Approximate Timeline

First draft of project proposal:	Tuesday, end of evening session.
Final draft of project proposal:	Wednesday, end of morning session, (Only if revisions are needed)
Final Project Report/Presentation:	Friday, morning session.

Time Investment:

You will have most of the remaining classtime this week to work on your projects. This means about 10-12 hours. I expect you to remain engaged for the majority of that time, and to have some kind of presentable product at the end. You can take small breaks, but I want to make sure your project is the best it can possibly be!

The Proposal

Before you begin working on the project, you must write a clear project proposal document. You may do this on paper or on your computer. The proposal must include the following 7 sections:
Names: Tell me who is working on your project (government names or hacker names are fine). If working together with others, submit ONE proposal with all of the names.

Brief Description: Summarize your project in a short paragraph or two. Start with a sentence that gives a high-level overview of the project. Then, break down the project into smaller pieces and talk briefly about those.

Example: "Our project is to write image compression software. Our software will feature two types of compression: image resolution reduction and color palette reduction. Resolution reduction works by reducing the number of pixels stored. Color palette reduction works by reducing the number of bits used to store a color (which necessarily reduces the number of colors that can be used). Our program will be able to read in an image from a file as a BufferedImage, and display the result in a JPanel window."

Learning: Tell me what new thing(s) you will have to learn in the completion of the project. The new thing must be something you did not know before entering in the course or learn in the course. The new thing CAN be "I want to learn more about how X topic in the course worked".

Example: "We will need to learn about the math involved in scaling an image down. We will also need to research what colors would be good choices to use in our reduced-color system."

Minimum Viable Product: Tell me what you believe you can definitely get done in a few hours. This can serve as a backup plan in case your full project idea proves to be too challenging.

Example: "We will start by writing a program to half the resolution of an image in each direction. So, an 800x600 image would turn into a 400x300 one. We will also write another program to convert a color picture to grayscale by averaging the red, green, and blue values for each pixel. Each program will take in the name of the input file from the user with a Scanner, and display the output compressed image to the screen. We expect these components to take us about 4-6 hours total to complete."

Stretch Goals: What can you do to improve the project further if you achieve your original vision with time to spare?

Example: "If we finish early, we will add a way to save our images to files. For the resolution reduction version, we can save directly using the code from the ImageCreationExample from class, but for the color reduction one we will need to make our own custom image format, which will require us to create code to write and read files of that format."

Division of Labor (only if working in a group): what will each member of the group be doing to contribute to the project? Whose job will it be to combine all of the parts into one at the end, if needed?

Example: "DJSekora will mostly be working on the color palette reduction component, and ill_ahi will be focusing on the resolution reduction component. We will work together to make the user interface."

Final Project Options

Option 1: Java Programming Project

This is the default option - it's the one most of you will do, in some form or another. You can create any kind of (school-appropriate) application you want: a game (graphical or text-based), a utility application (like a specialized calculator or database system), a tour-de-force multimedia experience - the sky's the limit. If you need to know about some topic we haven't covered, there's a good chance we can help with that.

Option 2: Conceptual Exploration/Research Project

We just barely scratched the surface of many topics in this course - data structures, storage (physical and theoretical), encryption, compression, game physics, more sophisticated graphics, etc.. If any of these topics interested you, you can do a deep dive on one topic. For example, if you wanted to study encryption, you could research different cipher techniques, the history of encryption, the advantages and disadvantages of different methods, the methods of cracking encryption, etc., which could be supplemented with a few programs demonstrating what you've learned.

Option 3: Code Analysis

Take some existing body of code (perhaps an open-source program or project you find online) and analyze it. Preferably, you would have access to the source code, so that you can discuss the code as well as the performance. You would need to put as much time into this as you would a programming project.

A good analysis should include, at a minimum, the following components:

- A high-level description of what the code is for.
  - If the code is trying to solve a problem or meet a need, describe what problem is being solved or what need is being met. Make sure to discuss why this is important, or at least interesting.
  - If the code is a game, describe how it works. What type of game is it? What are the controls? What are the mechanics involved? What is the goal or objective?
  - If the code is another type of entertainment application, describe the intended user experience.
- A discussion of whether the program adequately does what it is supposed to do. Are there obvious bugs? Does it crash frequently? Does it have all the main features it advertises?
- A comparison to other related programs.
  - If there are multiple programs trying to solve the same problem, does the program you chose to analyze solve that problem best? If so, explain what criteria you used to make that decision. If not, explain why another program might be a better choice.
  - If this program is a direct extension, improvement, or remake of an older program, mention this fact. Briefly describe the other program, and what new features are present in this one.

If you have access to the source code, you might also include some of the following components:

- A high-level description of what each file does, and how the different files relate to each other.
- A high-level description of what some of the most important or interesting methods (**at least 3-5 methods**) do - what types of parameters do they take, and what do they return? Are there cases in which they fail which may not be accounted for?

If you do not have access to the source code, you should also include the following:

- Think about what kinds of methods/functions might have been used in the code to accomplish various tasks. Write algorithms, draw flowcharts, or write simulations or code fragments for several of these.

Other types of projects are possible - speak with me if you have an idea that doesn't fit neatly into one of these categories.