Как создать графический интерфейс с примерами Swing на Java

Swing в Java является частью базового класса Java, который является независимым от платформы. Он используется для создания оконных приложений и включает в себя такие компоненты, как кнопка, полоса прокрутки, текстовое поле и т. д.

Объединение всех этих компонентов создает графический интерфейс пользователя.

Что такое Swing в Java?

Swing в Java — это легкий инструментарий с графическим интерфейсом, который имеет широкий спектр виджетов для создания оптимизированных оконных приложений. Это часть JFC (Java Foundation Classes). Он построен на основе AWT API и полностью написан на Java. Он не зависит от платформы в отличие от AWT и имеет легкие компоненты.

Создавать приложения становится проще, поскольку у нас уже есть компоненты GUI, такие как кнопка, флажок и т. д.

Контейнерный класс

Любой класс, в котором есть другие компоненты, называется контейнерным классом. Для создания приложений с графическим интерфейсом необходим как минимум один класс контейнеров.

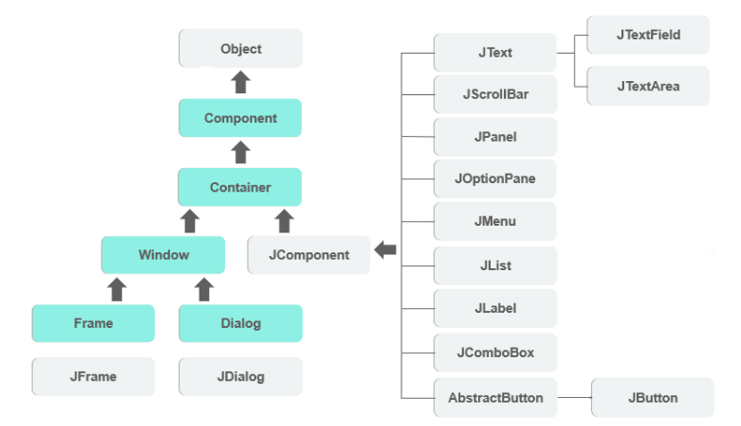
Ниже приведены три типа контейнерных классов:

1. Панель — используется для организации компонентов в окне.
2. Рамка — полностью функционирующее окно со значками и заголовками.
3. Диалог — это как всплывающее окно, но не полностью функциональное, как рамка.

Разница между AWT и Swing

|  |  |
| --- | --- |
| AWT | SWING |
| * платформо-зависимая | * не зависит |
| * не соответствует MVC | * соответствует MVC |
| * меньше компонентов | * Более мощные компоненты |
| * Не поддерживает подключаемый внешний вид и ощущение | * Поддержка подключаемого внешнего вида |
| * тяжеловесный | * легкий |

Иерархия



Объяснение: Все компоненты в свинге, такие как JButton, JComboBox, JList, JLabel, унаследованы от класса JComponent, который можно добавить в классы контейнера.

Контейнеры — это окна, такие как рамка и диалоговые окна. Основные компоненты являются строительными блоками любого графического приложения. Такие методы, как setLayout, переопределяют макет по умолчанию в каждом контейнере. Контейнеры, такие как JFrame и JDialog, могут добавлять только компонент к себе. Ниже приведены несколько компонентов с примерами, чтобы понять, как мы можем их использовать.

JButton Class

Он используется для создания помеченной кнопки. Использование ActionListener приведет к некоторым действиям при нажатии кнопки. Он наследует класс AbstractButton и не зависит от платформы.

Пример:

**import** javax.swing.\*;

**public** **class** example{

**public** **static** **void** main(**String** args[]) {

JFrame a = **new** JFrame("example");

JButton b = **new** JButton("click me");

b.setBounds(40,90,85,20);

a.add(b);

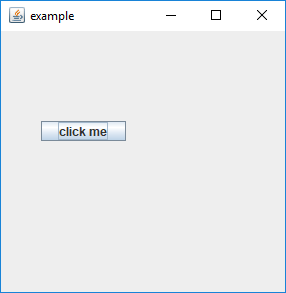
a.setSize(300,300);

a.setLayout(**null**);

a.setVisible(**true**);

}

}



JTextField Class

Он наследует класс JTextComponent и используется для редактирования однострочного текста.

Пример:

**import** javax.swing.\*;

**public** **class** example{

**public** **static** **void** main(**String** args[]) {

JFrame a = **new** JFrame("example");

JTextField b = **new** JTextField("edureka");

b.setBounds(50,100,200,30);

a.add(b);

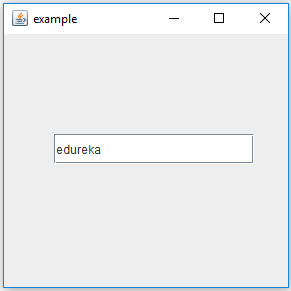
a.setSize(300,300);

a.setLayout(**null**);

a.setVisible(**true**);

}

}



JScrollBar Class

Он используется для добавления полосы прокрутки, как горизонтальной, так и вертикальной.

Пример:

**import** javax.swing.\*;

**class** example{

example(){

JFrame a = **new** JFrame("example");

JScrollBar b = **new** JScrollBar();

b.setBounds(90,90,40,90);

a.add(b);

a.setSize(300,300);

a.setLayout(**null**);

a.setVisible(**true**);

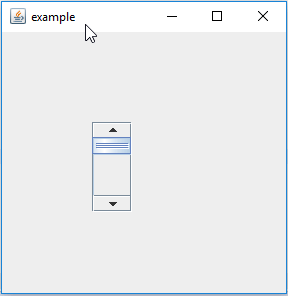
}

**public** **static** **void** main(**String** args[]){

**new** example();

}

}



JPanel Class

Он наследует класс JComponent и предоставляет пространство для приложения, которое может присоединить любой другой компонент.

**import** java.awt.\*;

**import** javax.swing.\*;

**public** **class** Example{

Example(){

JFrame a = **new** JFrame("example");

JPanel p = **new** JPanel();

p.setBounds(40,70,200,200);

JButton b = **new** JButton("click me");

b.setBounds(60,50,80,40);

p.add(b);

a.add(p);

a.setSize(400,400);

a.setLayout(**null**);

a.setVisible(**true**);

}

**public** **static** **void** main(**String** args[])

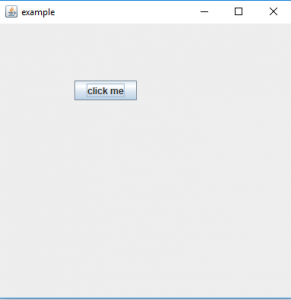
{

**new** Example();

}

}

Вывод:



JMenu Class

Он наследует класс JMenuItem и является компонентом выпадающего меню, которое отображается из строки меню.

**import** javax.swing.\*;

**class** Example{

JMenu menu;

JMenuItem a1,a2;

Example()

{

JFrame a = **new** JFrame("Example");

menu = **new** JMenu("options");

JMenuBar m1 = **new** JMenuBar();

a1 = **new** JMenuItem("example");

a2 = **new** JMenuItem("example1");

menu.add(a1);

menu.add(a2);

m1.add(menu);

a.setJMenuBar(m1);

a.setSize(400,400);

a.setLayout(**null**);

a.setVisible(**true**);

}

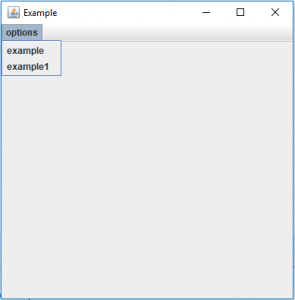
**public** **static** **void** main(**String** args[])

{

**new** Example();

}

}

Вывод:  


Класс JList

Он наследует класс JComponent, объект класса JList представляет список текстовых элементов.

**import** javax.swing.\*;

**public** **class** Example

{

Example(){

JFrame a = **new** JFrame("example");

DefaultListModel<**String**> l = **new** DefaultListModel< >();

l.addElement("first item");

l.addElement("second item");

JList<**String**> b = **new** JList< >(l);

b.setBounds(100,100,75,75);

a.add(b);

a.setSize(400,400);

a.setVisible(**true**);

a.setLayout(**null**);

}

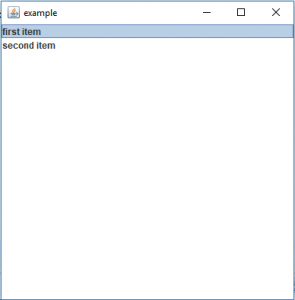
**public** **static** **void** main(**String** args[])

{

**new** Example();

}

}

Вывод:  


JLabel Class

Используется для размещения текста в контейнере. Он также наследует класс JComponent.

**import** javax.swing.\*;

**public** **class** Example{

**public** **static** **void** main(**String** args[])

{

JFrame a = **new** JFrame("example");

JLabel b1;

b1 = **new** JLabel("edureka");

b1.setBounds(40,40,90,20);

a.add(b1);

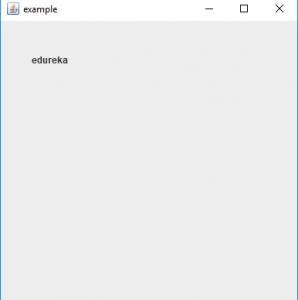
a.setSize(400,400);

a.setLayout(**null**);

a.setVisible(**true**);

}

}

Вывод:  


JComboBox Class

Он наследует класс JComponent и используется для отображения всплывающего меню выбора.

**import** javax.swing.\*;

**public** **class** Example{

JFrame a;

Example(){

a = **new** JFrame("example");

string courses[] = { "core java","advance java", "java servlet"};

JComboBox c = **new** JComboBox(courses);

c.setBounds(40,40,90,20);

a.add(c);

a.setSize(400,400);

a.setLayout(**null**);

a.setVisible(**true**);

}

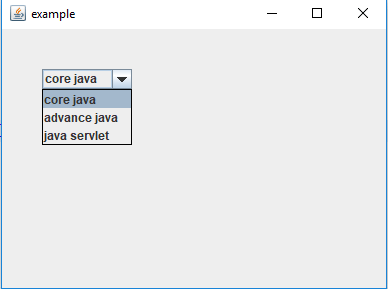
**public** **static** **void** main(**String** args[])

{

**new** Example();

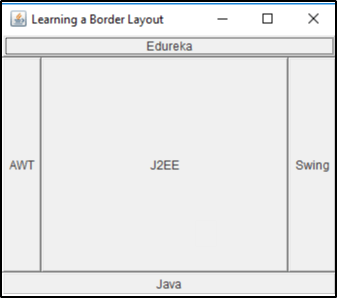
}

}

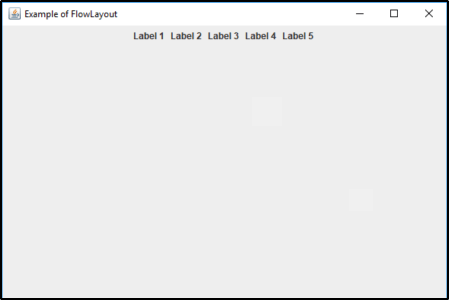
Вывод:  


Для размещения компонентов внутри контейнера мы используем менеджер макета. Ниже приведены несколько менеджеров макетов:

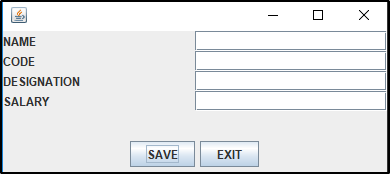
Макет границы

Менеджер по умолчанию для каждого JFrame — BorderLayout. Он размещает компоненты в пяти местах: сверху, снизу, слева, справа и по центру.  


Макет потока

FlowLayout просто кладет компоненты в ряд один за другим, это менеджер компоновки по умолчанию для каждого JPanel.  


GridBag Layout

GridBagLayout размещает компоненты в сетке, что позволяет компонентам охватывать более одной ячейки.  


Пример: фрейм чата

**import** javax.swing.\*;

**import** java.awt.\*;

**class** Example {

**public** **static** **void** main(**String** args[]) {

JFrame frame = **new** JFrame("Chat Frame");

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setSize(400, 400);

JMenuBar ob = **new** JMenuBar();

JMenu ob1 = **new** JMenu("FILE");

JMenu ob2 = **new** JMenu("Help");

ob.add(ob1);

ob.add(ob2);

JMenuItem m11 = **new** JMenuItem("Open");

JMenuItem m22 = **new** JMenuItem("Save as");

ob1.add(m11);

ob1.add(m22);

JPanel panel = **new** JPanel(); // the panel is not visible in output

JLabel label = **new** JLabel("Enter Text");

JTextField tf = **new** JTextField(10); // accepts upto 10 characters

JButton send = **new** JButton("Send");

JButton reset = **new** JButton("Reset");

panel.add(label); // Components Added using Flow Layout

panel.add(label); // Components Added using Flow Layout

panel.add(tf);

panel.add(send);

panel.add(reset);

JTextArea ta = **new** JTextArea();

frame.getContentPane().add(BorderLayout.SOUTH, panel);

frame.getContentPane().add(BorderLayout.NORTH, tf);

frame.getContentPane().add(BorderLayout.CENTER, ta);

frame.setVisible(**true**);

}

}

