

DOGAĐAJI

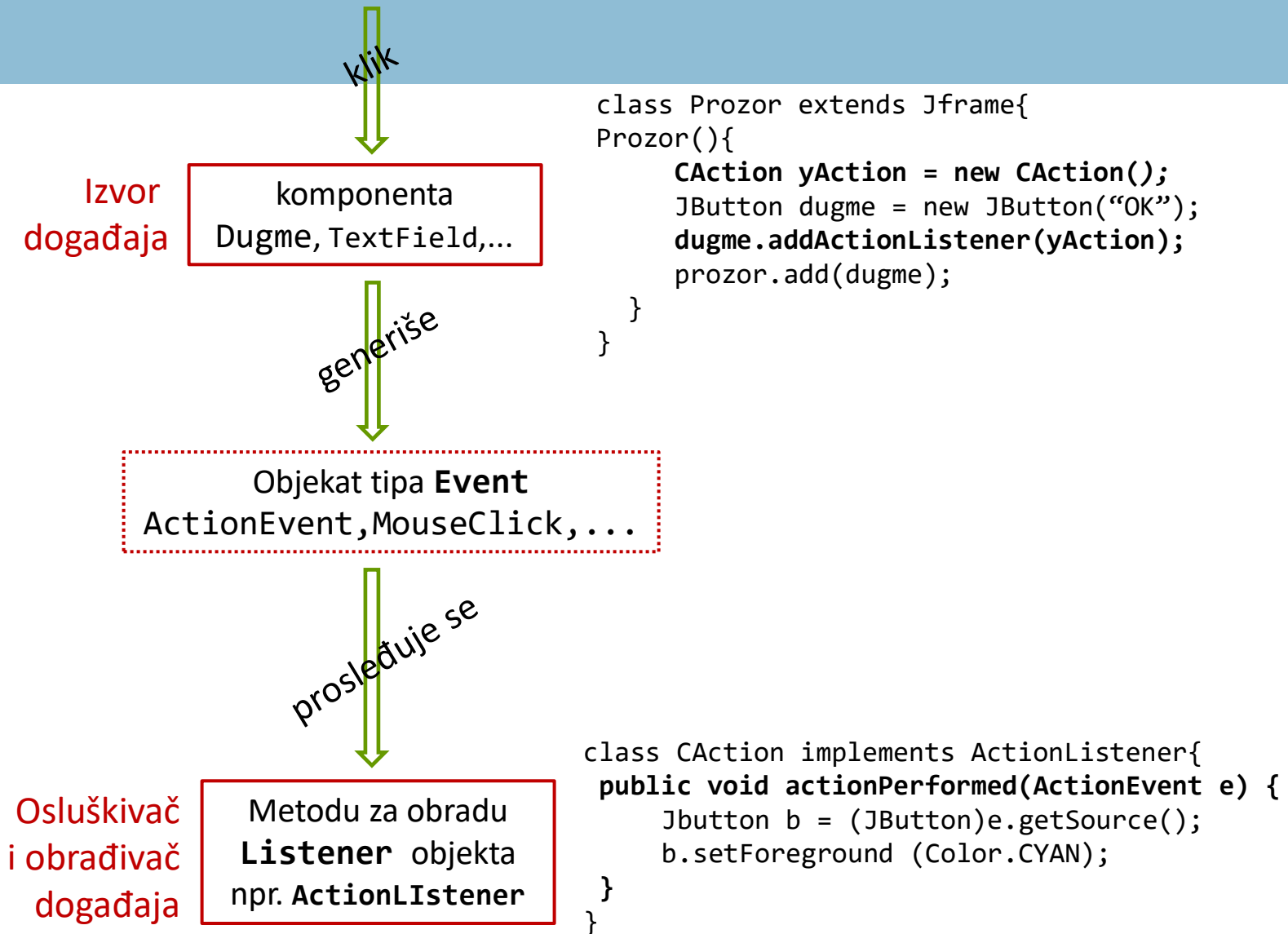
2016/17

PRIRODNO-MATEMATIČKI FAKULTET, UNIVERZITET U KRAGUJEVCU

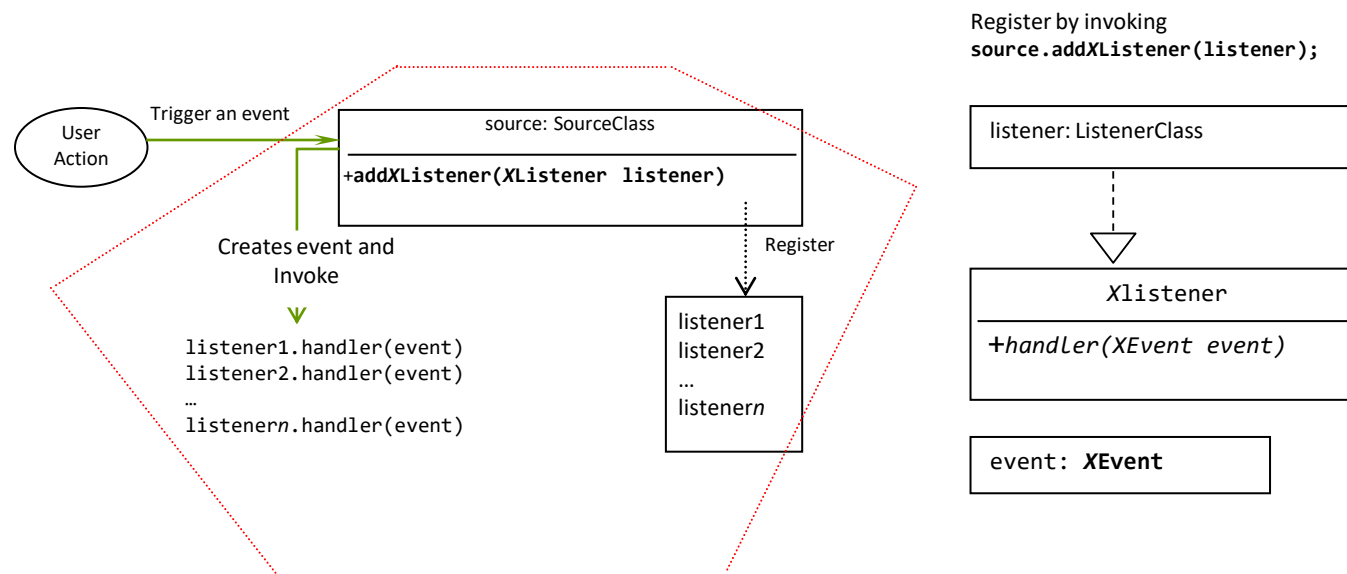
EVENT-DRIVEN PROGRAMMING

- event-driven programming – programiranje vođeno događajima
- kod se izvršava nakon aktivacija događaja
- Događaj se može shvatiti kao signal (određenog tipa) koji se šalje programu kao informacija o tome da se nešto desilo.
- Događaj se generiše nakon akcije koja nije desila od strane same aplikacije, kao što su akcija korisnika (pomeranje miša, klik mišem i sl) ili signal poslat od strane OSa.
- Java događaje
 - **generišu izvori(sources) događaja**
 - a **obrađuju objekti klase osluškivača(listeners) događaja**
 - Ako aplikacija odgovara na neku vrstu događaja onda se kodovi koji se izvršavaju nalaze u osluškivačima.
 - Za različite vrste događaja su definisani različiti tipovi osluškivača.

UČESNICI

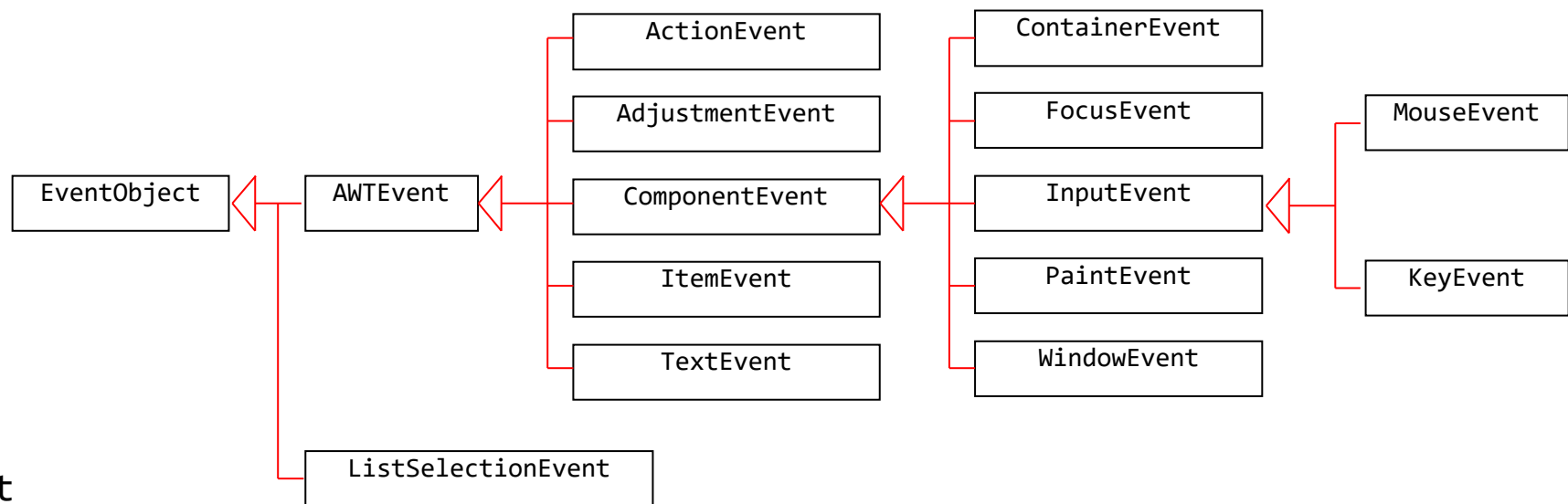


SISTEM DELEGIRANJA



- **Izvori događaja (npr. dugme)** generišu objekat tipa događaj i šalju osluškivačima
- Svaki izvor događaja održava/ima sopstvenu listu osluškivača, za svaku vrstu događaja posebnu listu.
- Da bi se neki osluškivač našao u listi nekog izvora mora se **registrovati** kod izvora.
- Ovakav model obrade događaja se naziva **delegiranje** ili **prosleđivanje**:
 - pravo obrade događaja se delegira **svakom objektu** koji implementira interfejs odgovarajućeg osluškivača

KLASE DOGAĐAJA



- `java.util.EventObject`
 - koren hijerarhije
 - metod `Object getSource()`;
- `java.awt.AWTEvent`
 - nadklasa svih AWT događaja koji koriste model prosleđivanja
 - metod `int getID()` – vraća ceo broj koji opisuje vrstu događaja
- U `java.awt.event` – razne klase događaja izvedenih iz `AWTEvent`

DEFINISANJE OBRADE DOGAĐAJA

- Dva koraka u definisanju obrade
 - **definisanje osluškivača**
 - **registracija osluškivača kod izvora**
- Definisanje **klase osluškivača**
 - Klasa mora da
 - da klasa implemetira interfejs nekog osluškivača ili
 - da klasa proširuje neku klasu koja implementira interfejs osluškivača

```
public class Obrada implements ActionListener {...}
```

- mora se implementirati metod koji propisuje interfejs osluškivača

```
public void actionPerformed(ActionEvent) {/*...*/}
```

- **Registrowanje** instance klase **osluškivača** događaja kod izvora

```
komponentaIzvor.addActionListener(obradjivac);
```

GRUPE AWT DOGADJAJA

- AWT događaji se mogu podeliti u 2 grupe:
 - **događaje niskog nivoa** - reprezentuju elementarna zbivanja u sistemu prozora ili elementarne ulaze
 - **semantičke događaje** - rezultat su korisničkih akcija koje su specifične za komponente

DOGAĐAJI NISKOG NIVOVA

- **Događaji koje generišu komponente, kontejneri, fokusi i prozori**
 - događaji komponentata se generišu pri promeni pozicije, veličine i vidljivosti
 - događaji kontejnera se generišu kada se komponenta dodaje ili uklanja iz kontejnera
 - događaji fokusa se generišu kada komponenta dobija ili gubi fokus tastature (fokus tastature je sposobnost da se prihvate karakteri koji se unose preko tastature)
 - događaji prozora daju informaciju o bazičnom stanju proizvoljne vrste prozora
- Elementarni **događaji koji potiču od ulaza miša ili tastature**
 - događaji miša su podeljeni u dve grupe:
 - događaje kretanja miša i
 - klik, pritisnut/otpušten taster,
 - ušao/izašao iz prostora komponente
 - događaji kretanja miša su češći, pa da se ne bi morali uvek obrađivati definisan je poseban interfejs

SEMANTIČKI DOGAĐAJI

- Semantički događaji uključuju **događaje akcije, prilagođenja, članske i tekstualne**
 - događaje akcije generišu ekranski tasteri (pritisak), stavke menija, liste i tekst polja
 - događaji prilagođenja se generišu kada korisnik promeni vrednost klizača (scrollbar)
 - članske događaje generiše izbor jedne od stavki iz liste
 - tekstualni događaji se generišu kada se menja tekst u prostoru za tekst ili u polju za tekst

DOGAĐAJI KOJE GENERIŠU AWT KOMPONENTE

	action	adjustment	component	container	focus	item	key	mouse	mouse motion	text	window
Button	X		X		X		X	X	X		
Canvas			X		X		X	X	X		
Checkbox			X		X	X	X	X	X		
Choice			X		X	X	X	X	X		
Component			X		X		X	X	X		
Container			X	X	X		X	X	X		
Dialog			X	X	X		X	X	X		X
Frame			X	X	X		X	X	X		X
Label			X		X		X	X	X		
List	X		X		X	X	X	X	X		
Panel			X	X	X		X	X	X		
Scrollbar		X	X		X		X	X	X		
ScrollPane			X	X	X		X	X	X		
TextArea			X		X		X	X	X	X	
TextComponent			X		X		X	X	X	X	
TextField	X		X		X		X	X	X	X	
Window			X	X	X		X	X	X		X

NEKE AKTIVNOSTI KORISNIKA

User Action	Source Object	Event Type Generated
Click a button	JButton	ActionEvent
Click a check box	JCheckBox	ItemEvent, ActionEvent
Click a radio button	JRadioButton	ItemEvent, ActionEvent
Press return on a text field	JTextField	ActionEvent
Select a new item	JComboBox	ItemEvent, ActionEvent
Select an item from a List	JList	ListSelectionEvent
Window opened, closed, etc.	Window	WindowEvent
Mouse pressed, released, etc.	Any Component	MouseEvent
Key released, pressed, etc.	Any Component	KeyEvent

STANDARDNI AWT OSLUŠKIVAČI

Interfejs	Adapter	Metodi
ActionListener	<i>nema</i>	actionPerformed
AdjustmentListener	<i>nema</i>	adjustmentValueChanged
ComponentListener	ComponentAdapter	componentHidden componentMoved componentResized componentShown
ContainerListener	ContainerAdapter	componentAdded componentRemoved
FocusListener	FocusAdapter	focusGained focusLost
ItemListener	<i>nema</i>	itemStateChanged
KeyListener	KeyAdapter	keyPressed keyReleased keyTyped

PRIMER OBRADJE DOGAĐAJA KOJI POTIČU OD MIŠA

- Interfejs osluškivača miša predviđa 5 metoda:
 - `public void mouseClicked(MouseEvent d);` //pritisnuto i otpušteno dugme
 - `public void mouseEntered(MouseEvent d);` //kursor ušao u polje komponente
 - `public void mouseExited(MouseEvent d);` //kursor izašao iz polja komponente
 - `public void mousePressed(MouseEvent d);` //pritisnuto dugme
 - `public void mouseReleased(MouseEvent d);` //otpušteno dugme
 - ako se dugme otpusti na istoj poziciji gde je pritisnuto - `mouseClicked`
 - ako se dugme otpusti na različitoj poziciji - `mouseReleased`
- Interfejs osluškivača kretanja miša predviđa 2 metoda:
 - `public void mouseMoved (MouseEvent d);`
//pomeren bez pritiskanja dugmeta
 - `public void mouseDragged (MouseEvent d);`
//pomeren sa pritisnutim dugmeta

PRIMER OBRADJE DOGAĐAJA KOJI POTIČU OD MIŠA

- Klasa `MouseEvent` je iz paketa `java.awt.event` - u hijerarhiji klasa na sledećoj poziciji:
 - `java.awt.event.MouseEvent` -> `java.awt.event.InputEvent` -> `java.awt.event.ComponentEvent` -> `java.awt.AWTEvent` -> `java.util.EventObject` -> `java.lang.Object`

- Konstruktor:

```
public MouseEvent(Component source, int id, long when,  
int modifiers, int x, int y, int clickCount, boolean popupTrigger)
```

- `source`: komponenta koja je izazvala događaj
- `id`: tip događaja (npr. `MOUSE_CLICKED`, `MOUSE_DRAGGED`,...)
- `when`: timestamp trenutka kada se događaj desio
- `modifiers`: modifikatori koji određuju da li je pritisnut `<ALT>`, `<SHIFT>`, `<MOUSE_BUTTONx>`
- `x,y`: koordinate tačke gde je se nalazio pointer pri događaju
- `clickCount`: broj "klikova" kojima je izazvan događaj
- `popupTrigger`: informacija da li je događaj izazvao pojavu pop-up menija

PRIMER OBRADJE DOGAĐAJA KOJI POTIČU OD MIŠA

```
public class OsluskivacMisa extends JFrame implements MouseListener{
    private Label t=new Label("Cekasena dogadjaj od misa...");
    public OsluskivacMisa(){
        super("Osluskivac misa"); setSize(300,100);
        t.setBackground(Color.cyan); add(t);
        addMouseListener(this); setVisible(true);
    }
    public void mouseClicked(MouseEvent d){ t.setText("Dog: clicked"); }
    public void mouseEntered(MouseEvent d){ t.setText("Dog: entered"); }
    public void mouseExited(MouseEvent d){ t.setText("Dog: exited"); }
    public void mousePressed(MouseEvent d){ t.setText("Dog: pressed"); }
    public void mouseReleased(MouseEvent d){ t.setText("Dog: released"); }
    public static void main(String[]args){
        OsluskivacMisa d=new OsluskivacMisa();
        d.setLayout(new FlowLayout());
        d.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```