```
private void doHandle() {

    opacity -= 0.01;
    lbl.opacityProperty().set(opacity);

    if (opacity <= 0) {

        stop();
        System.out.println("Animation stopped");
    }
```

```
try {

    // set title for the stage
    stage.setTitle("VBox");

    // create a VBox
    VBox vbox = new VBox();

    // create a label
    Label label = new Label("this is VBox example");

    // add label to vbox
    vbox.getChildren().add(label);

    // add buttons to VBox
    for (int i = 0; i < 10; i++)
    {
        vbox.getChildren().add(new Button("Button " + (int)(i + 1)));
    }

    // create a scene
    Scene scene = new Scene(vbox, 300, 300);

    // set the scene
    stage.setScene(scene);
```

```
Group root = new Group();
Scene s = new Scene(root, 300, 300, Color.BLACK);

final Canvas canvas = new Canvas(250,250);
GraphicsContext gc = canvas.getGraphicsContext2D();

gc.setFill(Color.BLUE);
gc.fillRect(75,75,100,100);

root.getChildren().add(canvas);
```

```
VBox root = new VBox(); // layout component which positions all its child nodes (components)
Canvas canvas = new Canvas(width*cornersize, height*cornersize );
//Canvas is an image that can be drawn on using a set of
// graphics commands provided by a GraphicsContext .
GraphicsContext graphicsContext = canvas.getGraphicsContext2D();
root.getChildren().add(canvas);

new AnimationTimer() {
    long lastTick = 0;

    public void handle(long now) {
        if (lastTick == 0) {
            lastTick = now;
            tick(graphicsContext);
            return;
        }

        if (now - lastTick > 1000000000 / speed) {
            lastTick = now;
            tick(graphicsContext);
        }
    }

}.start();
```

```
public void keyPressed(KeyEvent e) {

    int key = e.getKeyCode();

    if ((key == KeyEvent.VK_LEFT) && (!rightDirection)) {
        leftDirection = true;
        upDirection = false;
        downDirection = false;
    }

    if ((key == KeyEvent.VK_RIGHT) && (!leftDirection)) {
        rightDirection = true;
        upDirection = false;
        downDirection = false;
    }

    if ((key == KeyEvent.VK_UP) && (!downDirection)) {
        upDirection = true;
        rightDirection = false;
        leftDirection = false;
    }

    if ((key == KeyEvent.VK_DOWN) && (!upDirection)) {
        downDirection = true;
        rightDirection = false;
        leftDirection = false;
```

```
Scene scene = new Scene(root, width*cornersize, height*cornersize);

// control the snake
scene.addEventFilter(KeyEvent.KEY_PRESSED, key -> {
    if (key.getCode() == KeyCode.UP) {
        dir = Dir.up;
    }
    if (key.getCode() == KeyCode.DOWN) {
        dir= Dir.down;
    }
    if (key.getCode() == KeyCode.LEFT) {
        dir = Dir.left;
    }
    if (key.getCode() == KeyCode.RIGHT) {
        dir = Dir.right;
    }
```

```
            catch (Throwable e) {

                // print stack trace
                e.printStackTrace();
            }
```

```
catch (Exception e) {
  //prints stacktrace for this Throwable Object
  e.printStackTrace();
```

```
switch(direction) {
    case UP :
        node = new Node(headX, headY - 1);
        break;
    case RIGHT :
        node = new Node(headX + 1, headY);
        break;
    case DOWN :
        node = new Node(headX, headY + 1);
        break;
    case LEFT :
        node = new Node(headX - 1, headY);
        break;
}
```

```
switch (dir) {
    case up:
        snake.get(0).y--;
        if (snake.get(0).y < 0) {
            gameOver = true;
        }
        break;
    case down:
        snake.get(0).y++;
        if (snake.get(0).y > height) {
            gameOver = true;
        }
        break;
    case left:
        snake.get(0).x--;
        if (snake.get(0).x < 0) {
            gameOver = true;
        }
        break;
    case right:
        snake.get(0).x++;
        if (snake.get(0).x > width) {
            gameOver = true;
        }
        break;
```