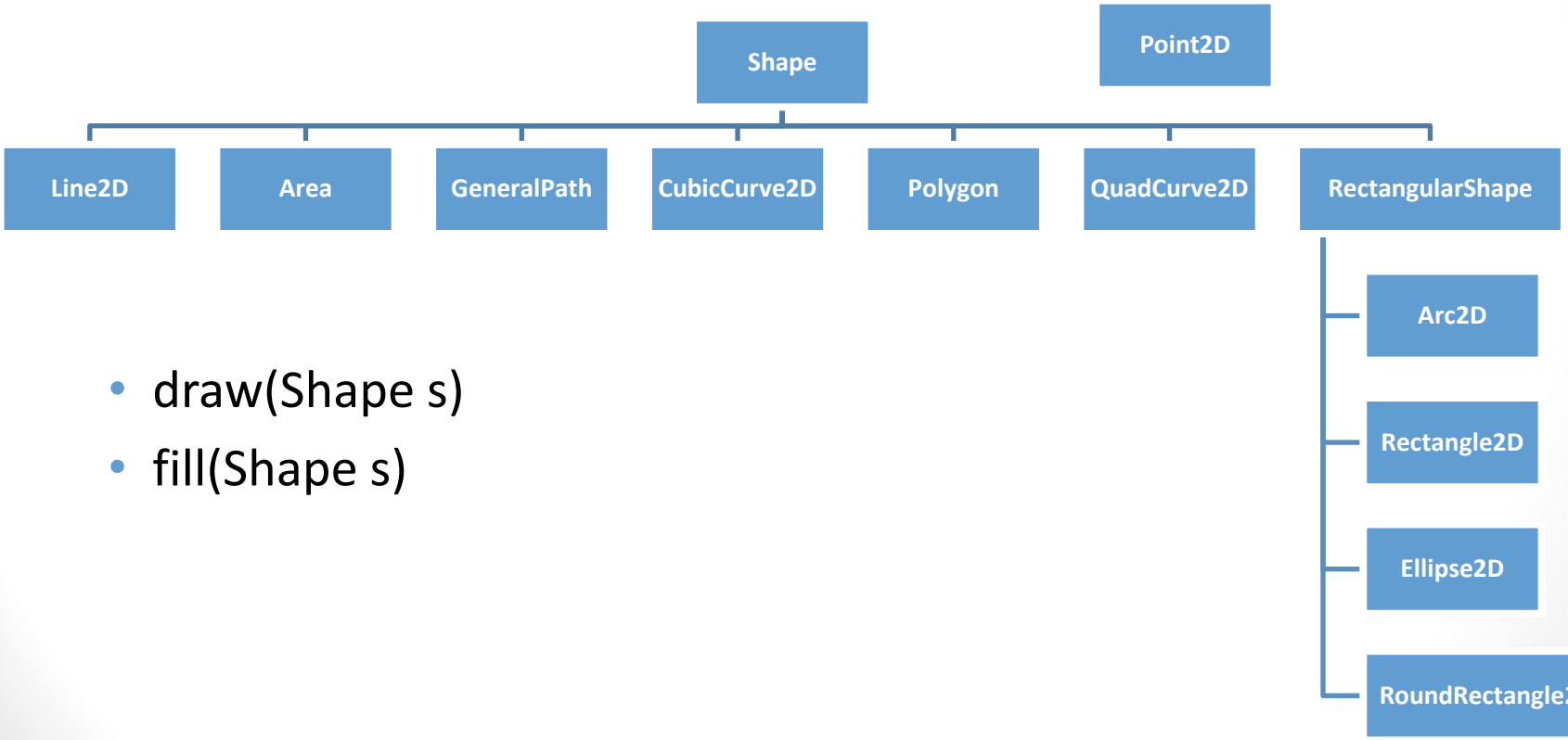
- abstract void **setTransform**(AffineTransform Tx)

- AffineTransform(double[] flatmatrix)
- AffineTransform(double m00, double m10, double m01, double m11, double m02, double m12)

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} m00 & m01 & m02 \\ m10 & m11 & m12 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} m00x + m01y + m02 \\ m10x + m11y + m12 \\ 1 \end{bmatrix}$$

Les formes graphiques

- Interface Shape
  - Coordonnées réelles



- draw(Shape s)
- fill(Shape s)

Les aires

- La class Area
  - Area(Shape s)
- Méthodes de combinaison de forme
  - add(Area rhs)
  - exclusiveOr(Area rhs)
  - intersect(Area rhs)
  - subtract(Area rhs)

Double buffer

- On crée l'image dans un BufferedImage
  - BufferedImage(int width, int height, int imageType)
  - Graphics2D createGraphics()

- Puis on l'affiche
  - abstract void drawImage(BufferedImage img, BufferedImageOp op, int x, int y)

Timer

- Permet d'envoyer un ActionEvent toutes les N millisecondes
  - Continue tant qu'on ne l'arrete pas
- Celui du package javax.swing
- **Timer**(int delay, ActionListener listener)
- Méthodes
  - start()
  - stop()

Exercice 2 : L'horloge