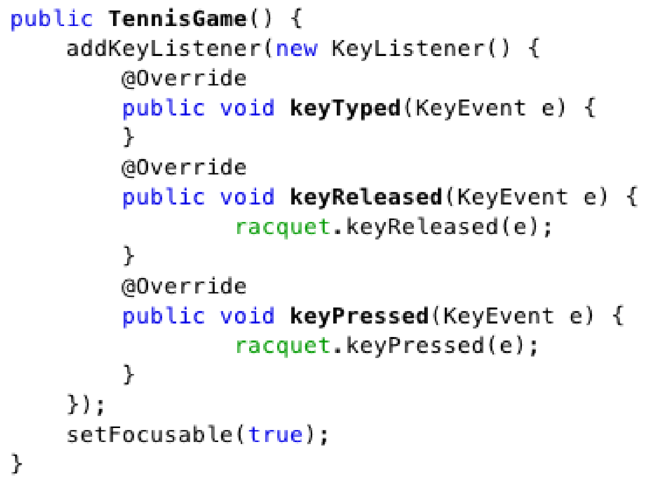
SchoolNova

IT101

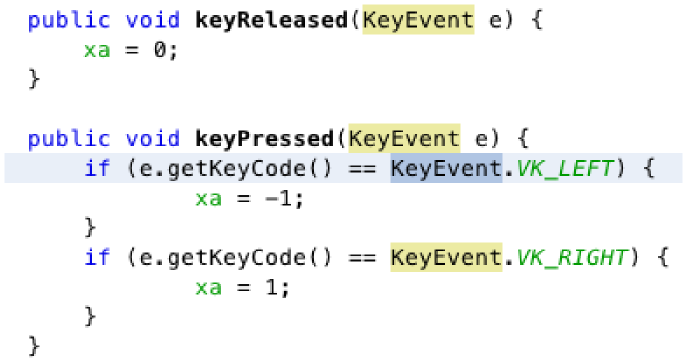
Graphical User Interface
part II

Event Handling

- In the previous lesson we created three objects: TennisGame, Ball and Racquet. Let's review the keyboard event handling.
- TennisGame subscribes to keyboard events listening, of which there are three types: keyTyped, keyReleased and keyPressed. All must be implemented.
- When keyboard event occurs, TennisGame calls the Racquet object's methods, passes the keyboard event and lets the Racquet handle it.Note that variable “xa” controls the speed of Racquet movement.
- Compile and run your game to see that you can now move the Racquet.



```java
public TennisGame() {
    addKeyListener(new KeyListener() {
        @Override
        public void keyTyped(KeyEvent e) {
        }
        @Override
        public void keyReleased(KeyEvent e) {
                racquet.keyReleased(e);
        }
        @Override
        public void keyPressed(KeyEvent e) {
                racquet.keyPressed(e);
        }
    });
    setFocusable(true);
}
```



```java
public void keyReleased(KeyEvent e) {
    xa = 0;
}

public void keyPressed(KeyEvent e) {
    if (e.getKeyCode() == KeyEvent.VK_LEFT) {
            xa = -1;
    }
    if (e.getKeyCode() == KeyEvent.VK_RIGHT) {
            xa = 1;
    }
}
```

Rectangles Collision

- To detect the collision between the ball and the racquet we will use the Java Rectangle object.
- The class java.awt.Rectangle has an “intersects” method which returns true when two rectangles occupy the same space (collide).
- Add the getBounds() method to both Ball and Racquet. In order not to repeat the width and height of the Ball and Racquet object, consider converting them to variables and using the variables instead of the hard-coded values.

```java
 public static final int DIAMETER = 30;

public Rectangle getBounds() {
    return new Rectangle(x, y, DIAMETER, DIAMETER);
}
```

```java
public static final int WIDTH = 60;
public static final int HEIGHT = 10;
public static final int Y = 330;

public Rectangle getBounds() {
    return new Rectangle(x, Y, WIDTH, HEIGHT);
}
```

- If you precede a class variable's name with “final” its value cannot be changed throughout the lifetime of the program. You can, of course, give that variable an initial value.
- What if the variables were not static?
- Add the collision handling code, then compile and run your game. The ball should now bounce off the racquet.

```java
public boolean collision() {
        return racquet.getBounds().intersects(ball.getBounds());
}
```

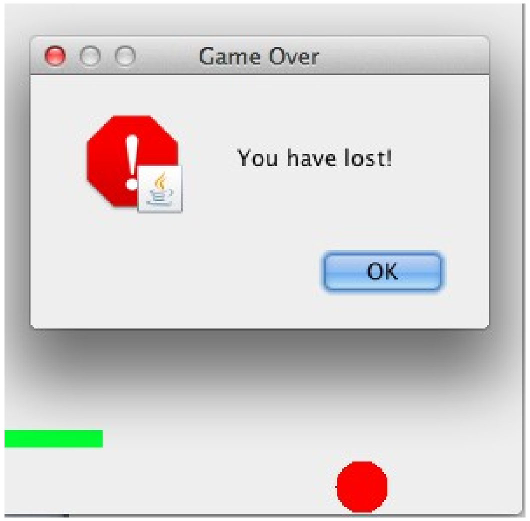
```java
if (game.collision()){
    ya = -1;
}
```

Game Over

- Use the JOptionPane object in gameOver method to display a dialog box when Racquet misses the Ball.
- Call the “gameOver” method when the Ball reaches the bottom of the canvas.
- Compile and run your code to test the Game Over scenario.

```
public void gameOver() {
    JOptionPane.showMessageDialog(this, "You have lost!", "Game Over", JOptionPane.YES_NO_OPTION);
    System.exit(ABORT);
}
```

```
if (y + ya == game.getHeight() - DIAMETER) {
    game.gameOver();
}
```

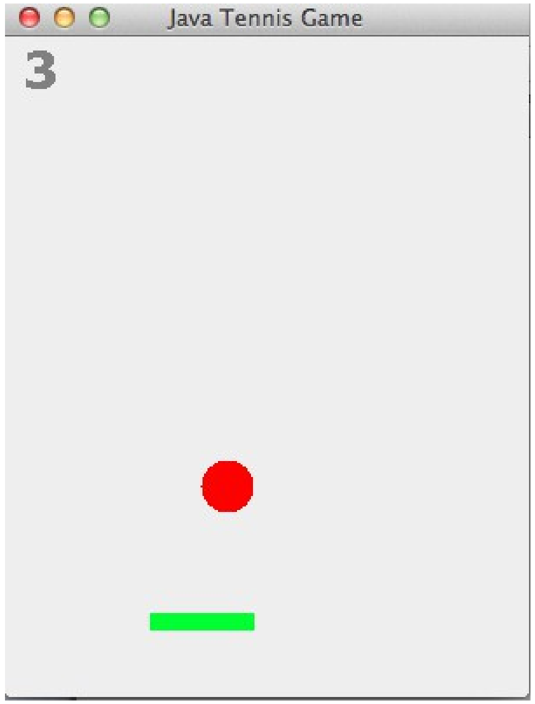


Score Counter

- Add score keeping to the game. Paint the current score as part of the game's "paint" method.
- Increment the score each time the Racquet touches the ball.
- Compile and run the game to check the score.

```java
private int score = 0;
/**
 * Return the current score
 * @return score
 */
public int getScore() {
    return score;
}
/**
 * Change the score
 * @param increment
 */
public void changeScore(int increment) {
    score += increment;
}
```

```java
g2d.setColor(Color.GRAY);
g2d.setFont(new Font("Verdana", Font.BOLD, 30));
g2d.drawString(String.valueOf(getScore()), 10, 30);
```



Homework

- Enhance the tennis game:
    1. Instead of stopping the game, decrement the score every time the Racquet misses the Ball.
    2. Increase the Ball speed with every score increase.
    3. Add another Ball, which starts from top right corner a little later than the first Ball.