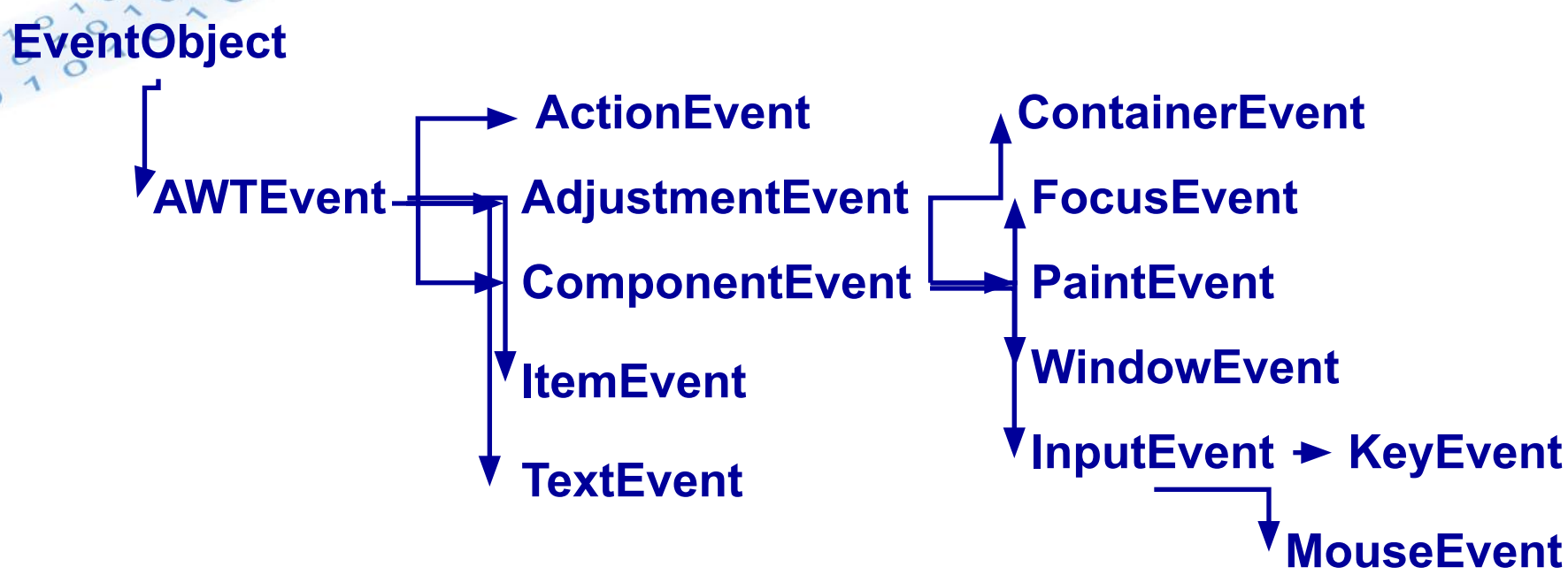
Обработка событий

Событие – объект специального класса, описывающий изменение состояния объекта источника.

Классы для обработки событий находятся в пакете java.awt.event

Обработка событий

Классы событий



Обработка событий

Классы событий

Методы класса EventObject:

- **Object getSource() – ссылка на объект-источник события**
- **String toString() – текстовое описание события**

Обработка событий

Источники событий

Источник	Событие	Когда возникает
JButton	ActionEvent	Нажата кнопка
JCheckBox	ItemEvent	Сброс/установка флажка
JRadioButton	ItemEvent	Сброс/установка флажка
JComboBox	ActionEvent ItemEvent	Выбор элемента списка Изменение состояния элемента
JList	ListSelection Event	Изменение состояния элемента
JMenuItem	ActionEvent ItemEvent	Изменилось сост-е пункта меню То же для пункта с меткой

Обработка событий

Источники событий

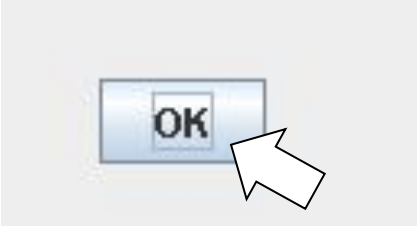
Источник	Событие	Когда возникает
JScrollbar	AdjustmentEvent	Манипуляции с полосами прокрутки
JTextField JTextArea	TextEvent	Ввод символов
JFrame JWindow Jdialog	WindowEvent	Любые действия с окном (открыть/закрыть, свернуть/развернуть и т.п.)
Все органы управления	FocusEvent	Компонент получает или теряет фокус ввода
Все контейнеры	ContainerEvent	Компонент добавляется или удаляется из контейнера

Обработка событий

Уровни событий

Высокоуровневые: все вышеперечисленные.

Низкоуровневые: KeyEvent, MouseEvent,



MouseEvent (move) - low

MouseEvent (click) – low

ActionEvent - high

Обработка событий

Интерфейсы - слушатели

Для каждого типа событий существует свой интерфейс-слушатель:

ActionEvent – ActionListener

ItemEvent – ItemListener

WindowEvent – WindowListener

и т.д. …

В интерфейсе-слушателе описаны методы, обрабатывающие события данного класса.

Обработка событий

Блоки прослушивания

Блок прослушивания – класс, реализующий интерфейс-слушатель (реализующий все его методы).

Для обработки события необходимо:

1. **Создать блок прослушивания**
2. **Связать блок прослушивания с источником события**

Обработка событий

Блоки прослушивания

Чтобы связаться с блоком прослушивания, классы-источники события должны получить ссылку на экземпляр блока прослушивания (класса-обработчика события) – зарегистрировать блок прослушивания:

```
public void addXXXListener(XXXListener obj)
```

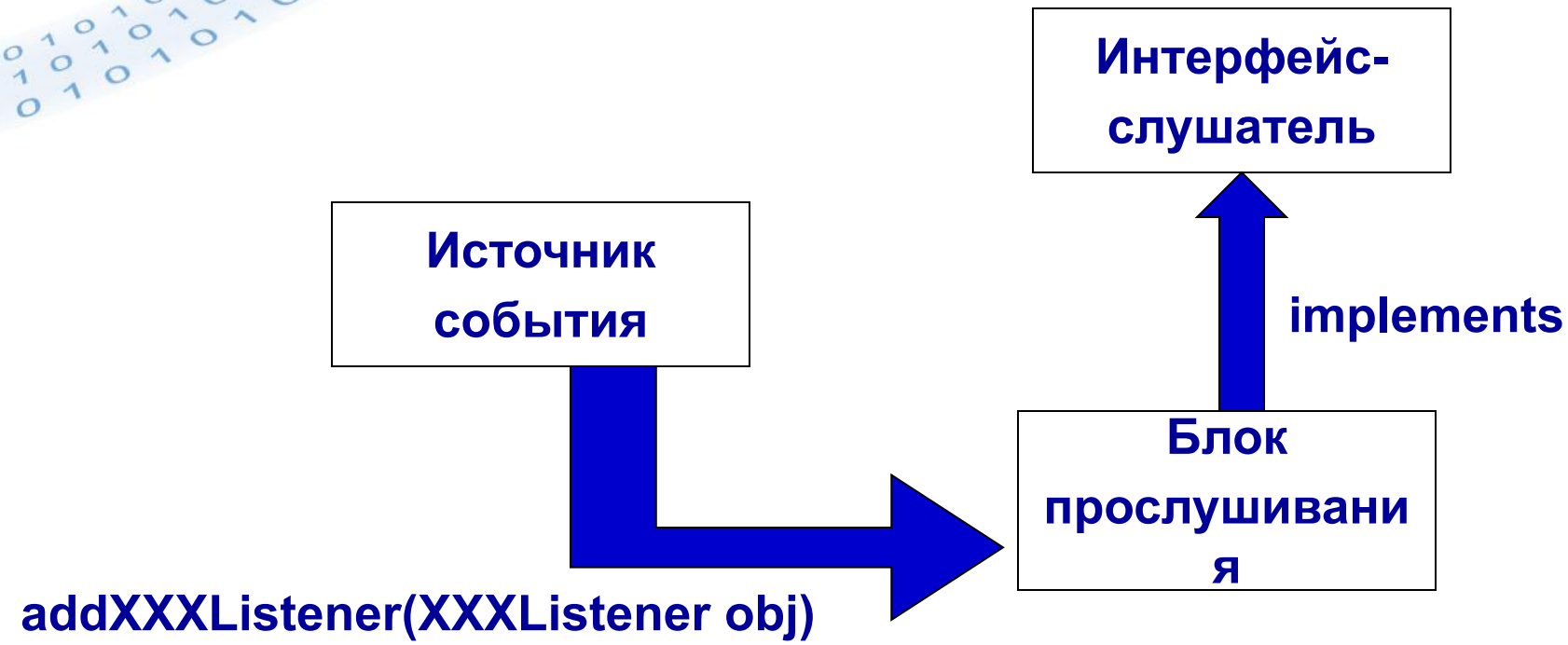
где XXX – это имя события,

obj – ссылка на экземпляр класса-обработчика события.

Методы, которые регистрируют блок прослушивания, обеспечиваются генерирующим событие источником.

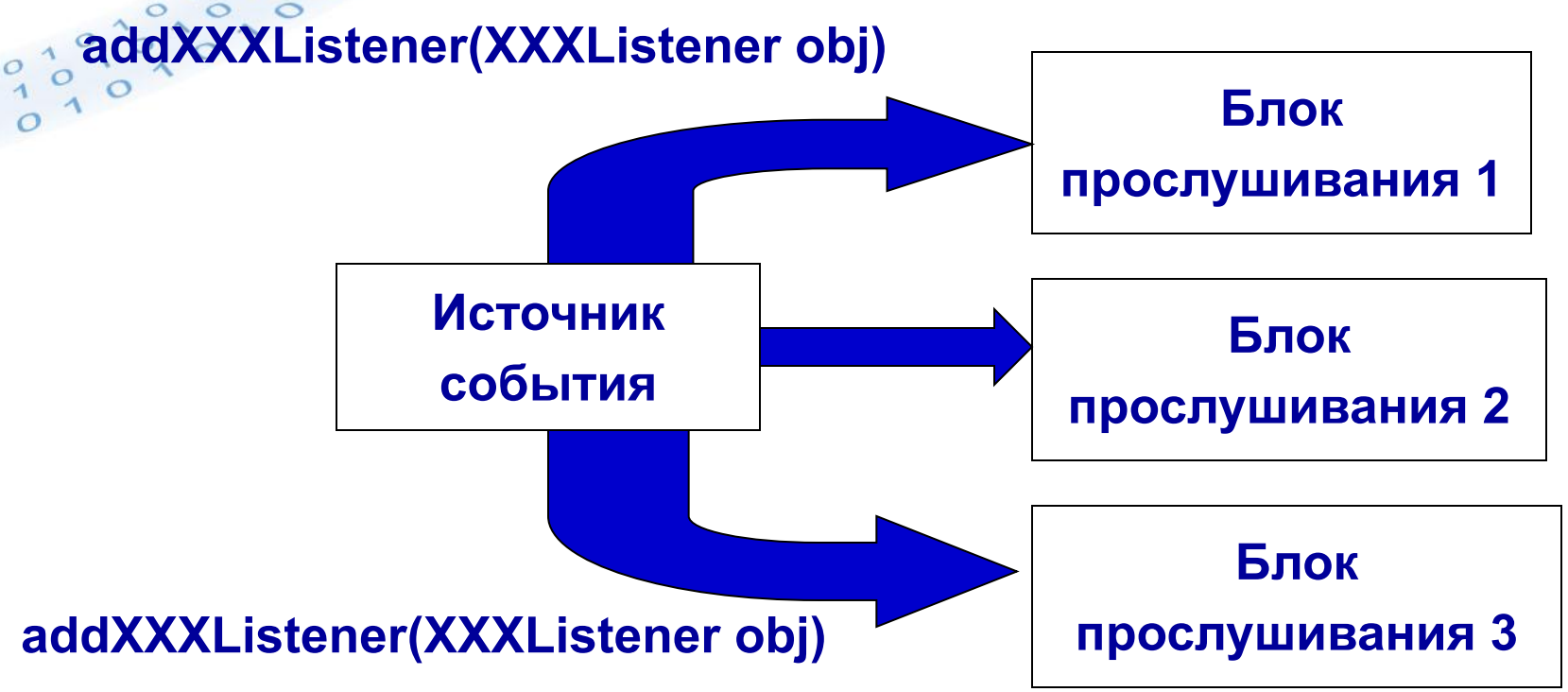
Обработка событий

Блоки прослушивания: унивещание



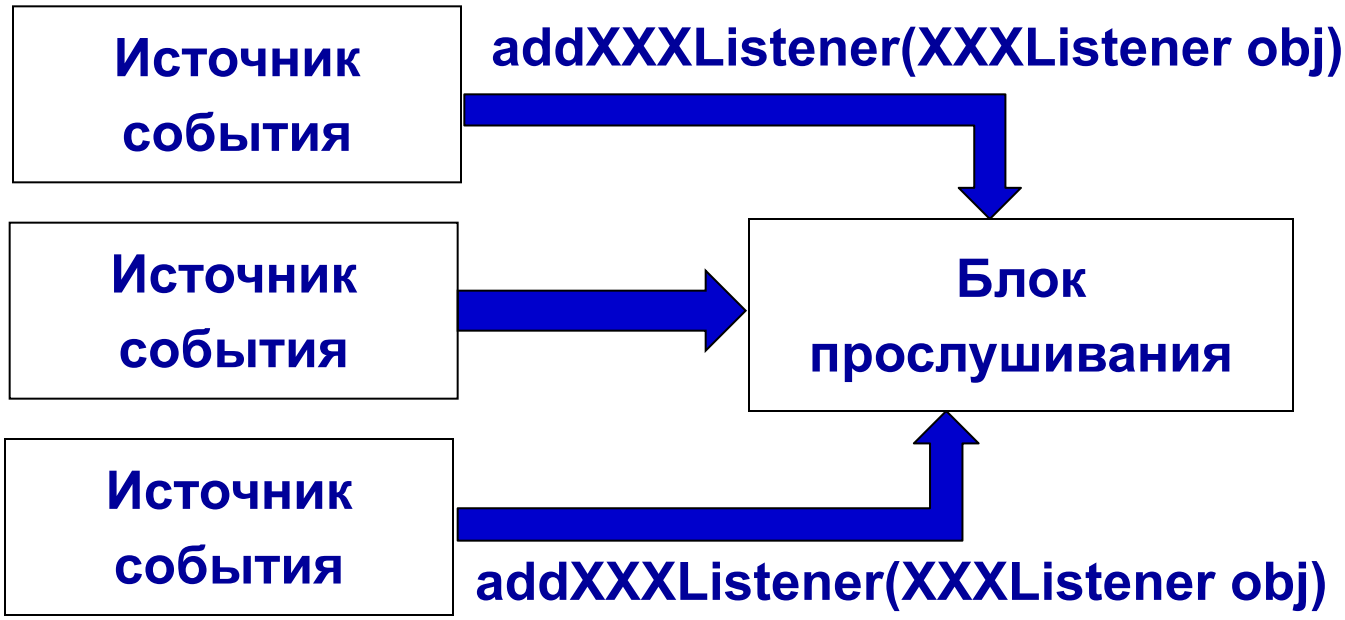
Обработка событий

Блоки прослушивания: мультивещание



Обработка событий

Блоки прослушивания: несколько источников



Обработка событий

С помощью отдельного класса

```
class TextMove implements ActionListener
{
 private JTextField jtf;
 TextMove(JTextField jtf)
        {  this.jtf = jtf; }
 public void actionPerformed(ActionEvent obj)
        { jtf.setText("Button pressed"); }
}
```

Обработка событий

С помощью отдельного класса

```
class MyNotebook extends JFrame
{public static void main(String args[ ])
  {JFrame jf = new JFrame(“Обработка события ActionEvent");
   Container cp = jf.getContentPane();
   jf.setLayout(new FlowLayout());
   JButton jb = new JButton("OK"); JTextField jtf = new JTextField(20);
   cp.add(jb);cp.add(jtf);
   jb.addActionListener(new TextMove(jtf));
   jf.setVisible(true);  }
}
```

Обработка событий

С помощью отдельного класса

Обработка событий

Внутри класса-источника

```
class MyNotebook extends JFrame implements ActionListener
{private JTextField jtf;
 public static void main(String args[ ])
  {JFrame jf = new JFrame(“Обработка события ActionEvent");
   Container cp = jf.getContentPane();
   jf.setLayout(new FlowLayout());
   JButton jb = new JButton("OK");
   JTextField jtf = new JTextField(20);
```

Обработка событий

Внутри класса-источника

```
 cp.add(jb);
 cp.add(jtf);
 jb.addActionListener(this);
 jf.setVisible(true);
 }
public void actionPerformed(ActionEvent obj)
 {
      jtf.setText("Button pressed");
 }
}
```

Обработка событий

С помощью вложенного класса

```
class MyNotebook extends JFrame
{private JTextField jtf;
 public static void main(String args[ ])
  {JFrame jf = new JFrame(“Обработка события ActionEvent");
   Container cp = jf.getContentPane();
   jf.setLayout(new FlowLayout());
   JButton jb = new JButton("OK");
   JTextField jtf = new JTextField(20);
```

Обработка событий

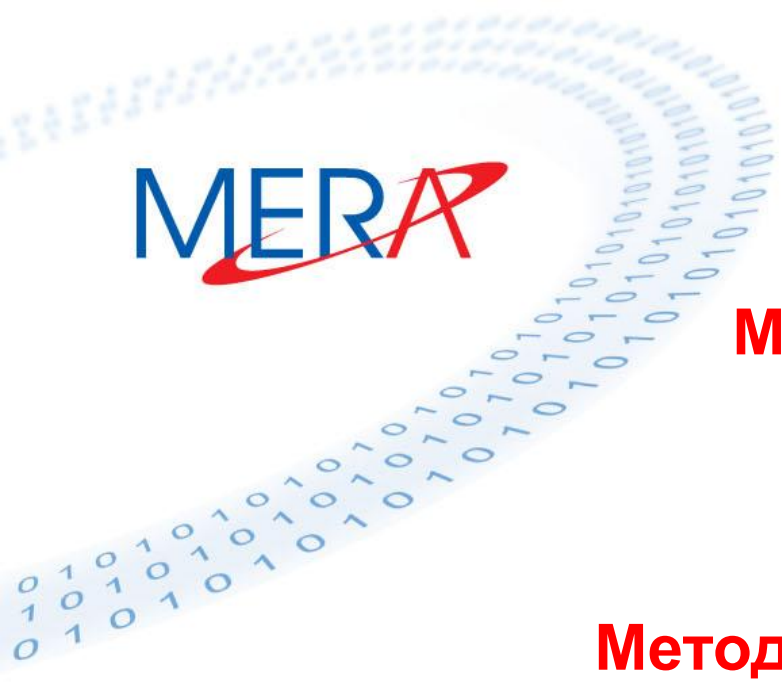
С помощью вложенного класса

```
 cp.add(jb);
 cp.add(jtf);
 jb.addActionListener(new TextMove());
 jf.setVisible(true);
}
class TextMove implements ActionListener
{public void actionPerformed(ActionEvent obj)
  { jtf.setText("Button pressed");  }
}
```

Обработка событий

С помощью анонимного вложенного класса

```
jb.addActionListener(new ActionListener() {
   public void actionPerformed(ActionEvent obj)
      { jtf.setText("Button pressed");  }
   } );
```



Обработка событий

Методы интерфейса ActionListener

void actionPerformed(ActionEvent obj)

Методы интерфейса AdjustmentListener

void adjustmentValueChanged(AdjustmentEvent obj)

Обработка событий

Методы интерфейса ComponentListener

void ComponentResized(ComponentEvent obj)
void ComponentMoved(ComponentEvent obj)
void ComponentShown(ComponentEvent obj)
void ComponentHidden(ComponentEvent obj)

Методы интерфейса ContainerListener

void componentAdded(ContainerEvent obj)
void componentRemoved(ContainerEvent obj)

Обработка событий

Методы интерфейса FocusListener

```
void focusGained(FocusEvent obj)
void focusLost(FocusEvent obj)
```

Методы интерфейса ItemListener

```
void ItemStateChanged(ItemEvent obj)
```

Методы интерфейса KeyListener

```
public void keyTyped(KeyEvent obj);
public void keyPressed(KeyEvent obj);
public void keyReleased(KeyEvent obj);
```

Обработка событий

Методы интерфейса MouseListener

```
public void mouseClicked(MouseEvent obj);
public void mousePressed(MouseEvent obj);
public void mouseReleased(MouseEvent obj);
public void mouseEntered(MouseEvent obj);
public void mouseExited(MouseEvent obj);
```

Методы интерфейса MouseMotionListener

```
public void mouseDragged(MouseEvent obj);
public void mouseMoved(MouseEvent obj);
```

Обработка событий

Методы интерфейса TextListener

```
void textValueChanged(TextEvent obj)
```

Методы интерфейса WindowListener

```
public void windowOpened(WindowEvent obj);
public void windowClosed(WindowEvent obj);
public void windowClosing(WindowEvent obj);
public void windowActivated(WindowEvent obj);
public void windowDeactivated(WindowEvent obj);
public void windowIconified(WindowEvent obj);
public void windowDeiconified(WindowEvent obj);
```

Обработка событий

Классы-адаптеры

Для реализации интерфейса-слушателя необходимо определить все его методы. Если необходимо обрабатывать в программе только некоторые из событий, обработка которых предусматривается конкретным интерфейсом-слушателем, а все прочие игнорировать, то можно воспользоваться специальным классами, которые называются классами-адаптерами. Классы-адаптеры обеспечивают пустую реализацию всех методов интерфейса, что дает возможность наследовать класс-обработчик события от класса-адаптера и реализовывать в нем не все методы интерфейса, а лишь те которые необходимы.

Обработка событий

Классы-адаптеры

Список классов-адаптеров:

- **ComponentAdapter**
- **ContainerAdapter**
- **FocusAdapter**
- **KeyAdapter**
- **MouseAdapter**
- **MouseMotionAdapter**
- **WindowAdapter**