



Программирование графики

Лекция 7

Цели

- Описать рисование графики
- Описать графические классы
- Обсудить классы шрифтов
- Описать класс FontMetrics
- Описать класс Color



Graphics

- **Graphics** - это (абстрактный) класс.
- Каждый контейнер и компонент, который можно нарисовать на экране, имеет связанный с ним объект класса **Graphics**.
- Этот связанный объект класса **Graphics** содержит данные, определяющие, какую область экрана занимает данный компонент или контейнер.
- Объект **g** типа **Graphics** также содержит все методы, которые мы будем использовать для рисования на экране таких фигур, как окружности и прямоугольники.
- Класс **Graphics** входит в состав пакета (библиотеки) AWT, и поэтому при его использовании необходимо включать в свою программу следующий оператор импорта:

```
import java.awt.*;
```

Класс Graphics

Базовые методы класса Graphics

- ❖ `public abstract void clearRect(int x, int y, int width, int height)` стирает содержимое прямоугольной области.
- ❖ `public abstract void clipRect(int x, int y, int width, int height)` задает область ограничения вывода.
- ❖ `public abstract void copyArea(int x, int y, int width, int height, int dx, int dy)` копирует содержимое прямоугольной области.
- ❖ `public abstract Graphics create()` или `public Graphics create(int x, int y, int width, int height)` создает контекст отображения.
- ❖ `public abstract void dispose()` удаляет контекст отображения.

Метод `paintComponent`

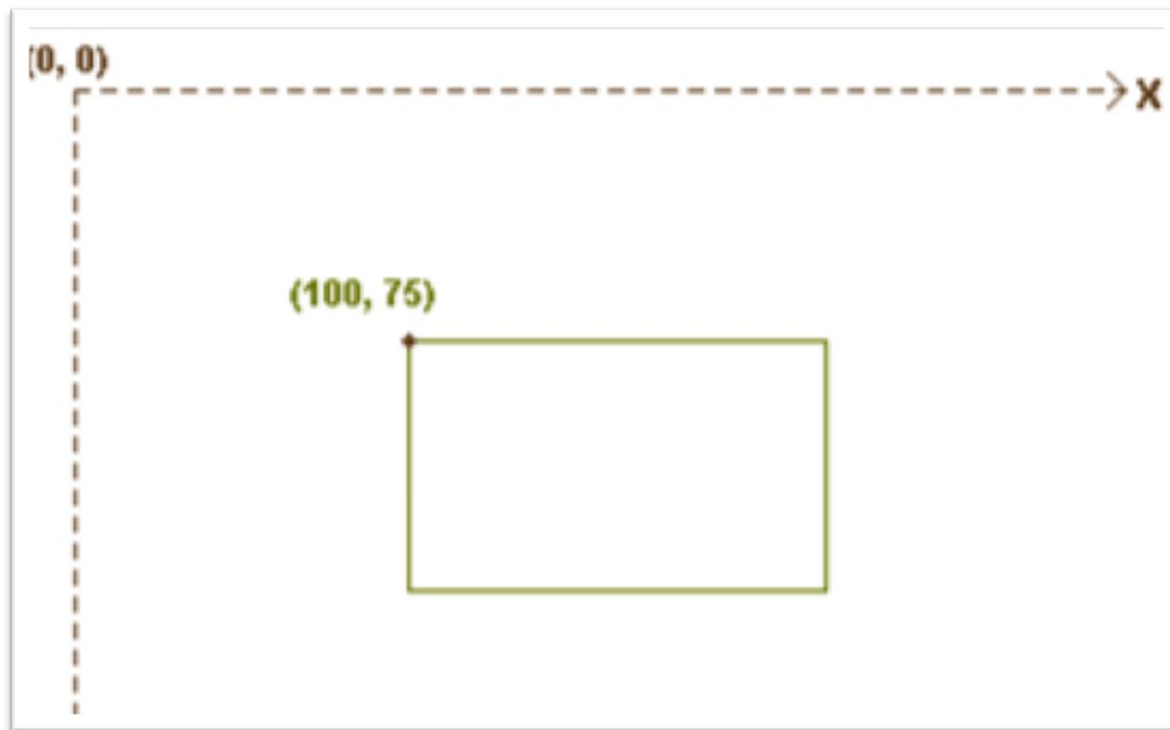
- На панелях класса `JPanel` можно рисовать фигуры и помещать эти `JPanel`-панели в окно класса `JFrame`.
- При определении класса `JPanel`, который содержит графические изображения, вместо метода `paint()` используется метод **`paintComponent()`**.
- Для того, чтобы изменить графическое изображение в окне и обновить его, изменения стали видны на экране, вызывается метод **`repaint ()`**.

Базовые фигуры

- Начало координат (origin) (0, 0) находится в верхнем левом углу экранной области, используемой для рисования (обычно в этом качестве выступает объект класса JFrame или JPanel).
- Координата **x** (горизонтальная координата) имеет положительное значение, которое увеличивается при перемещении вправо.
- Координата **y** (вертикальная координата) имеет положительное значение, которое увеличивается при перемещении вниз.

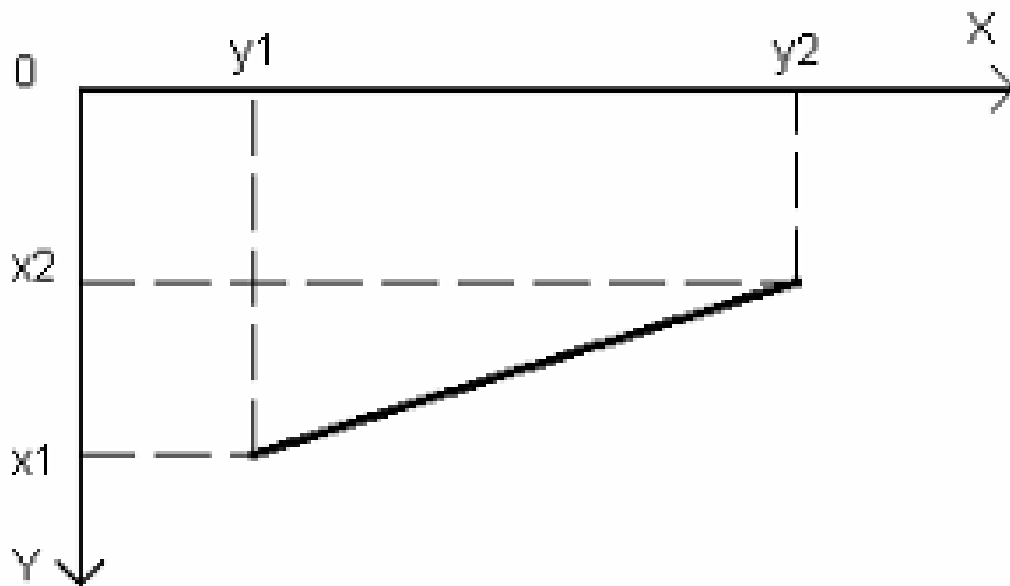
Базовые фигуры

- `public abstract void translate(int x, int y)` - сдвигает начало системы координат в контексте отображения таким образом, что оно перемещается в точку с координатами (x, y) .



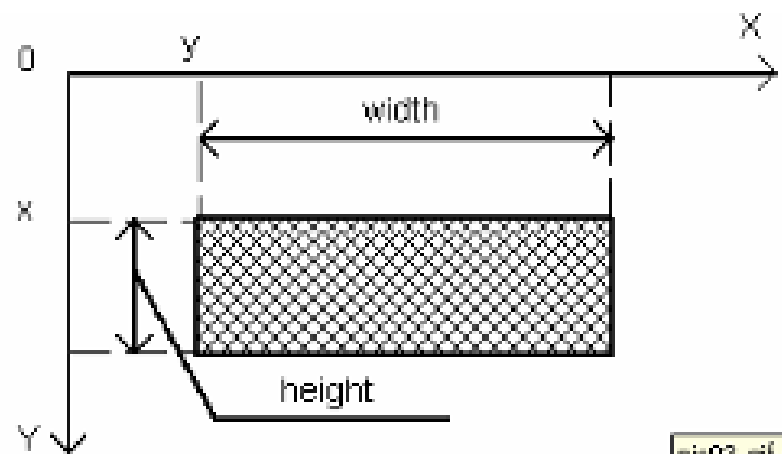
Методы класса Graphics

- `public abstract void drawLine(int x1, int y1, int x2, int y2)`
рисует текущим цветом отрезок прямой между точками с координатами $(x1, y1)$ и $(x2, y2)$.



Методы класса Graphics

- `public void drawRect(int x, int y, int width, int height)` рисует контур прямоугольника со сторонами, параллельными краям экрана, задаваемый координатами верхнего левого угла **(x, y)**, шириной **width** пикселей и высотой **height** пикселей.
- `public abstract void fillRect(int x, int y, int width, int height)` рисует залитый текущим цветом прямоугольник, задаваемый координатами верхнего левого угла **(x, y)**, шириной **width** пикселей и высотой **height** пикс.

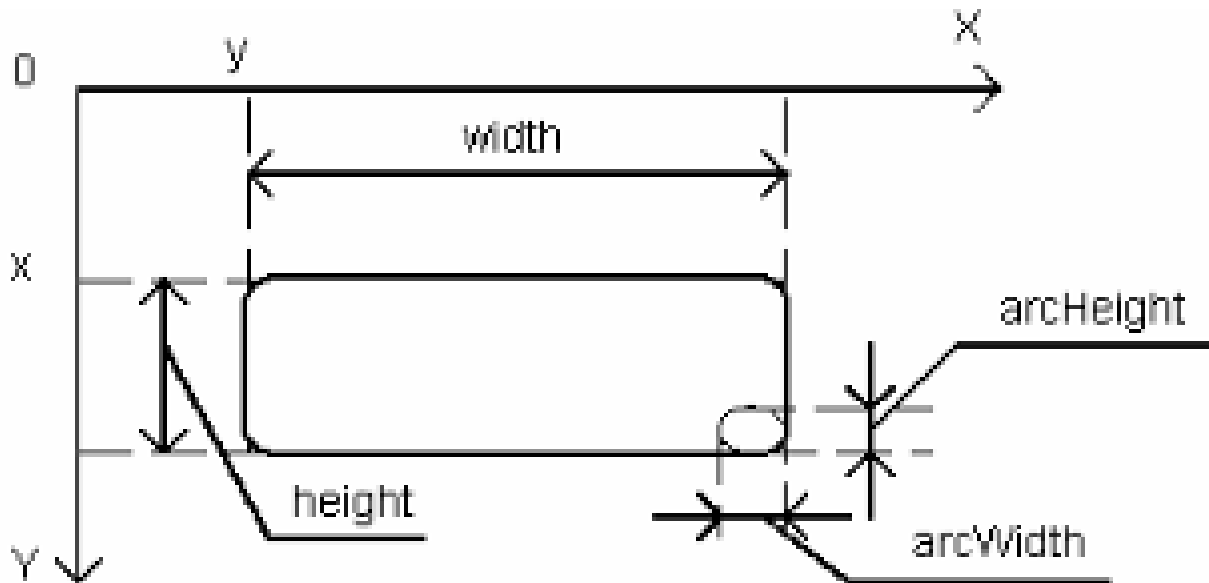


Методы класса Graphics для

- **public void draw3DRect** (int x, int y, int width, int height, boolean raised) рисует контур заданного прямоугольника. Координаты (x, y) определяют местоположение верхнего левого угла. Прямоугольник выделяется, чтобы создавалось впечатление, что он имеет толщину. Если аргумент raised равен значению true, создается впечатление, будто прямоугольник выступает над фоновой поверхностью. В противном случае (raised равен false) — впечатление утопленности в фоновой поверхности.
- **public void fill3DRect** (int x, int y, int width, int height, boolean raised) рисует залитый текущим цветом прямоугольник, как будто выделяющийся из плоскости рисования, если аргумент raised равен true, или как будто вдавленный в плоскость, если аргумент raised равен false.

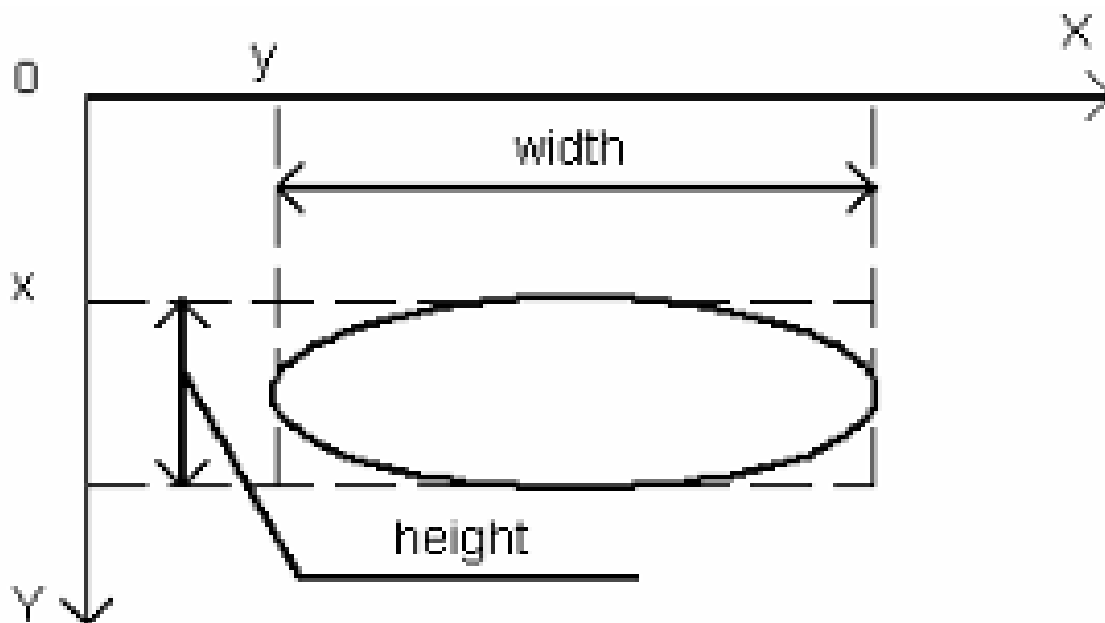
Методы класса Graphics

- **public abstract void drawRoundRect** (int x, int y, int width, int height, int arcWidth, int arcHeight) рисует контур прямоугольника со скругленными углами. Координаты (x, y) определяют местоположение верхнего левого угла подразумеваемого (описанного) обычного прямоугольника. Аргументы **arcwidth** и **arcHeight** задают форму скругленных углов.
- **public abstract void fillRoundRect**(int x, int y, int width, int height, int arcWidth, int arcHeight) рисует залитый текущим цветом прямоугольник со скругленными углами, заданный аргументами метода.



Методы класса Graphics

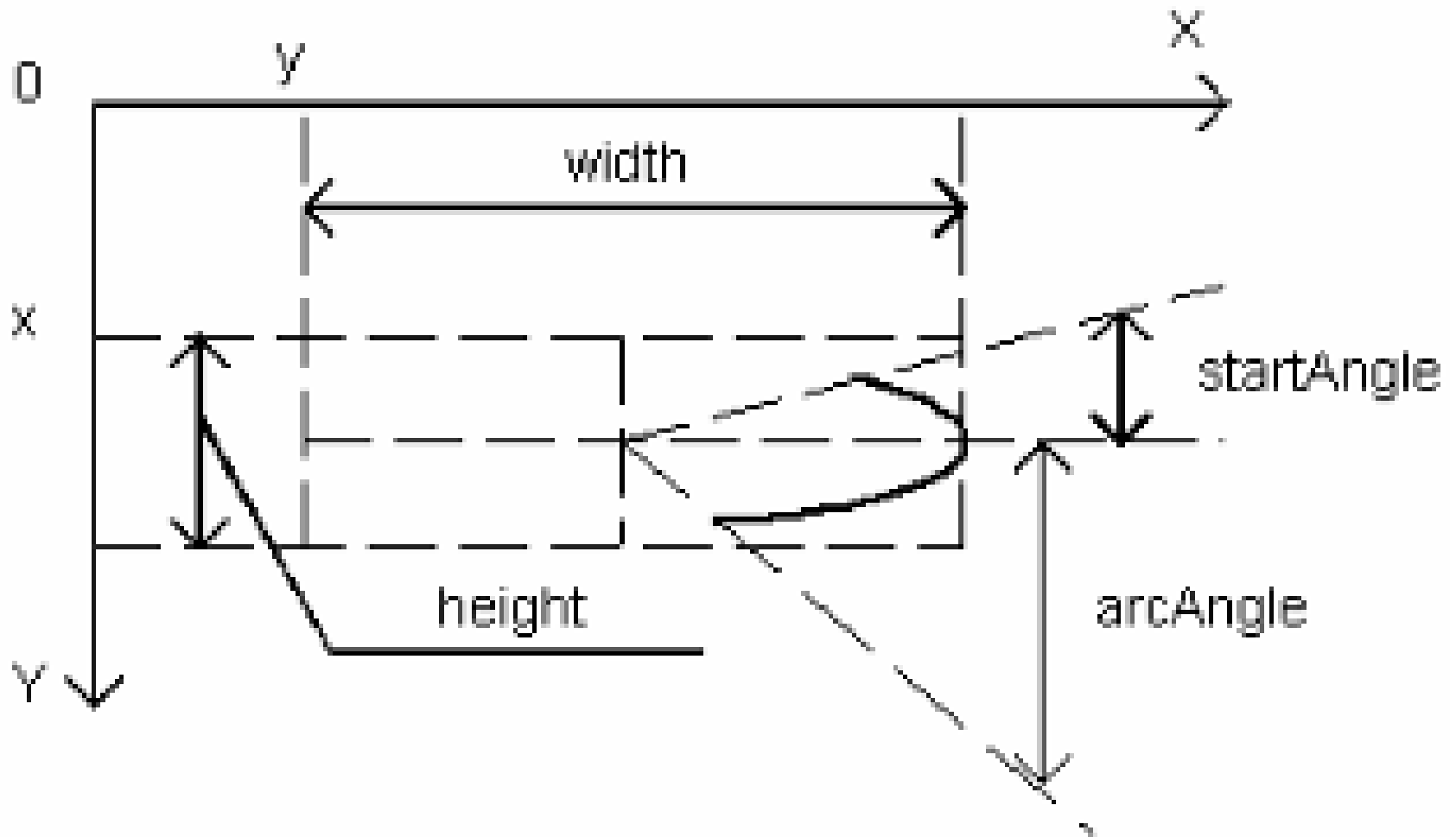
- `public abstract void drawOval(int x, int y, int width, int height)` рисует контур эллипса, вписанный в воображаемый прямоугольник с заданными значениями ширины **width** и высоты **height**. Координаты (x, y) определяют местоположение верхнего левого угла этого воображаемого прямоугольника. Если **width = height**, то получится окружность.
- `public abstract void fillOval(int x, int y, int width, int height)` рисует залитый текущим цветом эллипс, заданный аргументами метода.



Методы класса Graphics

- `public abstract void drawArc(int x, int y, int width, int height, int startAngle, int arcSweep)` рисует часть эллипса, который вписан в невидимый прямоугольник, заданный первыми четырьмя аргументами. Видимая часть эллипса задана последними двумя аргументами. Дуга имеет величину `arcSweep` градусов и отсчитывается от угла `startAngle`. Угол отсчитывается в градусах от оси Oх. Положительный угол отсчитывается против часовой стрелки, отрицательный — по часовой стрелке.
- `public abstract void fillArc(int x, int y, int width, int height, int startAngle, int arcSweep)` рисует залитый текущим цветом часть эллипса, заданную аргументами метода.

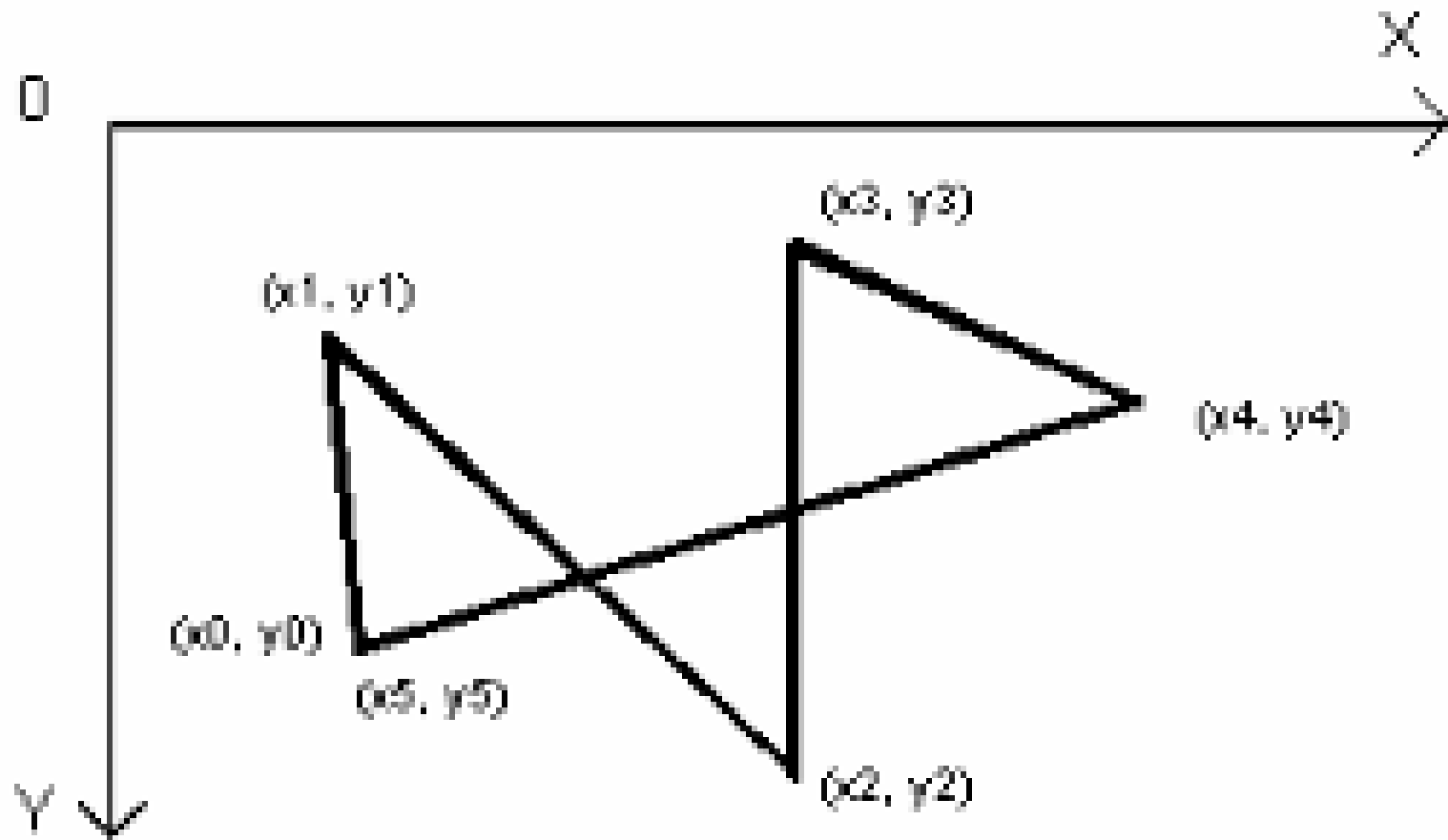
Методы класса Graphics



Методы класса Graphics

- `public abstract void drawPolyline(int[] x, int[] y, int points)`
рисует ломаную с вершинами в точках `(x[0], y[0]), (x[1], y[1]), ... (x[points-1], y[points-1])` и числом вершин `points`.
- `public abstract void drawPolygon(int[] x, int[] y, int points)`
рисует многоугольник с вершинами в точках `(x[0], y[0]), (x[1], y[1]), ... (x[points-1], y[points-1])` и числом вершин `points`. Если координаты первой и последней точки не равны, рисует линию, соединяющую последнюю точку с первой.
- `public abstract void fillPolygon(int[] x, int[] y, int points)`
рисует залитый текущим цветом многоугольник, заданный аргументами метода.

Методы класса Graphics



Определение цвета

- **Цвет**, как и все в Java, — объект определенного класса, а именно, класса **Color**.
- Основу класса **Color** составляют несколько конструкторов цвета.
 - ❖ **public Color** (int red, int green, int blue)
 - ❖ **public Color** (float red, float green, float blue)
 - ❖ **public Color** (int rgb)
 - ❖ **public Color** (int red, int green, int blue, int alpha) и **Color**(float red, float green, float blue, float alpha)
 - ❖ **public Color** (ColorSpace cspace, float[] components, float alpha)
- Метод, устанавливающий цвет для рисования в контексте отображения:
 - **public abstract void setColor** (Color c);

Класс Color

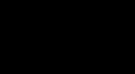
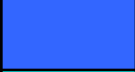
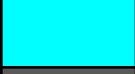










Примеры:

- ❖ `Color k = Color.red;`
- ❖ `Color a = new Color (255,255,0);`
- ❖ `Color b = new Color (0.907F,2F,0F);`

■ Для изменения или установки цветов компонента:

- ❖ `void setForeground(Color c)` класса **Component**, наследуемый различными компонентами
- ❖ `void setBackground(Color c)` класса **Component**, наследуемый различными компонентами

Класс Color

Объект	Цвет	Вид
<code>public final static Color black;</code>	черный	
<code>public final static Color blue;</code>	голубой	
<code>public final static Color cyan;</code>	циан	
<code>public final static Color darkGray;</code>	темно-серый	
<code>public final static Color gray;</code>	серый	
<code>public final static Color green;</code>	зеленый	
<code>public final static Color lightGray;</code>	светло-серый	
<code>public final static Color magenta;</code>	малиновый	
<code>public final static Color orange;</code>	оранжевый	
<code>public final static Color pink;</code>	розовый	
<code>public final static Color red;</code>	красный	
<code>public final static Color white;</code>	белый	
<code>public final static Color yellow;</code>	желтый	

Методы класса Color

- **public int getRed()** возвращает красную составляющую вызывающего объекта. Возвращаемое значение находится в диапазоне 0-255 (включительно).
- **public int getGreen()** возвращает зеленую составляющую вызывающего объекта. Возвращаемое значение находится в диапазоне 0-255 (включительно).
- **public int getBlue()** возвращает синюю составляющую вызывающего объекта. Возвращаемое значение находится в диапазоне 0-255 (включительно).
- **public int getAlpha()** возвращает компонент прозрачности вызывающего объекта. Возвращаемое значение находится в диапазоне 0-255 (включительно).
- **public int getRGB()** определяет компоненты RGB для цвета, выбранного в контекст отображения.
- **public Color brighter()** возвращает более яркую версию изображения вызывающего объекта.
- **public Color darker()** возвращает более темную версию изображения вызывающего объекта.

Методы класса Color

- `public ColorSpace getColorSpace()` возвращает цветовую модель цвета.
- `public float[] getComponents()` возвращает массив, содержащий цветовые компоненты и компонент прозрачности в указанной цветовой модели.
- `public static float[] RGBtoHSB (int r, int g, int b, float hsbvals[])` преобразует цвет, заданный тремя базовыми компонентами, в массив типа float со значениями HSB, соответствующими данному цвету.
- `public String toString()` возвращает текстовую строку названия цвета.

Класс Font

- Класс `java.awt.Font` используется для установки или извлечения характеристик шрифтов.
- Один из конструкторов класса Font:
 - ❖ `public Font(String name, int style, int pointsize)`
 - Именем `name` может быть “Times New Roman”, “Arial” и т.д.
 - Стилем `style` может быть `Font.PLAIN`, `Font.BOLD`, `Font.ITALIC`
 - Размером `pointsize` для шрифтов могут быть значения 11,12,14,16 и т.д.

Методы класса Font

- `public static Font getFont (String nm)` и `public static Font getFont (String nm, Font font)` получают шрифт по его характеристикам.
- `public String getName()` определяет название шрифта.
- `public int getSize()` определяет размер шрифта.
- `public int getStyle()` определяет стиль шрифта.
- `public boolean isBold()` проверяет, является ли шрифт жирным.
- `public boolean isItalic()` проверяет, является ли шрифт наклонным.
- `public boolean isPlain()` проверяет, есть ли шрифтовое выделение.
- `public String toString()` возвращает текстовой строки для объекта.

Класс FontMetrics

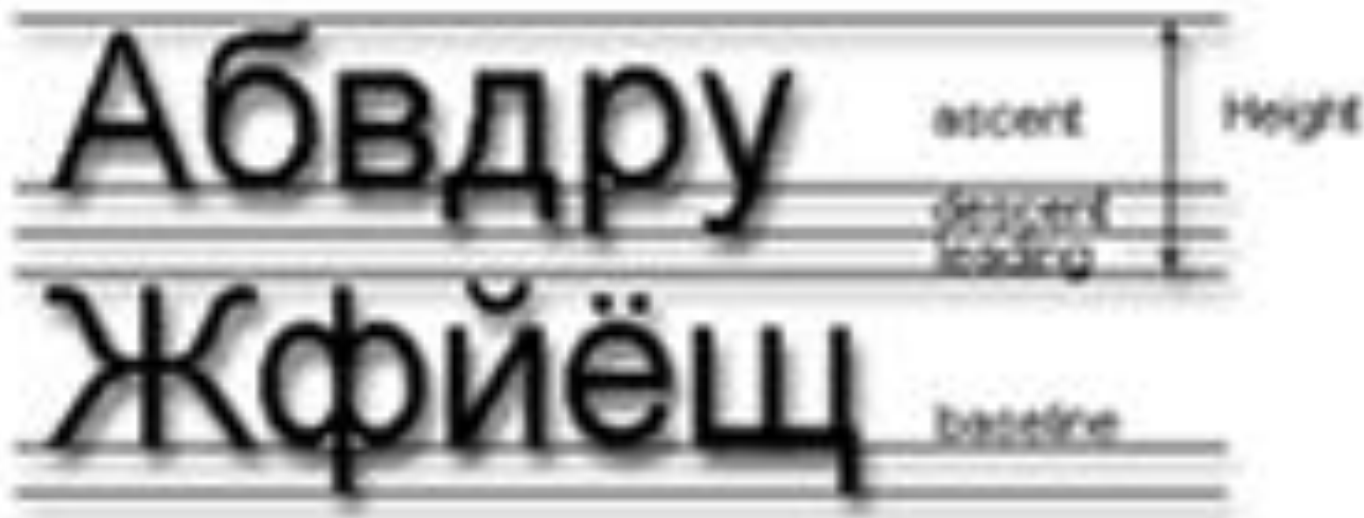
- При выводе строки в окно приложения очень часто возникает необходимость расположить ее определенным образом относительно других элементов изображения: центрировать, вывести над или под другим графическим объектом. Для этого надо знать метрику строки: ее высоту и ширину.
- Для измерения размеров отдельных символов и строки в целом разработан класс FontMetrics.
- Класс **FontMetrics** является абстрактным, поэтому нельзя воспользоваться его конструктором.
- Для получения объекта класса **FontMetrics**, содержащего набор метрических характеристик шрифта **f**, надо обратиться к методу **getFontMetrics()** класса **Graphics** или класса **Component**.

```
public FontMetrics getFontMetrics();
```


Класс FontMetrics

- Наиболее часто используемые методы класса `FontMetrics`:
 - ❖ `int stringWidth(String s)` – возвращает полную ширину (области) строки
 - ❖ `int charWidth(char c)` – возвращает ширину заданного символа
 - ❖ `int getHeight()` – возвращает общую высоту текущего шрифта

Класс FontMetrics



Объекты класса Toolkit

- Чтобы получить объект класса **Toolkit**, используется метод **getDefaultToolkit()** класса **Toolkit**.
- Список имен доступных шрифтов можно просмотреть следующими операторами:

```
String FontList[];  
FontList = getToolkit().getFontList();  
for (int i = 0; i < FontList.length; i++) {  
    System.out.println(i + ": " + FontList[i]);  
}
```

- Файлы с таблицами сопоставления логических и физических имен шрифтов находятся в папке `j2sdk1.5.0\jre\lib`.

Возможности Java 2D

- Интерфейс прикладного программирования Java 2D предоставляет усовершенствованные возможности двумерной графики для разработчиков, которым в их программах требуется сложная графика.
- Java 2D является составной частью Java 2 Platform, Standard Edition (J2SE).
- Java 2D включает функции для отображения линий, текста и изображений в пакетах `java.awt.image`, `java.awt.color`, `java.awt.font`, `java.awt.geom`, `java.awt.print` и `java.awt.image.renderable`.

Возможности Java 2D



JOIN_MITER



JOIN_ROUND



JOIN_BEVEL



CAP_BUTT



CAP_ROUND



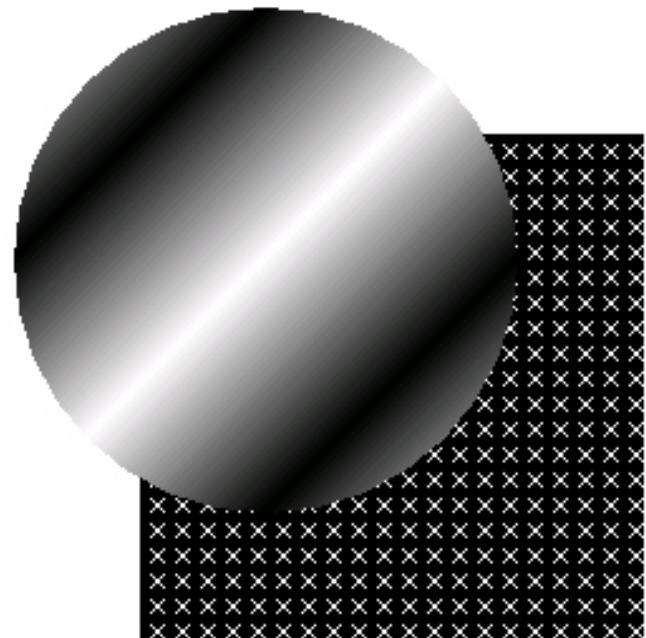
CAP_SQUARE



{10, 5, 5, 5, ...}



(5, 10, ...)



Возможности Java 2D

- **Цвета:** графика может быть нарисована градиентом или текстурой.
- **Прозрачное рисование:** Непрозрачность управляется через альфа-канал.
- **Локальные шрифты:** все шрифты, установленные в системе доступны.
- **Рисование:** толщина линий, изменение стилей.
- **Трансформация системы координат:** вращение, масштабирование, трансляция.