The SportingEvent Applet

- We are going to modify the Event program from previous tutorials to use GUI programming.
- The applet consists of 4 classes: SportingEvent, Competition, Contestant, BarGraph.
- Competition, Contestant are unchanged from previous tutorials. BarGraph has been altered to use graphics.
- We also need an html file:

```html
<title> SportingEvent </title>
<hr>
<applet codebase = "Java Classes"
        code="SportingEvent.class"
        width=300 height= 400 >
</applet>
</hr>
```

Slide 11–1

---

SportingEvent I

- Starting up the applet.

Slide 11–2

---

SportingEvent II

- Adding the first contestant.

Slide 11–3
• The text fields are automatically emptied when Enter is pressed.

Slide 11–4

• Adding the second contestant.

Slide 11–5

• Pressing Result prints the bar graph.

Slide 11–6

import java.awt.*; import java.applet.Applet;

public class SportingEvent { 

    public void startLayout() { 
        private void resultAction() { 
            if (event.target == result) 
                return 
        
        public void print(Graphics g) { 
            public void paint(Graphics g, int size, int width, int height) { 
            
            
            Applet started.

Slide 11–7
Fields in SportingEvent

- We have three text fields where the user can enter data: name, country, score.

- Each text field should have a label: nameLabel, countryLabel, scoreLabel.

- We also have two buttons: enter, result.

- The competition variable has to be global so that the enterAction and resultAction methods both can get to it.

private Label nameLabel;
private Label countryLabel;
private Label scoreLabel;
private TextField name, country, score;
private Button enter, result;
private Competition competition;
private BarGraph graph;

The SportingEvent Constructor

public SportingEvent () {
    nameLabel = new Label("Name:");
    name = new TextField(20);
    countryLabel = new Label("Country:");
    country = new TextField(20);
    scoreLabel = new Label("Scores:");
    score = new TextField(20);
    enter = new Button("Enter");
    result = new Button("Result");

    add(nameLabel); add(name);
    add(countryLabel);
    add(country); add(scoreLabel);
    add(score); add(enter); add(result);

    competition = new Competition("Golf");
    graph = new BarGraph("Golf");
}

Component Layout

- The layout method gets called by the system to place the different components in their right position.

- We use a private method place to move the components.

public void layout () {
    place(nameLabel, 10, 10);
    place(name, 70, 10);
    place(countryLabel, 10, 50);
    place(country, 70, 50);
    place(scoreLabel, 10, 90);
    place(score, 70, 90);
    place(enter, 10, 130);
    place(result, 10, 130);
}

public void place (Component c, int x, int y) {
    c.resize(c.preferredSize());
    c.move(x, y);
}

Actions on Pressing Enter

private void enterAction () {
    Contestant contestant = new Contestant();
    competition = new Competition(contestant);
    new StringTokenizer().{
        int score = Integer.parseInt(score.getText());
        contestant.addScore(score);
        name.setText(" ");
        country.setText(" ");
        score.setText(" ");
    }
}
Printing Labels and Bars I

```java
private static int barWidth = 30;
private static int bottom = 100;
private static int maxHeight = 100;

private void printLabels (Graphics g, int width, int height) {
    for (int c=0; c<valueCount; c++)
        g.drawString(labels[c],
                    barWidth * (c * 2 + 1),
                    height - bottom + 30);
}
```

Printing Labels and Bars II

```java
private void printBars (Graphics g, int width, int height) {
    if (valueCount > 0) {
        int max = getMaxValue();
        for (int v=0; v < valueCount; v++) {
            int x1 = (2*v+1)*barWidth;
            int barHeight = maxHeight * values[v]/max;
            g.fillRect(x1, (height-bottom)-barHeight,
                        barWidth, barHeight);
        }
    }
}
```

Actions on Pressing Result

- When the Result button is pressed we go through all the contestants' results and add them to the bar graph.
- We then call repaint() which calls paint() (in SportingEvent) which calls print() (in BarGraph).

```java
private void resultAction() {
    for (int c=0; c<competition.getNumberOfContestants(); c++)
        graph.addValue(competition.getContestant(c).getName(),
                        competition.getContestant(c).getTotal());
    repaint();
}
```

The BarGraph Class

```java
class BarGraph {
    BarGraph (String banner) { }
    public void addValue (String label, int value) { }
    private int getMaxValue () { }

    private void printLabels (Graphics g, int width, int height) {
    }
    private void printBars (Graphics g, int width, int height) {

        public void print (Graphics g, int width, int height) {
            printBars(g, width, height);
            printLabels(g, width, height);
        }
    }
}
```