

# ВИЗУАЛНО ПРОГРАМИРАНЕ

ЛЕКЦИОНЕН КУРС “ПРОГРАМИРАНЕ НА JAVA”



# СТРУКТУРА НА ЛЕКЦИЯТА

- Въведение
- Свойства
- Събития
- JavaBeans
- Примери

# ВЪВЕДЕНИЕ

- Java изключително ценен за създаване на годин за многократно използване код
  - Композиция
  - Наследяване
- Наследяването и полиморфизмът представляват съществена част от ООП
- Модулно сглобяване
  - Бихме искали, когато „сглобяваме“ едно приложение да можем да оперираме като електронните инженери - разполагане чипове върху една платка
- Визуалното програмиране
  - Visual Basic
  - Delphi (главно вдъхновение за дизайна на JavaBeans)
- С инструментите на визуалното програмиране компонентите на потребителския интерфейс се представят визуално по точния начин, както ще изглеждат в работещата програма
  - Така, част от процеса на визуалното програмиране представлява извличане и от палитрата и поставяне върху формата
  - Инструментите за изграждане на приложения създават код за създаване на компонента в работещата програма



# СВОЙСТВА

- Поставянето на компонента върху формата обикновено не е достатъчно за завършване на програмата
- Често се налага да променяме характеристиките на даден компонент
  - Напр. цвят, текст върху него, връзка към БД
- Характеристиките, които могат да се променят по време на проектиране се наричат „свойства“ (properties)
  - Характеристиките могат да се манипулират посредством инструментите за създаване на приложения
  - При създаване на едно приложение тези конфигурационни данни се записват, така че да бъдат възстановявани, когато програмата се стартира

# СЪБИТИЯ

- Обектите имат също набор от поведения
- Поведенията на визуалните компоненти се представят частично от събития (events)
- Обикновено решаваме каква да бъде реакцията на приложението при настъпване на дадено събитие, като свързваме код към това събитие
  - Това е критично важна част
  - Инструментите за изграждане на приложения използват механизма на рефлексията за да отправят динамични запитвания към компонента и да откриват събитията и свойствата, поддържани в компонента

# ПРЕДИМСТВА

- Голямата част от работата се извършва от инструментите за изграждане на приложения
- Можем да се концентрираме върху това, как трябва да изглежда програмата и какво трябва да прави
  - Детайловете на свързването се поемат от съответния инструмент
- Ускорява се значително процесът за изграждане на едно приложение



# JAVABEANS

- В Java визуалните компоненти (JavaBeans) е блок от код, обикновено вграден в клас
  - Т.е. класове
  - Съществено е инструментите за изграждане на приложения да откриват свойствата и събитията на тези компоненти
- За да се създаде един бийн не се налага да пишем външен код или да използваме специални разширения на езика
  - Единственото нещо, което трябва да направим, е да променим начина, по който именуваме методите
  - Името на метода е това, което указва на инструмента за изграждане на приложение дали това е свойство, събитие или обикновен метод

# ИМЕНУВАНЕ

- Конвенцията за именуване е следната:
  - За дадено свойство xxx обикновено се създават два метода:
    - `getXxx`, `setXxx`
    - Първата буква след `get` и `set` автоматично се преобразува в малка буква за да получим свойството
    - Полученият от метода `get` тип е същият като типа на аргумента на метода `set`
  - За свойство `Boolean` можем да използваме `is` вместо `get`
  - Обикновените методи на един бийн не съответстват на тази конвенция за именуване, но те са `public`
  - За събитията се използва подхода на Swing „слушател“



# ПРИМЕР

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

1 class Spots {}

```
public class Frog {
```

2

```
    private int jumps;
    private Color color;
    private Spots spots;
    private boolean jmptr;
    public int getJumps() { return jumps; }
    public void setJumps(int newJumps) {
        jumps = newJumps;
    }
    public Color getColor() { return color; }
    public void setColor(Color newColor) {
        color = newColor;
    }
    public Spots getSpots() { return spots; }
    public void setSpots(Spots newSpots) {
        spots = newSpots;
    }
    public boolean isJumper() { return jmptr; }
    public void setJumper(boolean j) {
        jmptr = j;
    }
}
```

```
public void addActionListener(ActionListener l) {
    // ...
}
```

Бийнът е просто един клас – обикновено всички полета са private

Съгласно конвенцията за именуване свойства са: jumps, color, spots и jmptr (за отбелязване е промяната на голяма буква в малка за първата буква в името на свойството)

```
    // ...
}
// Един "обикновен" public метод
public void croak() {
    System.out.println("Ribbet!");
}
}
```

3

# ПРИМЕР

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

1 class Spots {}

```
public class Frog {
    private int jumps;
    private Color color;
    private Spots spots;
    private boolean jmp;
    public int getJumps() { return jumps; }
    public void setJumps(int newJumps) {
        jumps = newJumps;
    }
    public Color getColor() { return color; }
    public void setColor(Color newColor) {
        color = newColor;
    }
    public Spots getSpots() { return spots; }
    public void setSpots(Spots newSpots) {
        spots = newSpots;
    }
    public boolean isJumper() { return jmp; }
    public void setJumper(boolean j) {
        jmp = j;
    }
}
```

```
public void addActionListener(ActionListener l) {
    // ...
}
public void removeActionListener(ActionListener l) {
    // ...
}
public void addKeyListener(ActionListener l) {
    // ...
}
public void removeKeyListener(ActionListener l) {
    // ...
}
// Един "обикновен" public метод
public void croak() {
    System.out.println("Ribbet!");
}
```

get() и set() методи за свойствата

- Събитията са `ActionEvent` и `KeyEvent`, базирани на именуването на "add" и "remove" за съответния слушател
- Методът `croak()` е обикновен

# ПРИМЕР

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

1 class Spots {}

```
public class Frog {
    private int jumps;
    private Color color;
```

3

Събитията са `ActionEvent` и `KeyEvent`, базирани на именуването на "add" и "remove" за съответния слушател

```
public Color getColor() { return color; }
public void setColor(Color newColor) {
    color = newColor;
}
public Spots getSpots() { return spots; }
public void setSpots(Spots newSpots) {
    spots = newSpots;
}
public boolean isJumper() { return jmpr; }
public void setJumper(boolean j) {
    jmpr = j;
}
```

3

```
public void addActionListener(ActionListener l) {
    // ...
}
public void removeActionListener(ActionListener l) {
    // ...
}
public void addKeyListener(ActionListener l) {
    // ...
}
public void removeKeyListener(ActionListener l) {
    // ...
}
// Един "обикновен" public метод
public void croak() {
    System.out.println("Ribbet!");
}
}
```

4

Методът `croak()` е обикновен



# ИЗВЛИЧАНЕ НА BEANINFO

- Една от най-важните части на схемата на Bean се осъществява, когато извлечем един Bean от палитрата и поставим във формата
- Инструментът за изграждане на приложения трябва да може:
  - Да създава Bean
  - След това, без достъп до първичния код на този Bean да извлича цялата необходима информация за създаване на схемата за свойствата и обработващите програми за събитията

# РЕШЕНИЕ

- Част от решението е ясно
  - Java механизъмът за рефлексия, който позволява откриване на всички методи на даден анонимен клас
  - Това е перфектно решаване на проблема, без да се изискват нови ключови думи и разширения на езика
    - Както е при други езици за визуално програмиране
  - Всъщност, една от основните причини за добавяне на рефлексия в Java е осигуряване за работа с JavaBeans
- Създателите на средства за изграждане на приложения трябва да използват рефлексията

# INTROSPECTOR

- Създателите на Java искаха да направят един стандартен инструмент
- Не само да направят Beans по-прости за използване
- Този инструмент е класът Introspector
  - Най-същественият метод в него е `getBeanInfo`
  - Към този метод подаваме Class референция
  - Методът връща един обект `BeanInfo`, доставящ информация за свойства, методи и събития



# ИЗПОЛЗВАНЕ НА СРЕДСТВАТА

- С помощта на стандартни инструменти, поддържащи визуално програмиране, изграждането на графични потребителски интерфейс е сравнително просто
- Подход:
  - Провличаме необходимите ни бийнове върху формата
  - Конфигуриране на свойствата им
  - Създаване на обработващи програми за интересуващите ни събития

# ПРИМЕР: ВИЗУАЛИЗИРАНЕ ИНФОРМАЦИЯ ЗА ДАДЕН BEAN

```
import java.beans.*;
import java.lang.reflect.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.beans.*;
import com.bruceeckel.swing.*;

public class BeanDumper extends JApplet {
    JTextField query = new JTextField(20);
    JTextArea results = new JTextArea();
    public void prt(String s) {
        results.append(s + "\n");
    }
    public void dump(Class bean) {
        results.setText("");
        BeanInfo bi = null;
        try {
            bi = Introspector.getBeanInfo(bean, java.lang.Object.class);
        } catch (IntrospectionException e) {
            prt("Couldn't introspect " + bean.getName());
            return;
        }
    }
}
```

- Методът `dump()` извършва основната работа
- Създава `BeanInfo` обект – с помощта на `Introspector.getBeanInfo`

```

PropertyDescriptor[] properties = bi.getPropertyDescriptors();
for(int i = 0; i < properties.length; i++) {
    Class p = properties[i].getPropertyType();
    prt("Property type:\n " + p.getName() + "Property name:\n " + properties[i].getName());
    Method readMethod = properties[i].getReadMethod();
    if(readMethod != null) prt("Read method:\n " + readMethod);
    Method writeMethod = properties[i].getWriteMethod();
    if(writeMethod != null) prt("Write method:\n " + writeMethod); prt("=====");
}

prt("Public methods:");
MethodDescriptor[] methods = bi.getMethodDescriptors();
for(int i = 0; i < methods.length; i++) prt(methods[i].getMethod().toString()); prt("=====");

prt("Event support:");
EventSetDescriptor[] events = bi.getEventSetDescriptors();
for(int i = 0; i < events.length; i++) {
    prt("Listener type:\n " + events[i].getListenerType().getName());
    Method[] lm = events[i].getListenerMethods();
    prt(methods[i].getMethod().toString());
    for(int j = 0; j < lm.length; j++) prt("Listener Method: " + lm[j].toString());
    MethodDescriptor[] lmd = events[i].getListenerMethodDescriptors();
    for(int j = 0; j < lmd.length; j++) prt("Listener Method: " + lmd[j].toString());
    Method addListener = events[i].getAddListenerMethod();
    prt("Add Listener Method:\n " + addListener);
    Method removeListener = events[i].getRemoveListenerMethod();
    prt("Remove Listener Method:\n " + removeListener); prt("-----");
}
}

```

- Извиква методите на BeanInfo, чрез които получава информация за свойства, методи и събития
- Напр., за свойствата – генерира масив от PropertyDescriptor, всеки от които съдържа тип, име, методи за четене и за писане
- Аналогично за публичните методи и събитията



```

class Dumper implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        String name = query.getText();
        Class c = null;
        try {
            c = Class.forName(name);
        } catch(ClassNotFoundException ex) {
            results.setText("Couldn't find " + name);
            return;
        }
        dump(c);
    }
}

public void init() {
    Container cp = getContentPane();
    JPanel p = new JPanel();
    p.setLayout(new FlowLayout());
    p.add(new JLabel("Qualified bean name:"));
    p.add(query);
    cp.add(BorderLayout.NORTH, p);
    cp.add(new JScrollPane(results));
    Dumper dmpr = new Dumper();
    query.addActionListener(dmpr);
    query.setText("Frog");
    dmpr.actionPerformed(new ActionEvent(dmpr, 0, ""));
}

public static void main(String[] args) {
    Console.run(new BeanDumper(), 600, 500);
}
}

```

5

Цялата информация се визуализира в един панел


6

В примера се извежда информация за бийна "Frog"

# ПРИМЕР

1

Какъв резултат?

**BeanDumper**

Qualified bean name: Frog

Property type:  
java.awt.ColorProperty name:  
color

Read method:  
public java.awt.Color Frog.getColor()

Read method:  
public void Frog.setColor(java.awt.Color)

=====

Property type:  
booleanProperty name:  
jumper

Read method:  
public boolean Frog.isJumper()

Read method:  
public void Frog.setJumper(boolean)

=====


Property type:  
intProperty name:  
jumps

Read method:  
public int Frog.getJumps()

Read method:  
public void Frog.setJumps(int)

=====

Property type:  
SpotsProperty name:

**BeanDumper**

Qualified bean name: Frog

=====

Public methods:

public void Frog.setJumper(boolean)

public boolean Frog.isJumper()

public void Frog.croak()

public void Frog.removeKeyListener(java.awt.event.ActionListener)

public void Frog.addActionListener(java.awt.event.ActionListener)

public void Frog.removeActionListener(java.awt.event.ActionListener)

public void Frog.addKeyListener(java.awt.event.ActionListener)

public void Frog.setSpots(Spots)

public Spots Frog.getSpots()

public int Frog.getJumps()

public java.awt.Color Frog.getColor()

public void Frog.setColor(java.awt.Color)

public void Frog.setJumps(int)

=====


Event support:

Listener type:  
java.awt.event.ActionListener

Listener method:  
actionPerformed

Method descriptor:  
public abstract void java.awt.event.ActionListener.actionPerformed(java.awt.event.ActionEvent)

Add Listener Method:  
public void Frog.addActionListener(java.awt.event.ActionListener)

**Distributed**  
@learning center

БЛАГОДАРЯ ЗА ВНИМАНИЕТО!

КРАЙ “ВИЗУАЛНО ПРОГРАМИРАНЕ”

