Java
for Game Development

Knoxville Game Design

February 2020

Levi D. Smith

Background

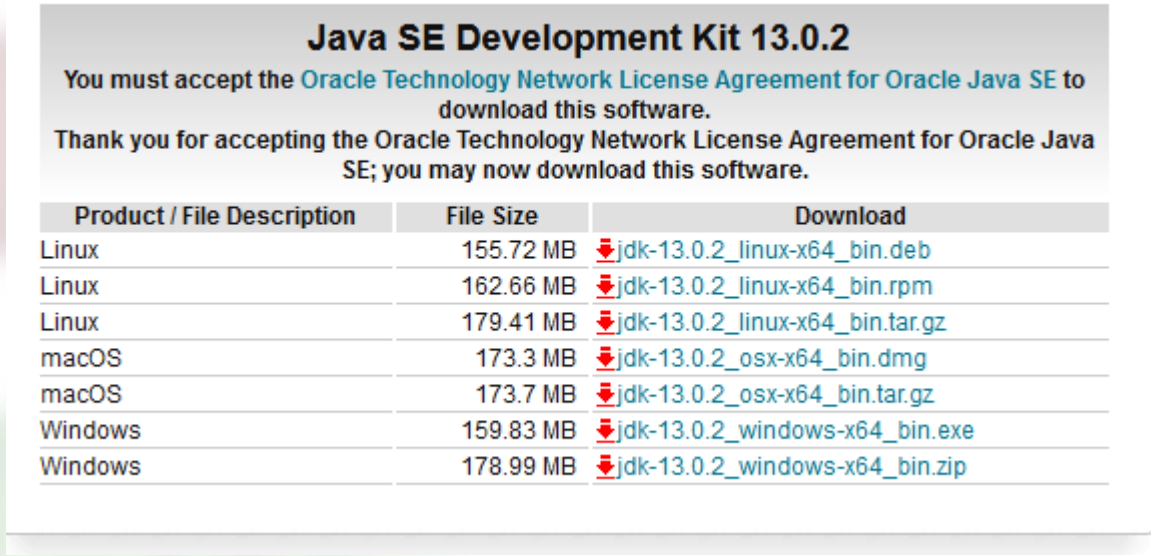
- Designed by James Gosling at Sun Microsystems
- 1995 - First version
- Object oriented
- Now owned by Oracle
- OpenJDK – open source implementation of Java
- Java is not Javascript!
  - Javascript is loosely typed, scripting language primarily used for client side web browsers; ECMAScript specification; developed by Netscape
  - Java is strongly typed, compiles code into platform independent byte code; Runs from command line using the Java Runtime Environment (also previously web “applets”)

- Pros
  - Bytecode runs on multiple platforms (Windows, Mac, Linux, etc)
  - Old bytecode still works with latest JRE (Java Runtime Environment)
  - Graphics API (AWT/Swing) built directly into the language
    - No libraries required

- Cons
  - Need to install or distribute JRE
  - Loading assets (graphics, sounds, etc) is not straight forward
  - “Slower” than platform specific executables (such as Windows EXEs)

Download and Install

- JRE – Java Runtime Environment
  - only for running Java programs
- JDK – Java Development Kit
  - for compiling Java programs
  - https://www.oracle.com/technetwork/java/javase/downloads/index.html
  - Current version is JDK 13.0.2
  - LTS – Long Term Service



Java SE Development Kit 13.0.2

You must accept the Oracle Technology Network License Agreement for Oracle Java SE to download this software.
Thank you for accepting the Oracle Technology Network License Agreement for Oracle Java SE; you may now download this software.

Product / File Description	File Size	Download
Linux	155.72 MB	jdk-13.0.2_linux-x64_bin.deb
Linux	162.66 MB	jdk-13.0.2_linux-x64_bin.rpm
Linux	179.41 MB	jdk-13.0.2_linux-x64_bin.tar.gz
macOS	173.3 MB	jdk-13.0.2_osx-x64_bin.dmg
macOS	173.7 MB	jdk-13.0.2_osx-x64_bin.tar.gz
Windows	159.83 MB	jdk-13.0.2_windows-x64_bin.exe
Windows	178.99 MB	jdk-13.0.2_windows-x64_bin.zip

Hello World

```
D:\ldsmith\presentations\java_gamedev\src>javac -version
javac 13.0.1
```

- Open command prompt (*Run* > *cmd* on Windows)
- javac -version
  - Older versions you have to add Java path to your PATH environment variable
- Class name must match file name
- javac HelloWorld.java
  - Compiles the HelloWorld.java class file
  - Will create HelloWorld.class file
  - .class files are platform independent (bytecode)
- java HelloWorld
  - Runs the HelloWorld class (must have main method)

```
class HelloWorld {
    public HelloWorld() {
        System.out.println("Hello World");
    }

    public static void main(String args[]) {
        new HelloWorld();
    }
}
```

```
D:\ldsmith\presentations\java_gamedev\src>javac *.java

D:\ldsmith\presentations\java_gamedev\src>java HelloWorld
Hello World
```

KNOXVILLE GAME DESIGN

Basics

- Entry point is *public static void main(String args[])*
- Constructor name must match the class name
- New objects (class instances) are created with the *new* keyword
- No need to delete/free objects, Garbage collection
- Methods and Variable scoping (public, private, protected)
- API / Javadocs
  - https://docs.oracle.com/javase/7/docs/api/index.html

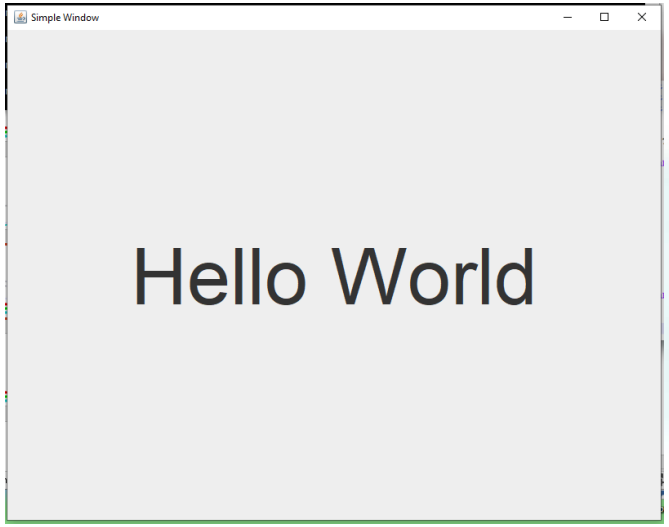
A Simple Number Guessing Game

```
D:\ldsmith\presentations\java_gamedev\src>java NumGuess
Guess the number between 1 and 100
50
Lower
Guess the number between 1 and 100
25
Lower
Guess the number between 1 and 100
12
Higher
Guess the number between 1 and 100
18
Higher
Guess the number between 1 and 100
21
Lower
Guess the number between 1 and 100
19
Correct!  The number was 19
6 total guesses
```

```java
import java.util.Scanner;

class NumGuess {

    public NumGuess() {
        int iSecretNumber;
        int iGuessNumber;
        int iGuesses;

        iSecretNumber = (int) (Math.random() * 99  + 1);
        iGuessNumber = -1;
        iGuesses = 0;

        while (iGuessNumber != iSecretNumber) {
            System.out.println("Guess the number between 1 and 100");
            Scanner s = new Scanner(System.in);
            iGuessNumber = s.nextInt();
            iGuesses++;

            if (iGuessNumber > iSecretNumber) {
                System.out.println("Lower");
            } else if (iGuessNumber < iSecretNumber) {
                System.out.println("Higher");
            } else if (iGuessNumber == iSecretNumber) {
                System.out.println("Correct!  The number was " + iSecretNumber);
                System.out.println(iGuesses + " total guesses");

            }
        }
    }

    public static void main(String args[]) {
        new NumGuess();
    }

}
```

Making a Window

- AWT - Abstract Window Toolkit
  - Java's GUI components
- Swing – Java's attempt at making platform independent GUI components




```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

class SimpleWindow {

    public SimpleWindow() {
        JFrame f = new JFrame("Simple Window");
        JLabel label = new JLabel("Hello World");
        label.setPreferredSize(new Dimension(800, 600));
        label.setHorizontalAlignment(SwingConstants.CENTER);
        label.setFont(new Font("Sans Serif", Font.PLAIN, 96));
        f.getContentPane().add(label);
        f.pack();
        f.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        f.setVisible(true);
    }

    public static void main(String args[]) {
        new SimpleWindow();
    }

}
```

KNOXVILLE GAME DESIGN

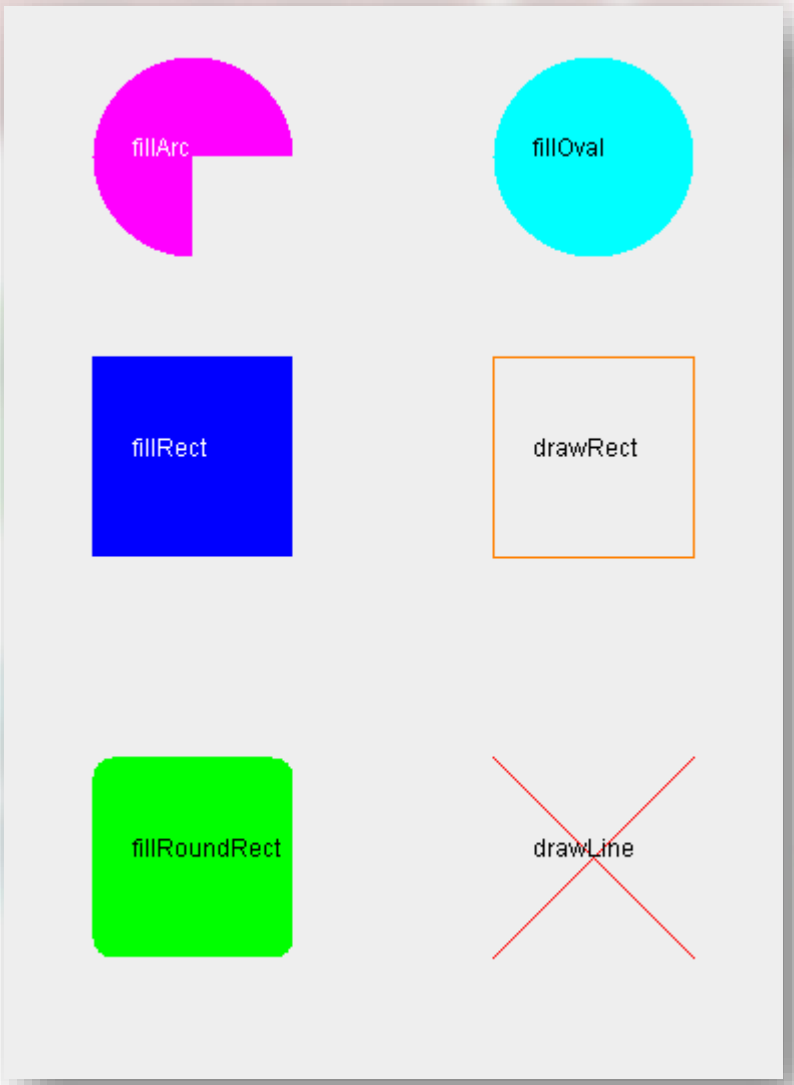
Drawing to the screen

- Make a new JComponent subclass for drawing
  - Previously Canvas
- Override paintComponent*(Graphics g)* to draw
- Coordinates are relative to the JComponent
- More complex drawing methods by casting to Graphics2D
  - rotate, scale, etc

- Rectangles / Squares
  - g.drawRect(x, y , width, height)
  - g.fillRect(x, y , width, height)
- Ovals / Circles
  - g.drawOval(x, y, width, height)
  - g.fillOval(x, y, width, height)
- Lines
  - g.drawLine(x1, y1, x2, y2)
- Text
  - g.drawString(string, x, y)
- Arcs
  - g.drawArc(x, y, width, height, start angle, stop angle)
  - g.fillArc(x, y, width, height, start angle, stop angle)
- Set color - g.setColor(Color.green)

Drawing Examples

```
public void paintComponent(Graphics g) {
    g.setColor(new Color(255, 0, 255));
    g.fillArc(200, 50, 100, 100, 0, 270);

    g.setColor(Color.cyan);
    g.fillOval(400, 50, 100, 100);

    g.setColor(Color.blue);
    g.fillRect(200, 200, 100, 100);

    g.setColor(new Color(255, 128, 0));
    g.drawRect(400, 200, 100, 100);

    g.setColor(Color.green);
    g.fillRoundRect(200, 400, 100, 100, 20, 20);

    g.setColor(Color.red);
    g.drawLine(400, 400, 500, 500);
    g.drawLine(500, 400, 400, 500);

    g.setColor(Color.white);
    g.drawString("fillArc", 220, 100);
    g.drawString("fillRect", 220, 250);

    g.setColor(Color.black);
    g.drawString("fillOval", 420, 100);
    g.drawString("drawRect", 420, 250);
    g.drawString("fillRoundRect", 220, 450);
    g.drawString("drawLine", 420, 450);
}
```



KNOXVILLE GAME DESIGN

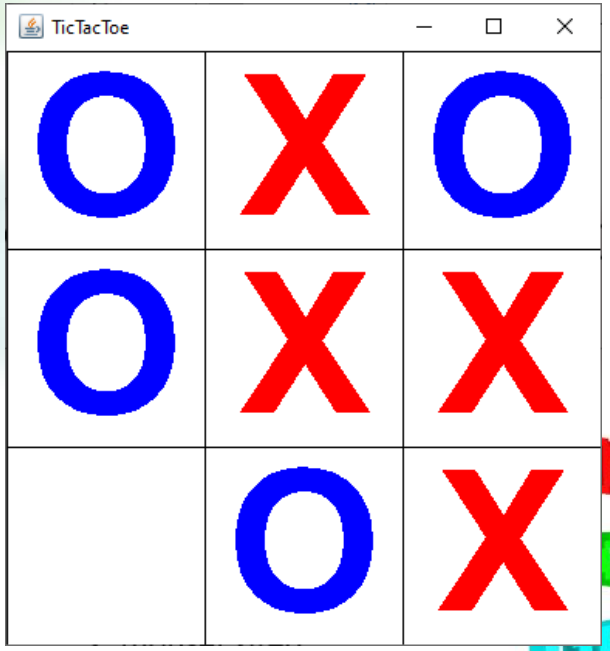
Types of Input

- Event driven
  - Mouse click
  - Key press, release
  - Button pressed, released
- Polling
  - Joystick position
  - Mouse pointer position
  - Button is pressed

Mouse Input (event driven)

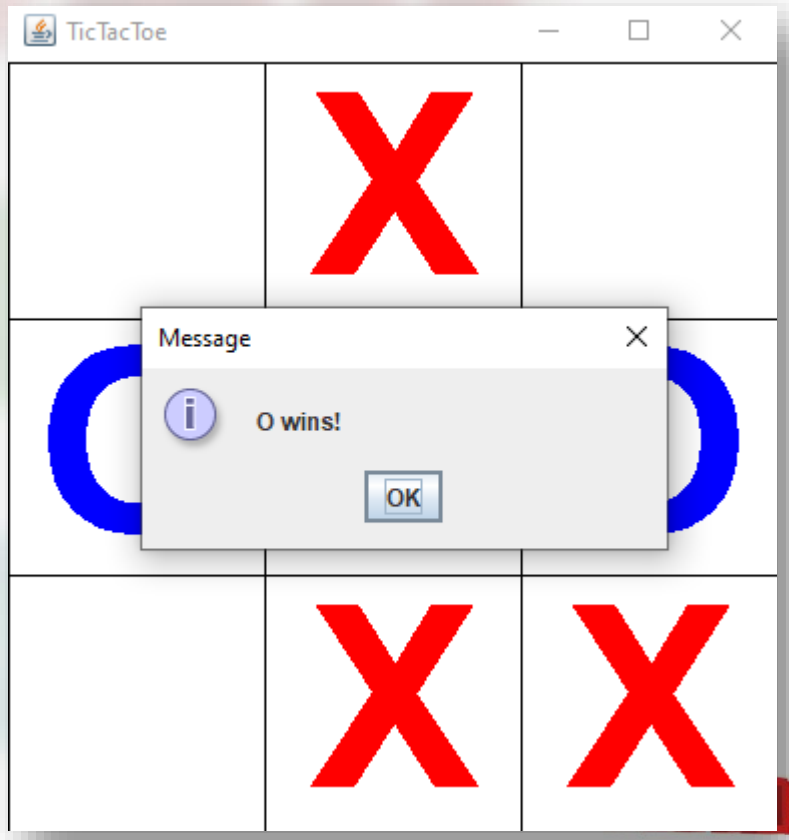
```
Mouse Pressed at: 97, 197
Mouse Pressed at: 225, 197
Mouse Pressed at: 298, 198
Mouse Pressed at: 276, 56
Mouse Pressed at: 164, 54
Mouse Pressed at: 35, 70
```

- Implement MouseListener
  - Define callback methods
    - mousePressed
    - mouseReleased
    - mouseClicked
    - mouseEntered
    - mouseExited
  - Add MouseListener to component
- Call repaint() after handling mouse input
- Get mouse click position with e.getX() and e.getY() from MouseEvent e
- Convert coordinates to cell row and column
- Implement MouseMotionListener to detect mouse movement



Dialog Box

- Use JOptionPane.showMessageDialog
  - Can pass null as the JFrame
  - Second parameter is the message



Check Win State / Game Over

- Win State
  - A row or column has all of same player mark
  - A diagonal has all of same player mark
- Game Over
  - All cells are filled with player marks

Other GUI Components

- JButton to add a button
  - Implement ActionListener and actionPerformed method to detect button presses
  - Use constant (String final) for button label
  - e.getActionCommand() in actionPerformed to check button press
  - string1.equals(string2) to compare Strings
- JPanel container to add multiple components
  - Layouts of containers – BorderLayout, GridLayout, BoxLayout

Differences from C#

	Java	**C#**
boolean (true/false) variable	boolean	bool
define subclass	extends	:
refer to super class	super	base
import libraries	import	using
get a random number	Math.random()	Random.Range()
constant	final	const
print debug text	System.out.println(“text”);	Debug.Log(“text”); //Unity

Packaging Your Game

- Multiple class files can be bundled into a JAR file (Java Archive)
- Must include Manifest file with default class property
- If JRE is installed on system, then double clicking the JAR will run the program