**Программирование графики**

**1)Посвящение в Swing:**

Swing стало официальным названием платформенно-независимого набора инструментальных средств для разработки ГПИ. у библиотеки Swing имеется ряд весьма существенных преимуществ:

• Содержит более богатый и удобный набор элементов пользовательского интерфейса.

• Намного меньше зависит от той платформы, на которой должна выполняться программа. Следовательно, она меньше подвержена ошибкам, характерным для конкретной платформы.

• Обеспечивает одинаковое восприятие конечными пользователями приложений с ГПИ на разных платформах.

**2) Наследование методов:**

Разумеется, благодаря наследованию в классе JFrame можно использовать методы из его суперклассов, задающие размеры и распоположение фрейма. К наиболее важным из них относятся следующие методы.

• Методы setLocation () и setBounds ( ) , устанавливающие положение фрейма.

• Метод dispose (), закрывающий окно и освобождающий все системные ресурсы, использованные при его создании.

• Метод setlconlmage () , сообщающий оконной системе, какая именно пиктограмма должна отображаться в строке заголовка, окне переключателя задач и т.п.

• Метод setTitle ( ) , позволяющий изменить текст в строке заголовка.

• Метод setResizable ( ) , получающий в качестве параметра логическое значение и определяющий, имеет ли пользователь право изменять размеры фрейма.

**3)Расположение окна:**

В приведенной ниже строке кода левый верхний угол фрейма размещается в точке, находящейся на расстоянии х пикселей вправо и у пикселей вниз от точки начала отсчета ( 0 ,0 ) в левом верхнем углу экрана

* setLocation(х,у)

Аналогично метод setBounds () из класса Component позволяет одновременно изменить размер и расположение компонента (в частности, объекта типа JFrame) с помощью следующего вызова:

* setBounds(х, у, width, height)

С другой стороны, можно предоставить оконной системе возможность самой управлять расположением окон. Так, если перед отображением окна сделать следующий вызов:

* setLoationByPlatform(true);

то оконная система сама выберет расположение (но не размеры) окна — как правило, с небольшим смещением относительно предыдущего окна.

**4)Title в java:**

По существу, title является свойством фрейма. При его установке предполагается, что заголовок окна на пользовательском экране изменится. А при получении данного свойства предполагается, что будет возвращено установленное в нем значение.

* public String getTitle();
* public void setTitle(String title);

//ВАЖНО: для свойств типа boolean имя метода получения начинается со слова «is».

**5)Определение размеров экрана:**

Чтобы определить размеры экрана, необходимо выполнить следующие действия.

* Toolkit kit = Toolkit.getDefaultToolkit();

Dimension screenSize = kit.getScreenSize();

int screenWidth = screenSize.width;

int screenHeight = screenSize.height;

Из этих размеров фрейма используется лишь половина для указания оконной системе расположения фрейма:

* setSize(screenWidth / 2, screenHeight / 2);

setLocationByPlatform(true);

Кроме того, для фрейма предоставляется пиктограмма. Процесс отображения рисунков на экране также зависит от операционной системы, и поэтому для загрузки рисунка снова потребуется объект типа Toolkit. Загруженный в итоге рисунок устанавливается затем в качестве пиктограммы, как показано ниже.

* Image img = new Imagelcon("icon.gif").getlmage();

setlconlmage(img);

**6)Рекомендации по работе с фреймом:**

Ниже дается ряд дополнительных рекомендаций по поводу обращения с фреймами.

• Если фрейм содержит только стандартные компоненты вроде кнопок и текстовых полей, вызовите метод pack (), чтобы установить размеры фрейма. Фрейм будет установлен с минимальными размерами, достаточными для размещения всех его компонентов. Зачастую главный фрейм программы приходится устанавливать с максимальными размерами. Фрейм можно развернуть до максимума, сделав следующий вызов:

frame.setExtendedState(Frame.MAXIMIZED \_BOTH);

• Целесообразно также запоминать расположение и размеры фрейма, заданные пользователем, чтобы восстановить эти значения при очередном запуске приложения

• Если вы разрабатываете приложение, в котором выгодно используются преимущества многоэкранного дисплея, применяйте классы GraphicsEnvironment и GraphicsDevice для определения размеров отдельных экранов.

• Класс GraphicsDevice позволяет также выполнять приложение в полноэкранном режиме.4

**7)Класс Component:**

Наиболее интересной для применения библиотеки Swing является панель содержимого. При оформлении фрейма его компоненты добавляются на панели содержимого с помощью следующего кода:

* Component c = . . . ;

frame.add(с);

Всякий раз, когда возникает потребность перерисовать окно независимо от конкретной причины, обработчик событий уведомляет об этом соответствующий компонент. В итоге метод paintComponent () выполняется для всех компнентов. Метод paintComponent (), т.е любой метод в классе, который реализует класс Component, вообще не следует вызывать вручную. Когда требуется перерисовать окно приложения, он вызывается автоматически, и вмешиваться в этот процесс не рекомендуется.

Вывод текста на экран считается особой разновидностью рисования. Для этой цели в классе Graphics имеется метод drawstring (), который вызывается следую­ щим образом:

* g.drawstring(text, х, у)

**8)Рисование 2D:**

Начиная с версии Java SE 2, такие методы, как paintComponent ( ) , автоматически получают объекты класса Graphics 2d. Нужно лиш ь произвести соответствующее приведение типов, как показано ниже.

* public void paintComponent(Graphics g)

{ Graphics2D g2 = (Graphics2D) g;

…}

При создании геометрических форм в библиотеке Java 2D применяется объектно-ориентированный подход. В частности, перечисленные ниже классы, предназначенные для рисования линий, прямоугольников и эллипсов, реализуют интерфейс Shape.

* Line2D
* Rectangle2D
* Ellipse2D

Чтобы нарисовать двухмерную форму, нужно сначала создать объект соответствующего класса, реализующего интерфейс Shape, а затем вызвать метод draw () из класса Graphics2D, как показано в следующем примере кода:

* Rectangle2D rect = … ;

g 2 .draw(rect);

**9)Приведение типов и абстрактные классы, которые облегчают всю жизнь:**

Это выражение не будет скомпилировано, поскольку константа 1 .2 имеет тип double, а компилятор зафиксирует потерю точности. В таком случае при формировании константы с плавающей точкой придется явно указать суффикс F следующим образом:

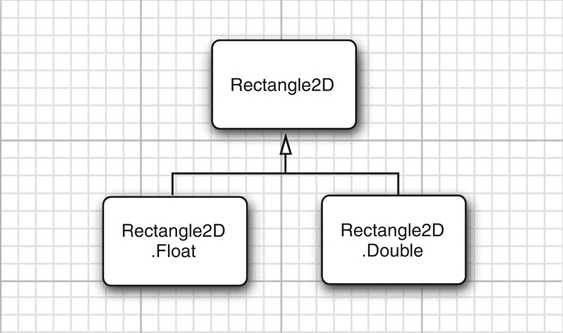
* float f = 1.2F; // Правильно!

Метод getWidth () возвращает число, имеющее тип double. На этот раз нужно выполнить приведение типов:

* float f = (float)г .getWidth(); // Правильно!

Разработчики данной библиотеки выбрали необычный и запутанный способ создания пакетов, соответствующих этим двум версиям. Рассмотрим в качестве примера класс Rectangle2D. Э го абстрактный класс, имеющий два следующих конкретных подкласса, каждый из которых является внутренним и статическим:

* Rectangle2D.Float
* Rectangle2D.Double

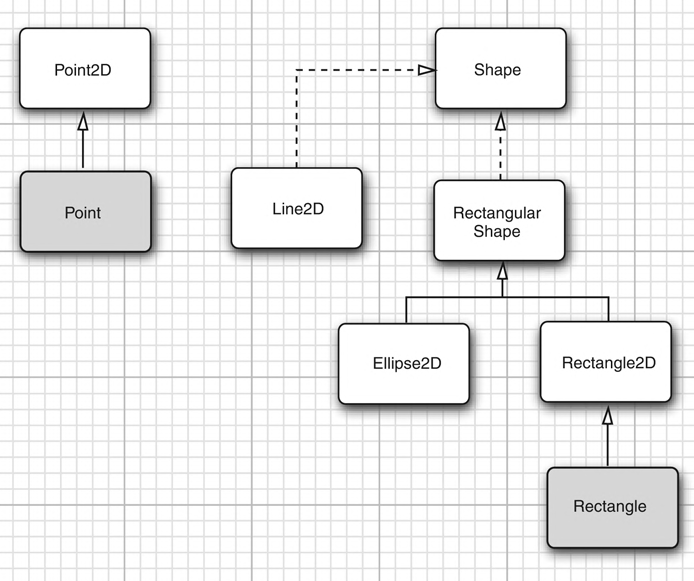


Все, что было сказано выше о классах Rectangle2D, относится к любым другим классам, предназначенным для рисования двухмерных форм. Кроме того, существует еще и класс Point2D с подклассами Point2D. Float и Point2D. Double. Ниже приведен пример создания объекта точки.

* Point2D р = new Point2D.Double(10, 20);

Классы Rectangle2D и Ellipse2D являются производными от одного и того же суперкласса RectangularShape. Как известно, эллипс не является прямоугольной фигурой, но он ограничен прямоугольником.

В классе RectangularShape определено более двадцати методов, общих для подобных двухмерных форм. К их числу относятся такие полезные методы, как getWidth (), getHeight (), getCenterX () и getCenterY ().



**10) Заливка цветом:**

Метод setPaint() из класса Graphics2D позволяет выбрать цвет, который будет применяться при всех дальнейших операциях рисования в графическом контексте. Ниже приведен пример применения этого метода в коде.

* g2 .setPaint(Color.RED);

g2.drawstring("Warning!", 100, 100);

Окрашивать цветом можно внутренние участки замкнутых геометрических форм вроде прямоугольников или эллипсов. Для этого вместо метода draw () достаточно вызвать метод fill () следующим образом:

* Rectangle2D rect = … ;

g 2 .setPaint(Color.RED);

g 2 .fill(rect); // заполнить прямоугольник красным цветом

Имеется возможность указать произвольный цвет по его красной, зеленой и синей составляющим, создав объект класса Color.

* Color(int redness, int greenness, int blueness)

Чтобы установить цвет фона, следует вызвать метод setBackground() из класса Component, предшественника класса Jpanel:

* MyComponent р = new MyComponent();

р.setBackground(Color.PINK);

11)Шрифтовое оформление текста:

Чтобы воспроизвести букву заданным шрифтом, сначала нужно создать объект класса Font , а затем указать название шрифта, его стиль и размер. Ниже показано, каким образом создается объект класса Font.

* Font sansboldl4 = new Font("SansSerif”, Font.BOLD, 14);

Первый параметр – это три гарнитуры шрифтов, которые всегда входят в Oracle: "Lucida Sans", " Lucida Bright" и "Lucida Sans Typewriter".

Второй параметр конструктора равным одному из следующих значений:

* Font.PLAIN
* Font.BOLD
* Font.ITALIC
* Font.BOLD + Font.ITALIC

В качестве третьего параметра в конструкторе класса F o n t задается размер шрифта. Для обозначения размера шрифта служит единица измерения, называемая пунктом. В одном дюйме содержится 72 пункта.

//Пример:

Font sansboldl4 = new Font("SansSerif", Font.BOLD, 14);

g 2.setFont(sansboldl4);

String message = "Hello, World!";

g2.drawstring(message, 75, 100); //