# Meme Magic
# Sprint 6 - Feature Add 2

Your team is ready to expand the text-based memes into image-based memes. It's time to create the GUI for the Meme Magic application.

First, set up your project:

- Create a new Java project on Eclipse, and call it `MemeMagic6`.
- Copy and import all of your `MemeMagic5` (or `MemeMagic4`) files into the `MemeMagic6` project. (Remember the Academic Integrity policy from the end of `MemeMagic4` - you are free to share your `MemeMagic4` code with your group members and use `MemeMagic5` code from your group members as long as you understand any code you include.)
- Maintain a copy of `MemeMagic5` (or `MemeMagic4`) files (i.e., do not overwrite them.)

**Note:** you may alternatively copy your project in Eclipse by selecting the `MemeMagic5` project in the Package Explorer, copy with Ctrl+c (or Command+c on mac), and paste with Ctrl+p (or Command+p on mac). On pasting, Eclipse will ask you to name the newly-pasted project.

For this assignment, we've provided a few implementations to start the process:

- Download our `GraphicalMeme.java` file and add it to your project. It extends your Meme class.
- Download our `MemeMagic.java` file and add it to your project. It provides a skeleton of the GUI for Meme Magic. Try it out! (It should look similar to Figure 1.)

## Learning Goals

In this assignment, we will practice:

1. Implementing graphical user interfaces (GUIs) using event-driven programming
2. Organizing GUI content using Java Swing containers and components
3. Handling Exceptions with try/catch blocks

## Implementing the Graphical User Interface

Let's get started. Open the `MemeMagic.java` file that we provided and run it. If everything is working correctly you should see the following screen (Figure 1).

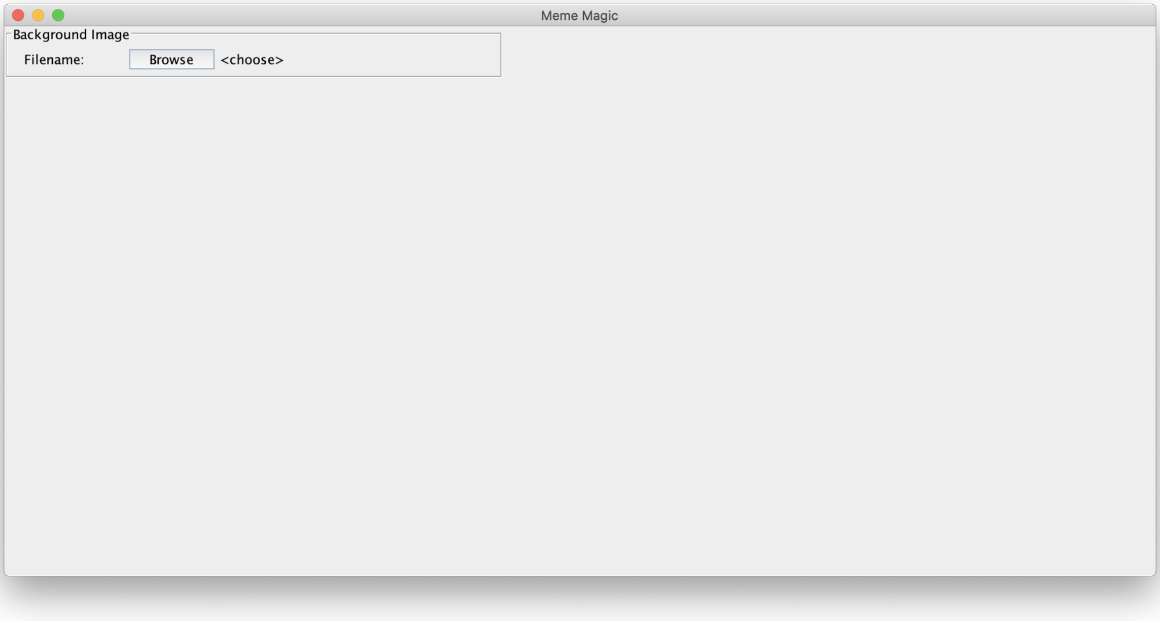


Figure 1: *Screenshot of the initial (scaffolding) interface of Meme Magic (in macOS).*

When we're done your design should look like this (Figure 2).

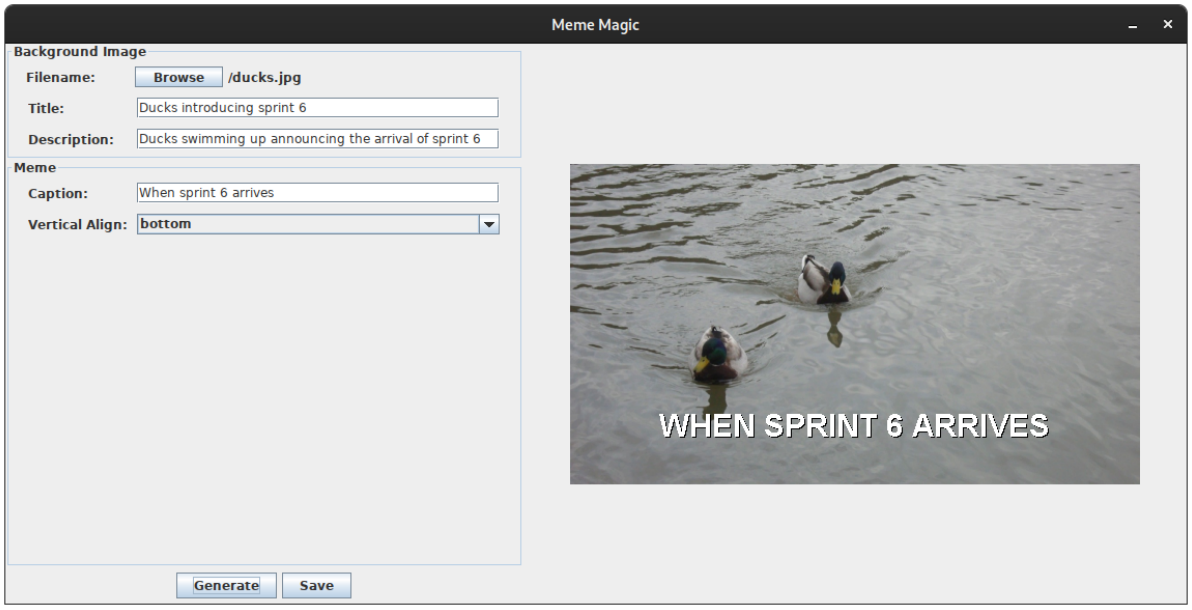


Figure 2: *Screenshot of the Meme Magic application once it is fully implemented (in Linux).*
Photo Credit: John R. Hott

## Implementing the BackgroundImage JPanel

We'll begin by adding the Title and Description fields to the `backgroundImagePanel`. Take a Look at the skeleton code in the `MemeMagic.java` file. We've implemented the first row as an example.

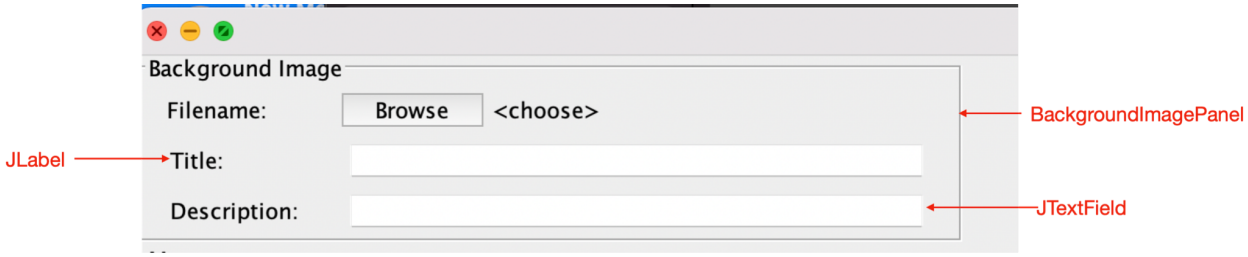


Figure 3: *GUI elements on the BackgroundImage JPanel.*

Adds the following to `backgroundImagePanel`:

1. A JLabel that shows the text value: "Title"
2. A JTextField that allows the user to enter the title they want.
3. A JLabel that shows the text value: "Description"
4. A JTextField that allows the user to enter the description they want.

**Hint:** Remember to group the JLabel and JTextField in a JPanel so that they all line up on one line.

## Connect the OpenButtonListener Listener

Now let's implement the feature that allows the user to click the Browse button and open file explorer dialog. We've implemented that functionally for you, but you will need to connect the OpenButtonListener in the `MemeMagic.java` file to the browse button by adding it as an action listener.

## Implementing the Meme JPanel


Figure 4: *Close-up view of the Meme JPanel.*

This panel (Figure 4) includes a component that allows a user to select items from a dropdown list. You can implement this component using a JComboBox. For examples on how to use the JComboBox and other Swing components, visit this link: [Helpful GUI examples in Swing]. The

combo box should have three options: “top”, “middle”, or “bottom”. This element will allow the user to indicate where the caption will be placed on the image.

Add the Meme JPanel to the JPanel called `controlPanel`. See Figure 5 below.

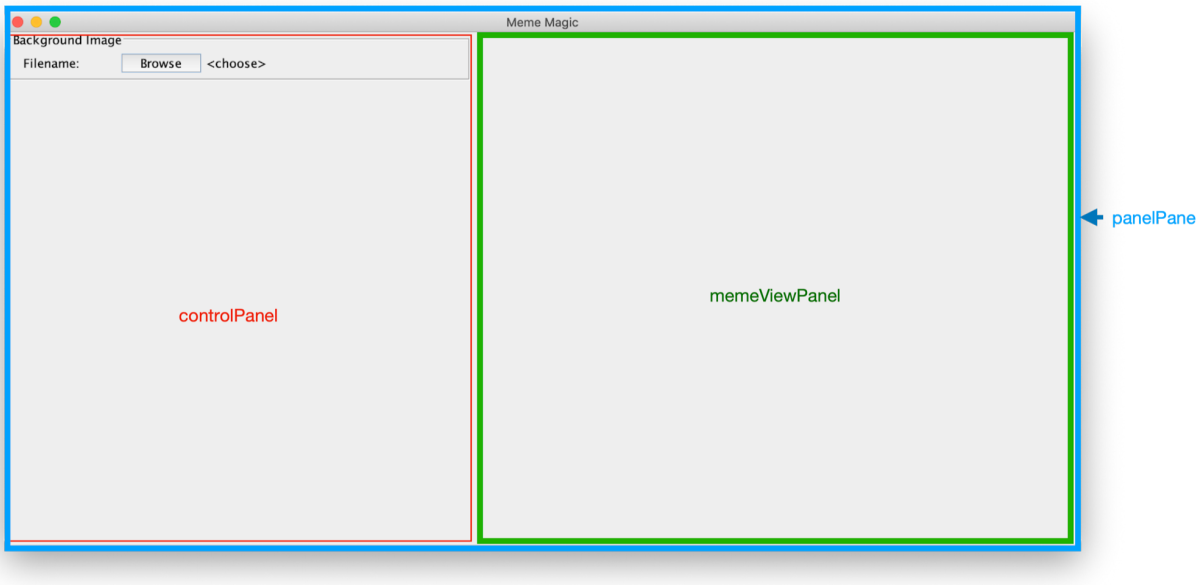


Figure 5: *Nesting JPanels like this allows us to structure the layout of the screen.*

## Implementing the Generate and Save Buttons

Provide the user with a **Generate** button to create and display the `GraphicalMeme` into the `imageDisplayLabel`. Add a listener to the button so that when the button is clicked, the following actions are taken:

- You can call the `getText()` method to JTextField variable to get the text.
    - For example, `String text = ExampleCaptionTextField.getText();`
- Instantiate a `GraphicalMeme` object (it extends your `Meme` class) with the caption, alignment, and background image information provided by the user in the GUI interface.
- Use `GraphicalMeme`’s `compileMeme()` method to compile the Meme into a `BufferedImage` (image data) and display the graphical version of the meme on the provided `imageDisplayLabel`.
    - Catch any exceptions that might be thrown from the `compileMeme()` method.
    - The Java API for `ImageIcon` and `JLabel` and our **Pineapple Pizza Example** on Collab provide resources for how to display an image on a label.

    - **Note**: You will need to `repaint()` the `memeViewPanel` for the image to display.

Add a **Save** button that opens a save dialog box, calls the `compileMeme()` method, and writes the image to a file. We have included a listener (`SaveButtonListener`) that opens the save dialog box and gets the chosen destination filename.

- Modify the listener to include the commented-out `ImageIO.write()` call and handle any exceptions that might be thrown. If an exception occurs, provide the user with an appropriate (and detailed) message of what went wrong, either by printing a message to `System.err` or displaying the message in a new GUI dialog box.
    - The Java API for `ImageIO` and `File` and our documentation for `GraphicalMeme` provide information about which exceptions may be thrown.

## Additional Resources

The following resources may be helpful when completing and submitting this sprint.

- JavaDoc style documentation for each of the classes and methods from Sprint 4
- Video on submitting to Gradescope
- Java API Documentation for
    - ImageIcon
    - ImageIO
    - File
    - JLabel
- Helpful GUI examples in Swing

## Submission Information

**Coding Style**: In real-world software development, it is important to write readable and maintainable code. That is typically achieved through the use of style and commenting guidelines. We have provided a style guide and formatting guide that we strongly encourage you to follow:

- Coding Style Guide (includes installation instructions for Eclipse)
- Eclipse Style File

**Submitting Code**: Upload your Eclipse project (the `.java` files) to the "Meme Magic 6" assignment on Gradescope. You should submit at least `User.java`, `Rating.java`, `BackgroundImage.java`, `Meme.java`, `GraphicalMeme.java`, and `MemeMagic.java`.

**Submitting Demonstration**: Create a short 2-5 minute screen recording demoing your implementation. If you didn't get everything working, that's okay, please show us what you have working and let us know what's missing.

- This video should start by briefly (1-2 minutes max) describing your code: walking through what MemeMagic.java does, what graphical elements you used, and any listeners you created.

- Then, you should run your code to open a MemeMagic window, fill out the fields, and generate a Meme. It can be as funny as you'd like, but please keep it clean. CS-related memes are highly appreciated!
- Use the save button to save your Meme, then open up the created image to show us that your meme saved successfully
- Lastly, close and restart the MemeMagic application and show us your error or Exception handling (display a scenario when one of the try-catch blocks would catch an Exception).

Save and name your video with your name, such as **last-first.mp4**. You can use Zoom to perform the recording, but we ask you to rename your video afterwards. **Upload your video** to the following link: [insert here]

**Note:** Please keep a copy of your video just in case.

**Optional:** Post your generated Meme to the class' "memes" channel (or post).

## Grading Rubric

The assignment will be worth a total of 100 points:

40 points - Graphical interface with all required elements (with video demo)
20 points - Event listeners for generating and saving memes (with video demo)
20 points - Exception handling and appropriate error messages (with video demo)
20 points - Code readability (organized, well-indented, readable by others)

## Where to go from here?

Well, the Meme Magic assignments are done, but the creativity isn't! There are many ways to continue expanding your Meme Magic system. You could customize the look and feel of the memes, provide options to your users when generating them, split the text around the image, build-in services to allow posting to social media, and more! The possibilities are (almost) endless!

We only ask that you do not post your source code publicly (at least without significant changes), since we may use these assignments again in the future.