Sarajevo School of Science and Technology
CS/IS 160 Programming and Problem Solving II
May 2015

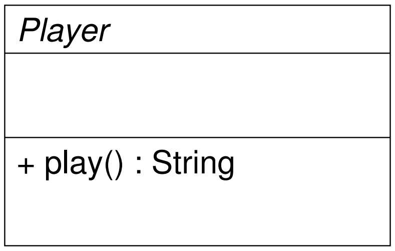
Week 11 Lab Exercises
Interfaces and Event-driven Programming

In this lab, we will explore interfaces and event-driven programming in Java.

Lab setup:

- Create a project named Week-11-Lastname-Firstname
- Use the *default* package or create appropriately named packages for your classes

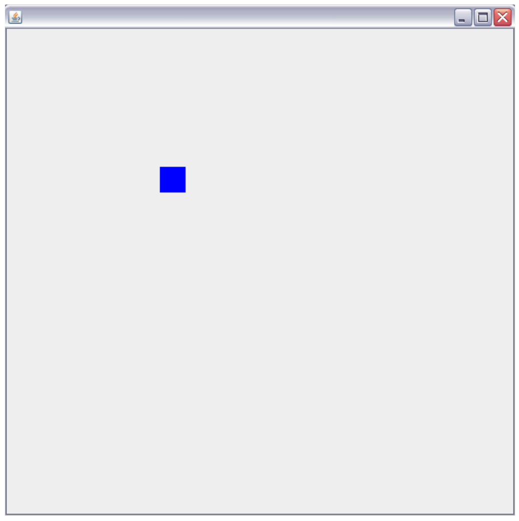
1) *(Interfaces and polymorphism)* In this problem, we will write three classes **`GoalKeeper`**, **`Defender`**, and **`Striker`**, where each class implements the **`Player`** interface shown below.



Implement the **`play`** method for each of the three classes to return appropriate strings. For example, GoalKeeper can return "Vidi me, ja branim", Defender can return "Ja oduzimam lopte", and Striker can return "Ja dajem golove."

50 points

2) Write a program that implements a simple GUI navigation using arrows on the keyboard. You will be navigating a blue square (see the image below). The square needs to keep going in the direction of the last arrow key pressed and needs to bounce off an edge when it reaches one.



Use the code provided below to help you start your implementation.

```
import java.awt.Color;
import java.awt.Graphics;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.Timer;

public class Bouncer extends JPanel implements KeyListener, ActionListener {

      // x & y are coordinates, velx & vely determine square's movement speed
      int x = 1, y = 1, vely = 2, velx = 2;
      final int size = 30;

      // determines in which direction square moves
      boolean right, left, up, down;

      // timer instance
      private Timer animationTimer;

      public Bouncer() {
            animationTimer = new Timer(10, this);
            animationTimer.start();
      }
```

```
	@Override
	public void paintComponent(Graphics g) {
		super.paintComponent(g);
		setBackground(Color.WHITE);
		g.setColor(Color.BLUE);
		g.fillRect(x, y, size, size);
	}

	@Override
	public void actionPerformed(ActionEvent e) {
		// if going right can keep going right or bounce left
		if (right) {
			// TODO: add going right and bouncing left logic here
		}

		// TODO: finish rest of the cases here: left, up, and down
		repaint();

	}

	@Override
	public void keyPressed(KeyEvent e) {
		if (e.getKeyCode() == KeyEvent.VK_DOWN) {
			down = true;
			up = false;
			right = false;
			left = false;
		} else if (e.getKeyCode() == KeyEvent.VK_UP) {
			// TODO: add up state code here
		} else if (e.getKeyCode() == KeyEvent.VK_RIGHT) {
			// TODO: add right state code here
		} else if (e.getKeyCode() == KeyEvent.VK_LEFT) {
			// TODO: add left state code here
		}
	}

	@Override
	public void keyReleased(KeyEvent e) {}

	@Override
	public void keyTyped(KeyEvent e) {}

	public static void main(String args[]) {
		JFrame window = new JFrame();
		Bouncer b = new Bouncer();
		window.add(b);
		window.addKeyListener(b);

		window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		window.setSize(400, 400);
		window.setVisible(true);
	}

}
```

50 points