# 12: GUIs

June 28, 2018

GUI Applications

Hangman

Setting up the Canvas

Building the GUI

The Scratch Method

The Drawing Method

# GUI Applications

# What is a GUI?

- GUI (pronounced “gooey”) stands for graphical user interface.
- It’s what we call desktop applications that mostly use a **mouse**.
- The library for building GUI applications in Java is called **JavaFX**.

# Quiz.

- What does GUI stand for?
- What piece of computer hardware is used with a GUI?
- What is the library in Java which allows us to make GUI applications?

# Hangman

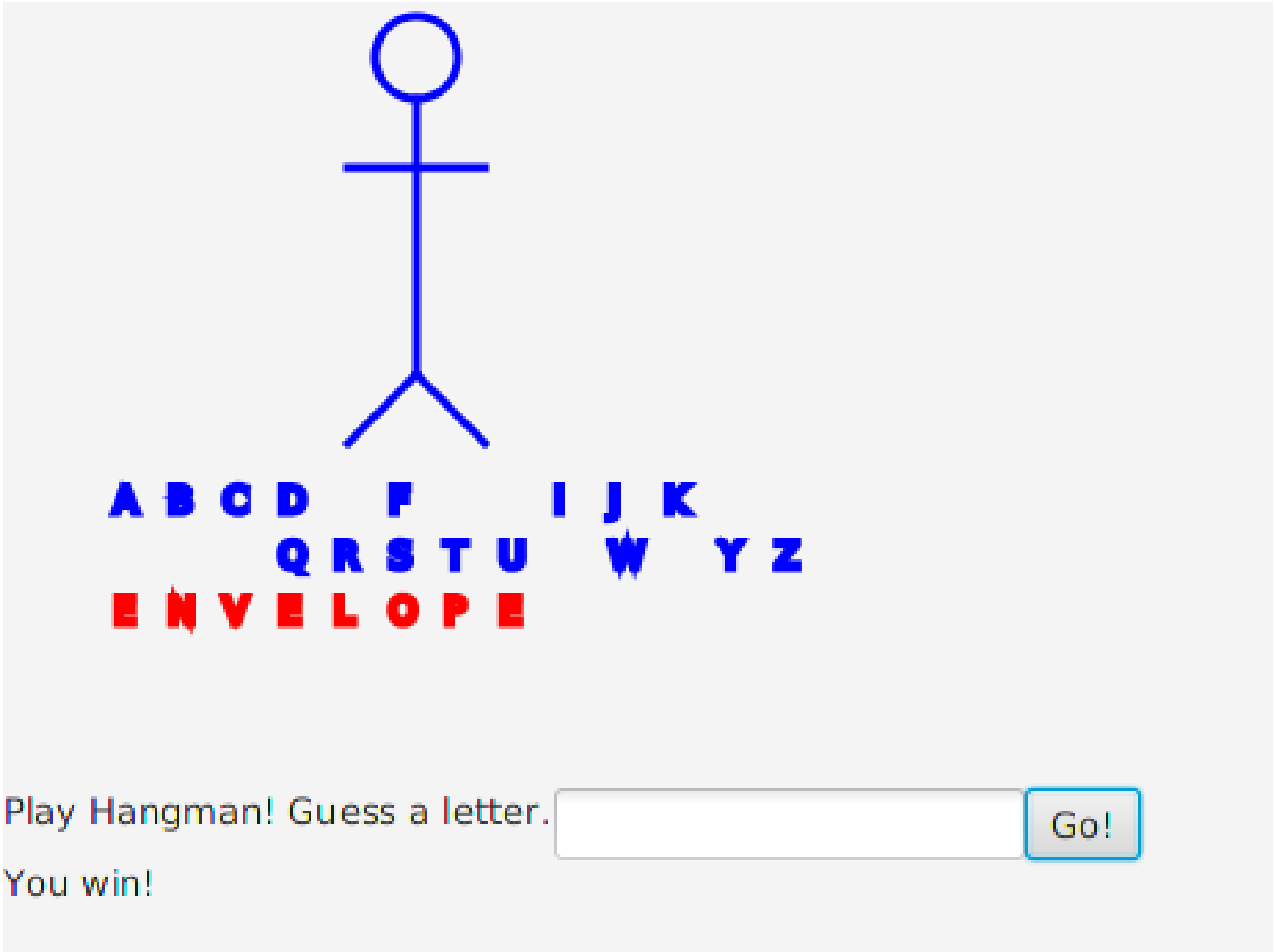
# Hangman

- Today we will be writing a Hangman game from scratch in Java.
- In Hangman, the program maintains a secret word.
- You get 5 wrong guess to select all of the letters in the secret word.
- If you finish the word without losing all of your wrong guesses, you win.
- If you guess wrong 5 times, you lose.

# Let's get started.

- Make a new project called "Hangman".
- JavaFX is already installed on everyone's computers, so there are no special libraries to install. Yay!
- Since this a complicated game, we are going to be writing some object oriented programming.
- Java programmers frequently bounce between files to get code written.

# Hangman


A B C D F I J K
Q R S T U W Y Z
E N V E L O P E
Play Hangman! Guess a letter.
Go!
You win!

# Extend as an Application.

We've got to do some setup first. Find the line that looks like this:

```
public class Hangman {
```

Change it to this:

```
public class Hangman extends Application {
```

# Modify Main

Add this line to `main`. We had to do this for our Turtles.

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

# Errors.

- There is still an error in the code. JavaFX requires a method to work, but NetBeans will introduce it for us.
- The word “Hangman” will have a red squiggly line under it.
- Click the word with your mouse once so that the cursor is in the word.
- Hit “ALT+Enter” on the keyboard.
- Select the first option “Implement all abstract methods.”

# Modify the method.

- You should find a new method in your code. NetBeans added this for us.
- In the method, you'll find this line.

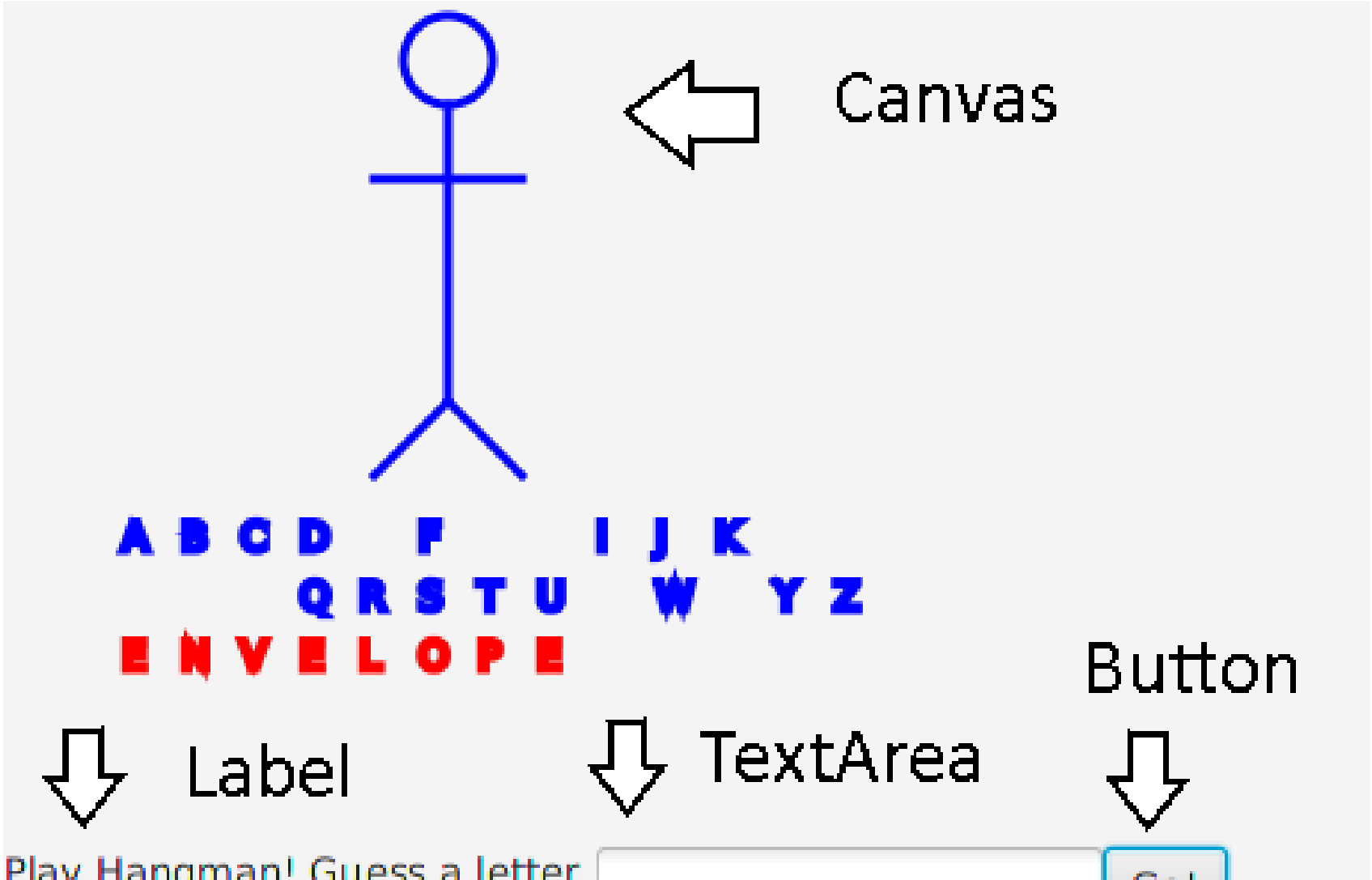
Code.

```
throw new UnsupportedOperationException("...");
```

Remove this line. NetBeans wants you to remove it. Your program will crash if you keep it.

# Let's identify the parts of our game.

There are 5 core parts of our interface. One of these, the Canvas, will be a challenge to write. We will focus on it first.



# Setting up the Canvas

# Make a new class: HangmanGame

- Make a new class called “HangmanGame”.
- Right click on the “hangman” package and select “New...”, then “Java Class...”.
- Name this “HangmanGame”. Make sure that it is in the “hangman” package.
- Your new file should have “package hangman;” at the top. If it doesn’t, delete the file and try again.

# HangmanGame: Add some variables.

These are all variables which our canvas will manage. There's lots going on in Hangman.

```
public class HangmanGame {

    private Canvas canvas;
    private char[] alpha;
    private char[] revealed;
    private int blanks;
    private int lives;
    private String word;
```

# Part 1: Add the Consturctor

The Constructor is a special method used to initialize variables.

```
public HangmanGame(double width, double height) {
    super(width, height);

    alpha = new char[] {'A','B','C','D','E','F','G','H',
                        'I','J','K','L','M','N','O','P',
                        'Q','R','S','T','U','V','W','X',
                        'Y','Z'};
```

We are going to test those typing fingers!

# Part 2: Add the Consturctor

Here, we can set our game's secret word. I picked "ENVELOPE".

```
        lives = 5;
        word = "ENVELOPE";
        revealed = new char[word.length()];
        blanks = word.length();
        canvas = new Canvas(width, height);
    }
```

# HangmanGame. Add these two methdods.

The first method tells us if we lost. The second tells us if we won.

```
public boolean lost() {
    return lives == 0;
}

public boolean won() {
    return blanks == 0;
}
```

# HangmanGame: getCanvas

```
public Canvas getCanvas() {
    return canvas;
}
```

So much of object oriented programming is getting objects to communicate with each other since they don't easily share information. We often have to write short methods like this.

# HangmanGame: scratch method.

Add a method to scratch off a called letter.

```
public void scratch(char letter) {
}
```

# HangmanGame: draw method.

Add a method to draw on the canvas. So far, we have nothing to draw, but we will need this later.

```
public void draw() {
}
```

If anyone asks what `draw` does, it does nothing. Watch this space.

# Building the GUI

# Return to Hangman.java

We finally have enough to start writing our GUI. Return to the `start` method in `Hangman.java`. Here are the five components to our window.

```
Label gameState = new Label("");
Label playHangman =
    new Label("Play Hangman! Guess a letter.");
TextField field = new TextField();
Button button = new Button("Go!");
HangmanGame game = new HangmanGame(300, 300);
```

# Let's build the scene

HBox is a structure that allows for horizontal content. VBox is for vertical content. We need a combination of both.

```
HBox hPane = new HBox();
hPane.getChildren().addAll(
    playHangman, field, button);

VBox vPane = new VBox();
vPane.getChildren().addAll(
    game.getCanvas(), hPane, gameState);
```

# Final Steps for our GUI

These last few lines draw the scene (which current does nothing), sets scene on the stage, then displays the stage.

```
game.draw();

primaryStage.setScene(new Scene(vPane));
primaryStage.show();
```

# Now would be a good time to test your code.

- Test your code.
- Nothing will work and very little will be displayed.
- The goal here is not to have errors in your code.
- I have no way of knowing what time it is, but now might be a good time for a break.

# Part 1. Configure the button

We aren't done with the display code. Now we must configure the button. This is still in the `start` method. Add this statement after "primaryStage.show()".

```
button.setOnAction(event -> {
        char guess = field.getText()
                        .toUpperCase().charAt(0);
        field.setText("");
        game.scratch(guess);
        game.draw();
```

This is a single statement and it is so long that it takes multiple slides to write it all. You should have an error in your code at the end of this. Don't panic!

# Part 1. Configure the button

These last two if statements set the text for the end of the game.

```
            if (game.won())
                gameState.setText("You win!");

            if (game.lost())
                gameState.setText("You lost!");
        });
    }
```

That very last curley bracket signals the end of the start method.

# The Scratch Method

# The Scratch Method

Our scratch method will do two things:

1. Mark the letter off the list of available letters.
2. Reveal the letter in the secret word (if it is there).

Let's get to it!

# Part 1. HangmanGame.java. scratch.

Here's the start of our scratch method. This first for-loop will find a letter in an array. Once found, it will set that letter to be a space. Make sure that you put a space between the two apostrophes.

```
public void scratch(char letter) {
    for (int i = 0; i < alpha.length; i++)
        if (letter == alpha[i])
            alpha[i] = ' ';
```

# Part 2. HangmanGame.java. scratch.

This second part will look for a letter in a word. If it can't be found, the player lost a guess. If it can be found, we keep looking until we find all of them.

```
    int i = word.indexOf(letter);
    if (i == -1)
        lives--;

    while (i != -1) {
        revealed[i] = letter;
        blanks--;
        i = word.indexOf(letter, i+1);
    }
} // End of scratch
```

# The Drawing Method

# The Drawing method

Our draw method will draw the following:

- All of the lost body parts.
- All of the letters that have not yet been called.
- All of the letters that have been revealed.

Because there are so many body parts (5), this method is rather long.

# Part 1. HangmanGame.java. draw()

The first part of `draw` clears the screen. This happens in most video games. Rather than modify the one part of the screen that needs to change, it's equally fast to redraw the entire screen.

```
public void draw() {
    GraphicsContext gc =
        canvas.getGraphicsContext2D();
    gc.clearRect(0, 0, 300, 300);
    gc.setStroke(Color.BLUE);
```

# Part 2. HangmanGame.java. draw()

These if statements draw the body and the two legs.

```
if (lives <= 4)
    gc.strokeLine(150, 50, 150, 150);

if (lives <= 3) {
    gc.strokeLine(125, 175, 150, 150);
    gc.strokeLine(175, 175, 150, 150);
}
```

# Part 3. HangmanGame.java. draw()

These if statements draw the arms and the head.

```
if (lives <= 2)
    gc.strokeLine(125, 75, 175, 75);

if (lives <= 1)
    gc.strokeOval(135, 20, 30, 30);
```

# Part 4: HangmanGame.java. draw()

This if statement draws the face on the head. It's our final body part.

```
if (lives == 0) {
    gc.fillOval(144, 30, 3, 3);
    gc.fillOval(154, 30, 3, 3);
    gc.strokeArc(145, 40, 10, 5, 0, 180,
                 ArcType.OPEN);
}
```

# Part 5: HangmanGame.java. draw()

Final part! This displays (in red) the correctly guessed letters in our secret word.

```
        String display = (new String(alpha)).replace("", " ");
        gc.strokeText(display.substring(0, 28), 40, 200);
        gc.strokeText(display.substring(28), 40, 220);
        gc.setStroke(Color.RED);
        gc.strokeText(new String(revealed).replace("", " "),
            40, 240);
}
```

That last curley bracket represents the end of the `draw` method.

# We are done.

- This is the end of the game.
- We are done.
- Yes, I know it took a lot of code.