Introduction to Objects Lab 1

Lab 1

The previous coding examples for objects have revolved around the `Actor` object. Using a real-world example makes the concept of classes a little easier to understand. But the `Actor` class does not accurately reflect how objects are used in programming. The following pages show how object could be used by a social network like Instagram. Imagine there exists a social network called Photogram that allows you to share, like, and comment on photos. Your feed is comprised of a series of posts containing information like the username, media(image or video), a message, likes, and a list of comments. We are going to create a `Post` object to reflect this.

▼ **Is this how Instagram really works?**
No, not really. This is just an approximation. Facebook undoubtedly uses a more complex and robust system to store, manage, and display Instagram posts.

Components of a Post

It is always a good idea to think about all the various pieces of information that need to be stored in an object. It is also important to think about how that information should be represented. Let's say you have 100 followers. You could represent that as a string, `"100"`. But storing the follower count as an int is a better idea. If you gain a follower, you cannot say `"100" + 1`. You would have to typecast `"100"` as an int, add the new follower, and then typecast the new follower count back to a string. Storing this information as an int is much easier. Here are the elements that make up a post for Photogram:
* Username - The user who creates the post should be stored as a string.
* Id - Some social networks let you change your username. To avoid confusion about usernames, an unique id number is used to refer to each user on a social network. This should be stored as an int.
* Media - Each post has an image or video to display. Media files are often stored elsewhere on a server. The object should store the path to the media file so it can be retrieved and shown to the public. This information should be stored as a string.
* Avatar - The user's avatar should appear next to their post. The object should store the path to an avatar as a string.
* Comment Button - Each post has a button so viewers can add their comment. The object should store the path to this button as a string. **Note**, this will not be a working button.

* Caption - The caption that accompanies the media file should be stored as a string.
* Likes - The number of times people have liked a post should be stored as an int.
* Comments - Comments should be stored as a string. However, each post can have a multitude of comments. So this information should be stored as an ArrayList of strings.
* Like Button - Heart-shaped icon views could click to like a post. This will be stored as a string.

Defining the Post Class

Now that you know all of the attributes needed to create a post for Photogram, you can define the `Post` class and its constructor.

```
//add class definitions below this line

class Post {
  String username;
  int userId;
  String media;
  String avatar;
  String commentButton;
  int likes;
  String caption;
  ArrayList<String> comments;
  String likeButton;

  Post (String un, int ui, String m,
        String a, String cb, int l,
        String ca, ArrayList<String> co,
        String lb) {
    username = un;
    userId = ui;
    media = m;
    avatar = a;
    commentButton = cb;
    caption = ca;
    likes = l;
    comments = co;
    likeButton = lb;
  }
}

//add class definitions above this line
```

Now, declare an instance of the `Post` class with some information. For the sake of readability, each of the parameters will be assigned to a variable. Then, the variables will be passed to the object for instantiation. The elements of the ArrayList are added one at a time. **Note** there is an actual image file that will be used, so be sure the file path is correct.

```
    //add code below this line
    String username = "Sally_17";
    int userId = 112010;
    String media = "studentFolder/photogram/waterfall.png";
    String avatar = "studentFolder/photogram/avatarIcon.png";
    String commentButton =
        "studentFolder/photogram/addComment.png";
    String caption = "First time at Yosemite. It has surpassed
        all of my expectations.";
    int likes = 23;
    ArrayList<String> comments = new ArrayList<String>();
    comments.add("Beautiful!");
    comments.add("I wish I was there too.");
    comments.add("Is that Nevada Falls?");
    comments.add("Love it!");
    comments.add("Can't wait for the Halfdome pictures");
    comments.add("More pics please");
    String likeButton = "studentFolder/photogram/likesIcon.png";

    Post post1 = new Post(username, userId, media, avatar,
                          commentButton, likes, caption,
                          comments, likeButton);

    //add code above this line
```

challenge

Check your work

Print each attribute of `post1` to see that everything is working as expected.

▼ **Code**

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

  //add class definitions below this line
```

```
class Post {
  String username;
  int userId;
  String media;
  String avatar;
  String commentButton;
  int likes;
  String caption;
  ArrayList<String> comments;
  String likeButton;

  Post (String un, int ui, String m,
        String a, String cb, int l,
        String ca, ArrayList<String> co,
        String lb) {
    username = un;
    userId = ui;
    media = m;
    avatar = a;
    commentButton = cb;
    caption = ca;
    likes = l;
    comments = co;
    likeButton = lb;
  }
}

//add class definitions above this line

public class Photogram {
  public static void main(String[] args) {

    //add code below this line
    String username = "Sally_17";
    int userId = 112010;
    String media = "studentFolder/photogram/waterfall.png";
    String avatar = "studentFolder/photogram/avatarIcon.png";
    String commentButton =
      "studentFolder/photogram/addComment.png";
    String caption = "First time at Yosemite. It has surpassed
      all of my expectations.";
    int likes = 23;
    ArrayList<String> comments = new ArrayList<String>();
    comments.add("Beautiful!");
    comments.add("I wish I was there too.");
    comments.add("Is that Nevada Falls?");
    comments.add("Love it!");
    comments.add("Can't wait for the Halfdome pictures");
    comments.add("More pics please");
```

```
      String likeButton =
        "studentFolder/photogram/likesIcon.png";

      Post post1 = new Post(username, userId, media, avatar,
                            commentButton, likes, caption,
                            comments, likeButton);

      System.out.println(post1.username);
      System.out.println(post1.userId);
      System.out.println(post1.media);
      System.out.println(post1.avatar);
      System.out.println(post1.commentButton);
      System.out.println(post1.caption);
      System.out.println(post1.likes);
      System.out.println(post1.comments);
      System.out.println(post1.likeButton);

      //add code above this line
    }
  }
```

Introduction to Objects Lab 2

Lab 2

Now that you have all of the information needed to make a post, you can turn that into visual output using the `Swing` library. `Swing` allows you to build simple graphical user interfaces with a minimal amount of code. The next lab is going to be an introduction to `Swing` and a few of its features.

▼ **What to learn more about `Swing`?**
These labs will only cover a tiny fraction of what can be done with the `Swing` library. The full documentation for `Tkinter` can be found here. This documentation is not very user-friendly. A more beginner-friendly way to learn about `Swing` is a video tutorial like this YouTube playlist.

Main Window

`Swing`'s output is a window. There are four steps needed to get a window up and running. First, create a `JFrame` object and pass it a string that represents the text to appear in the title bar of the window. Second, set the size (in pixels) of the window. Three, tell Java to stop the program when you close the window. Finally, set the window visibility to `true`.

```
    //add code below this line

    JFrame window = new JFrame("Hello");
    window.setSize(300, 300);
    window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    window.setVisible(true);

    //add code above this line
```

That is really all it takes to get a window up and running. You can adjust the title and size by changing the arguments with `setSize` and when initializing the `window` object.

Label Widget - Text

A `JLabel` is an element that can contain text, and image, or both. Labels are going to be the building block for these labs.

Swing Label

Create a text label for your window. Be sure to add the label to the `window` object so that it appears on the screen.

```
//add code below this line

JFrame window = new JFrame("Hello");
window.setSize(300, 300);

JLabel text = new JLabel("I am a label");
window.add(text);

window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
window.setVisible(true);

//add code above this line
```

Other options for text `Labels`:
* Font - You can set the font family and size by creating a `Font` object. The available fonts are `Monospaced`, `Serif`, `SansSerif`, `Dialog`, `DialogInput`. You can also set the font to `BOLD`, `ITALIC`, or `PLAIN` as well. Use the `setFont` method to change the font for the label.

```
JLabel text = new JLabel("I am a label");
Font font = new Font("Serif", Font.BOLD, 24);
text.setFont(font);
window.add(text);
```

- Justify - You can justify text in a label when creating a `JLabel` object. The three options are `LEFT`, `CENTER`, and `RIGHT`.

```
JLabel text = new JLabel("I am a label", JLabel.CENTER);
```

- Background Color - Before you can set the background color of a label, you must first make the background opaque. Then you can select a color. Here is a list of colors you can use.

```
text.setOpaque(true);
text.setBackground(Color.blue);
```

- Text Color - The `setForeground` method is used to set the color of a label's text. Use the same colors in the link above.

```
text.setForeground(Color.red);
```

Grid System

Adding elements to the window is a two-step process. First define the element, then place it in the window using the grid system. Use the `setLayout` method to make a grid. The `Grid` object takes two arguments, the number of rows and the number of columns. The order in which you add the labels to `window` determines the position. Java places the elements from left to right and top to bottom.

```
    //add code below this line

    JFrame window = new JFrame("Hello");
    window.setSize(300, 300);
    window.setLayout(new GridLayout(2, 2));

    JLabel text1 = new JLabel("One", JLabel.CENTER);
    JLabel text2 = new JLabel("Two", JLabel.CENTER);
    JLabel text3 = new JLabel("Three", JLabel.CENTER);
    JLabel text4 = new JLabel("Four", JLabel.CENTER);

    window.add(text1);
    window.add(text2);
    window.add(text3);
    window.add(text4);

    window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    window.setVisible(true);

    //add code above this line
```

challenge

Try this variation:

Change the order in which the text labels are added to the window.

```
    window.add(text4);
    window.add(text2);
    window.add(text1);
    window.add(text3);
```

Image Label

Using an image in a label is also a two-step process. First create an image object for `Tkinter`, then attach the image to the label by replacing the `text` option with `image`. You still need to use `grid` to place the image in the window.

```
    //add code below this line

    JFrame window = new JFrame("Hello");
    window.setSize(300, 300);

    ImageIcon feather = new
        ImageIcon("studentFolder/feather.png");
    JLabel image = new JLabel(feather);
    window.add(image);

    window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    window.setVisible(true);

    //add code above this line
```

challenge

Explore `Swing`

- Try out various fonts and font sizes
- Experiment with different colors
- Position labels around the window with the grid system

Introduction to Objects Lab 3

Lab 3

Now that you have had a brief introduction to `Swing`, you are going to create a mockup of a post on Photogram using the information stored in the `Post` object created in Lab 1. Before beginning, remove all of the print statements in the `main` method.

Setting up the Window

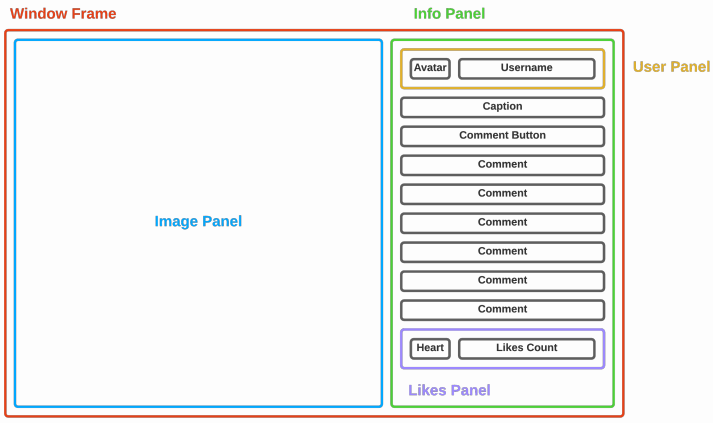
The first step is to setup the window for the app. Just as in Lab 2, you are going to create a window, give it a title and set the size. In addition, set the layout to `FlowLayout`. This means panels added to the window will be added left to right.

```
    Post post1 = new Post(username, userId, media, avatar,
                          commentButton, likes, caption,
                          comments, likeButton);

    // create window
    JFrame window = new JFrame("Photogram");
    window.setSize(800, 500);
    window.setLayout(new FlowLayout(3));

    // add panels to window
    window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    window.setVisible(true);
```

It is important to know in advance what the layout of the app should be. This will influence how the window is created. The image below shows the general layout of Photogram.



Photogram Layout

Add Panels

This project uses several panels to contain information. In some cases, there will be panels inside other panels. Create the following panels. The `infoPanel` uses a `GridLayout` which means content will be added vertically. The `infoPanel` has 10 rows (the size of the `comments` ArrayList plus 4) and 1 column. The panels with a `FlowLayout` will add content horizontally. The arguments for `FlowLayout` mean that the components will be aligned to the left (the `3`), have a vertical gap of 0 pixels, and a horizontal gap of 5 pixels.

```
// create window
JFrame window = new JFrame("Photogram");
window.setSize(800, 500);
window.setLayout(new FlowLayout(3));

// create panels
JPanel imagePanel = new JPanel();
JPanel infoPanel = new JPanel(new
    GridLayout(comments.size()+4, 1));
JPanel userPanel = new JPanel(new FlowLayout(3, 0, 5));
JPanel likesPanel = new JPanel(new FlowLayout(3, 0, 5));
```

Be sure that `imagePanel` and `infoPanel` are added to the window. The panels for the user and likes will be added later. Running the program produces a window that is 800 pixels by 500 pixels. You cannot see the panels. Be sure to close the window before running the code again.

```
// add panels to window
window.add(imagePanel);
window.add(infoPanel);
window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
window.setVisible(true);
```

▼ **Larger viewing area**

Click the blue triangle icon to open the `Swing` project in a new tab. You can leave this tab open. Each time you click a `TRY IT` button, the tab will update with the latest version of your project.

Create Fonts

This project uses two different fonts. Both of them will be from the `SansSerif` family, and they will both be size 14. However, one will be `BOLD` while the other is `PLAIN`.

```
    // create panels
    JPanel imagePanel = new JPanel();
    JPanel infoPanel = new JPanel(new
        GridLayout(comments.size()+4, 1));
    JPanel userPanel = new JPanel(new FlowLayout(3, 0, 5));
    JPanel likesPanel = new JPanel(new FlowLayout(3, 0, 5));

    // create fonts
    Font boldFont = new Font("SansSerif", Font.BOLD, 14);
    Font plainFont = new Font("SansSerif", Font.PLAIN, 14);
```

Running the program should produce no visible changes. However, you want to make sure your code runs before moving on the next page.

▼ **Code**

Your code should look like this:

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

//add class definitions below this line

class Post {
  String username;
  int userId;
  String media;
  String avatar;
  String commentButton;
  int likes;
  String caption;
  ArrayList<String> comments;
  String likeButton;

  Post (String un, int ui, String m,
        String a, String cb, int l,
```

```
        String ca, ArrayList<String> co,
        String lb) {
    username = un;
    userId = ui;
    media = m;
    avatar = a;
    commentButton = cb;
    caption = ca;
    likes = l;
    comments = co;
    likeButton = lb;
  }
}

//add class definitions above this line

public class Photogram {
  public static void main(String[] args) {

    //add code below this line
    String username = "Sally_17";
    int userId = 112010;
    String media = "studentFolder/photogram/waterfall.png";
    String avatar = "studentFolder/photogram/avatarIcon.png";
    String commentButton =
      "studentFolder/photogram/addComment.png";
    String caption = "First time at Yosemite. It has surpassed
      all of my expectations.";
    int likes = 23;
    ArrayList<String> comments = new ArrayList<String>();
    comments.add("Beautiful!");
    comments.add("I wish I was there too.");
    comments.add("Is that Nevada Falls?");
    comments.add("Love it!");
    comments.add("Can't wait for the Halfdome pictures");
    comments.add("More pics please");
    String likeButton =
      "studentFolder/photogram/likesIcon.png";

    Post post1 = new Post(username, userId, media, avatar,
                          commentButton, likes, caption,
                          comments, likeButton);

    // create window
    JFrame window = new JFrame("Photogram");
    window.setSize(800, 500);
    window.setLayout(new FlowLayout(3));

    // create panels
```

```
        JPanel imagePanel = new JPanel();
        JPanel infoPanel = new JPanel(new
          GridLayout(comments.size()+4, 1));
        JPanel userPanel = new JPanel(new FlowLayout(3, 0, 5));
        JPanel likesPanel = new JPanel(new FlowLayout(3, 0, 5));

        // create fonts
        Font boldFont = new Font("SansSerif", Font.BOLD, 14);
        Font plainFont = new Font("SansSerif", Font.PLAIN, 14);

        // add panels to window
        window.add(imagePanel);
        window.add(infoPanel);
        window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        window.setVisible(true);

        //add code above this line
    }
}
```

Introduction to Objects Lab 4

Lab 4

In this lab, you will add the main image, an avatar, the username, and a caption to the app. All of these components will follow a two-step process: create the component and then add it to a panel.

Add the Main Image

To add an image, you first need to make an `ImageIcon` object. Use the string stored in the `post1.media` attribute to tell Java where to find the image. Then add the image object to a `JLabel` object. Finally, add the label to the `imagePanel`.

```
    // create fonts
    Font boldFont = new Font("SansSerif", Font.BOLD, 14);
    Font plainFont = new Font("SansSerif", Font.PLAIN, 14);

    // create waterfall image
    ImageIcon image = new ImageIcon(post1.media);
    JLabel imageLabel = new JLabel(image);
    imagePanel.add(imageLabel);
```

Add the Avatar

Remember, the avatar and username both go in the `userPanel`, not the `imagePanel`. Just like the main image, create an `ImageIcon` object with the string in the `post1.avatar` attribute. Add the image to a `JLabel`. Add the `avatarIcon` to the `userPanel`. **Note**, you will not see the avatar because the `userPanel` has not yet been added to `infoPanel`.

```
    // create waterfall image
    ImageIcon image = new ImageIcon(post1.media);
    JLabel imageLabel = new JLabel(image);
    imagePanel.add(imageLabel);

    // create avatar image
    ImageIcon avatarImage = new ImageIcon(post1.avatar);
    JLabel avatarIcon = new JLabel(avatarImage, JLabel.LEFT);
    userPanel.add(avatarIcon);
```

Add the Username

Create a `JLabel` object with the string found in the `post1.username` attribute. Use `JLabel.LEFT` to tell Java that the text should be aligned to the far left of the label. We also want to use a bold font for the username. Use the `setFont` method and the `boldFont` object to make the text appear in bold. Add `usernameLabel` to `userPanel`, and then add `userPanel` to `infoPanel`.

```
    // create avatar image
    ImageIcon avatarImage = new ImageIcon(post1.avatar);
    JLabel avatarIcon = new JLabel(avatarImage, JLabel.LEFT);
    userPanel.add(avatarIcon);

    // create username text
    JLabel usernameLabel = new JLabel(post1.username,
        JLabel.LEFT);
    usernameLabel.setFont(boldFont);
    userPanel.add(usernameLabel);
    infoPanel.add(userPanel);
```

Add the Caption

The caption will be a bit different than the username. The caption is a long sentence, so we need to have word wrapping. To do this, treat the caption as HTML. Create a string variable for the `post1.caption` attribute. Then create the `html` variable. The text `%1spx` is used to tell Java how "wide" the text can be before going to the next line. The `%1s` tells Java what text is to be applied to the HTML formatting. The `captionLabel` uses `String.format` to set the width of the caption to 200 pixels, and the contents of the `txt` variable will be treated as HTML. Use a plain font for the caption, and add it to `infoPanel`.

```
  // create username text
  JLabel usernameLabel = new JLabel(post1.username,
      JLabel.LEFT);
  usernameLabel.setFont(boldFont);
  userPanel.add(usernameLabel);
  infoPanel.add(userPanel);

  // create caption text
  String txt = post1.caption;
  String html = "<html><body style='width: %1spx'>%1s";
  JLabel captionLabel = new JLabel(String.format(html, 200,
      txt), JLabel.LEFT);
  captionLabel.setFont(plainFont);
  infoPanel.add(captionLabel);
```

▼ Code

Your code should look like this:

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

//add class definitions below this line

class Post {
  String username;
  int userId;
  String media;
  String avatar;
  String commentButton;
  int likes;
  String caption;
  ArrayList<String> comments;
  String likeButton;

  Post (String un, int ui, String m,
        String a, String cb, int l,
        String ca, ArrayList<String> co,
        String lb) {
    username = un;
    userId = ui;
    media = m;
    avatar = a;
    commentButton = cb;
    caption = ca;
    likes = l;
```

```
    comments = co;
    likeButton = lb;
  }
}

//add class definitions above this line

public class Photogram {
  public static void main(String[] args) {

    //add code below this line
    String username = "Sally_17";
    int userId = 112010;
    String media = "studentFolder/photogram/waterfall.png";
    String avatar = "studentFolder/photogram/avatarIcon.png";
    String commentButton =
      "studentFolder/photogram/addComment.png";
    String caption = "First time at Yosemite. It has surpassed
      all of my expectations.";
    int likes = 23;
    ArrayList<String> comments = new ArrayList<String>();
    comments.add("Beautiful!");
    comments.add("I wish I was there too.");
    comments.add("Is that Nevada Falls?");
    comments.add("Love it!");
    comments.add("Can't wait for the Halfdome pictures");
    comments.add("More pics please");
    String likeButton =
      "studentFolder/photogram/likesIcon.png";

    Post post1 = new Post(username, userId, media, avatar,
                          commentButton, likes, caption,
                          comments, likeButton);

    // create window
    JFrame window = new JFrame("Photogram");
    window.setSize(800, 500);
    window.setLayout(new FlowLayout(3));

    // create panels
    JPanel imagePanel = new JPanel();
    JPanel infoPanel = new JPanel(new
      GridLayout(comments.size()+4, 1));
    JPanel userPanel = new JPanel(new FlowLayout(3, 0, 5));
    JPanel likesPanel = new JPanel(new FlowLayout(3, 0, 5));

    // create fonts
    Font boldFont = new Font("SansSerif", Font.BOLD, 14);
    Font plainFont = new Font("SansSerif", Font.PLAIN, 14);
```

```
        // create waterfall image
        ImageIcon image = new ImageIcon(post1.media);
        JLabel imageLabel = new JLabel(image);
        imagePanel.add(imageLabel);

        // create avatar image
        ImageIcon avatarImage = new ImageIcon(post1.avatar);
        JLabel avatarIcon = new JLabel(avatarImage, JLabel.LEFT);
        userPanel.add(avatarIcon);

        // create username text
        JLabel usernameLabel = new JLabel(post1.username,
          JLabel.LEFT);
        usernameLabel.setFont(boldFont);
        userPanel.add(usernameLabel);
        infoPanel.add(userPanel);

        // create caption text
        String txt = post1.caption;
        String html = "<html><body style='width: %1spx'>%1s";
        JLabel captionLabel = new JLabel(String.format(html, 200,
          txt), JLabel.LEFT);
        captionLabel.setFont(plainFont);
        infoPanel.add(captionLabel);

        // add panels to window
        window.add(imagePanel);
        window.add(infoPanel);
        window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        window.setVisible(true);

        //add code above this line
    }
}
```

Introduction to Objects Lab 5

Lab 5

We will finish up by adding the components for the comment button, the comments, the like icon, and the number of likes.

Add the Comment Button

Just as with the other images, create a `ImageIcon` object with the string in the `post1.commentButton` attribute. Add the image to a `JLabel`, and then add the label to `infoPanel`.

```
    // create caption text
    String txt = post1.caption;
    String html = "<html><body style='width: %1spx'>%1s";
    JLabel captionLabel = new JLabel(String.format(html, 200,
        txt), JLabel.LEFT);
    captionLabel.setFont(plainFont);
    infoPanel.add(captionLabel);


    // create "+" image
    ImageIcon commentButtonImage = new
        ImageIcon(post1.commentButton);
    JLabel commentButtonLabel = new JLabel(commentButtonImage);
    infoPanel.add(commentButtonLabel);
```

Add the Comments

The comments are stored in an ArrayList in the `comments` attribute of `post1`. Iterate over the ArrayList. Create a new `JLabel` for each comment and then add it to `infoPanel`.

```
// create "+" image
ImageIcon commentButtonImage = new
    ImageIcon(post1.commentButton);
JLabel commentButtonLabel = new JLabel(commentButtonImage);
infoPanel.add(commentButtonLabel);

// create user comments
for (String comment : post1.comments) {
  JLabel commentLabel = new JLabel(comment, JLabel.LEFT);
  commentLabel.setFont(plainFont);
  infoPanel.add(commentLabel);
}
```

Add the Like Icon

Like the avatar and the username, the like icon and the number of likes will be side-by-side. To do this, these components will first go into the `likesPanel`, which will then be added to the `infoPanel`. **Note**, you will not see the heart icon because the `likesPanel` has not yet been added to the `infoPanel`.

```
// create user comments
for (String comment : post1.comments) {
  JLabel commentLabel = new JLabel(comment, JLabel.LEFT);
  commentLabel.setFont(plainFont);
  infoPanel.add(commentLabel);
}

// create heart image
ImageIcon likesImage = new ImageIcon(post1.likeButton);
JLabel likesIconLabel = new JLabel(likesImage, JLabel.LEFT);
likesPanel.add(likesIconLabel);
```

Add the Likes

The attribute `likes` is an integer. In order to add this value to a `JLabel`, you need to convert it to a string with the `String.valueOf` method. Set the font to plain. Then add the label to `likesPanel` and the `likesPanel` to the `infoPanel`.

```
    // create heart image
    ImageIcon likesImage = new ImageIcon(post1.likeButton);
    JLabel likesIconLabel = new JLabel(likesImage, JLabel.LEFT);
    likesPanel.add(likesIconLabel);

    // create number of likes
    JLabel likesCountLabel = new
        JLabel(String.valueOf(post1.likes), JLabel.LEFT);
    likesCountLabel.setFont(plainFont);
    likesPanel.add(likesCountLabel);
    infoPanel.add(likesPanel);
```

Fix the Spacing

Finally, we want the information in the `infoPanel` to be "pushed up" so it aligns with the top of the waterfall image. To do this, we are going to add more rows to the `infoPanel`. When declaring `infoPanel`, add 7 to the size of the `comments` ArrayList.

```
    // create panels
    JPanel imagePanel = new JPanel();
    JPanel infoPanel = new JPanel(new
        GridLayout(comments.size()+7, 1));
    JPanel userPanel = new JPanel(new FlowLayout(3, 0, 5));
    JPanel likesPanel = new JPanel(new FlowLayout(3, 0, 5));
```

▼ **Code**

Your code should look like this:

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

//add class definitions below this line

class Post {
  String username;
  int userId;
  String media;
  String avatar;
  String commentButton;
  int likes;
  String caption;
  ArrayList<String> comments;
  String likeButton;
```

```
  Post (String un, int ui, String m,
        String a, String cb, int l,
        String ca, ArrayList<String> co,
        String lb) {
    username = un;
    userId = ui;
    media = m;
    avatar = a;
    commentButton = cb;
    caption = ca;
    likes = l;
    comments = co;
    likeButton = lb;
  }
}

//add class definitions above this line

public class Photogram {
  public static void main(String[] args) {

    //add code below this line
    String username = "Sally_17";
    int userId = 112010;
    String media = "studentFolder/photogram/waterfall.png";
    String avatar = "studentFolder/photogram/avatarIcon.png";
    String commentButton =
      "studentFolder/photogram/addComment.png";
    String caption = "First time at Yosemite. It has surpassed
      all of my expectations.";
    int likes = 23;
    ArrayList<String> comments = new ArrayList<String>();
    comments.add("Beautiful!");
    comments.add("I wish I was there too.");
    comments.add("Is that Nevada Falls?");
    comments.add("Love it!");
    comments.add("Can't wait for the Halfdome pictures");
    comments.add("More pics please");
    String likeButton =
      "studentFolder/photogram/likesIcon.png";

    Post post1 = new Post(username, userId, media, avatar,
                          commentButton, likes, caption,
                          comments, likeButton);

    // create window
    JFrame window = new JFrame("Photogram");
    window.setSize(800, 500);
```

```
        window.setLayout(new FlowLayout(3));

        // create panels
        JPanel imagePanel = new JPanel();
        JPanel infoPanel = new JPanel(new
          GridLayout(comments.size()+7, 1));
        JPanel userPanel = new JPanel(new FlowLayout(3, 0, 5));
        JPanel likesPanel = new JPanel(new FlowLayout(3, 0, 5));

        // create fonts
        Font boldFont = new Font("SansSerif", Font.BOLD, 14);
        Font plainFont = new Font("SansSerif", Font.PLAIN, 14);

        // create waterfall image
        ImageIcon image = new ImageIcon(post1.media);
        JLabel imageLabel = new JLabel(image);
        imagePanel.add(imageLabel);

        // create avatar image
        ImageIcon avatarImage = new ImageIcon(post1.avatar);
        JLabel avatarIcon = new JLabel(avatarImage, JLabel.LEFT);
        userPanel.add(avatarIcon);

        // create username text
        JLabel usernameLabel = new JLabel(post1.username,
          JLabel.LEFT);
        usernameLabel.setFont(boldFont);
        userPanel.add(usernameLabel);
        infoPanel.add(userPanel);

        // create caption text
        String txt = post1.caption;
        String html = "<html><body style='width: %1spx'>%1s";
        JLabel captionLabel = new JLabel(String.format(html, 200,
          txt), JLabel.LEFT);
        captionLabel.setFont(plainFont);
        infoPanel.add(captionLabel);

        // create "+" image
        ImageIcon commentButtonImage = new
          ImageIcon(post1.commentButton);
        JLabel commentButtonLabel = new
          JLabel(commentButtonImage);
        infoPanel.add(commentButtonLabel);

        // create user comments
        for (String comment : post1.comments) {
          JLabel commentLabel = new JLabel(comment, JLabel.LEFT);
          commentLabel.setFont(plainFont);
          infoPanel.add(commentLabel);
```

```
        }

        // create heart image
        ImageIcon likesImage = new ImageIcon(post1.likeButton);
        JLabel likesIconLabel = new JLabel(likesImage,
          JLabel.LEFT);
        likesPanel.add(likesIconLabel);

        // create number of likes
        JLabel likesCountLabel = new
          JLabel(String.valueOf(post1.likes), JLabel.LEFT);
        likesCountLabel.setFont(plainFont);
        likesPanel.add(likesCountLabel);
        infoPanel.add(likesPanel);

        // add panels to window
        window.add(imagePanel);
        window.add(infoPanel);
        window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        window.setVisible(true);

        //add code above this line
    }
}
```

Introduction to Objects Lab Challenge

Lab Challenge

Create the variable `dog1`, and instantiate an object of the `Dog` class. This dog's name is `Marceline` and she is a `German Shepherd`. Create the variable `dog2` and make a **deep copy** of `dog1`. `dog2` should be named `Cajun` and have the breed `Belgian Malinois`.

Expected Output

Test your code by printing the `name` and `breed` of each dog to make sure they fulfill the requirements above. Most importantly, the third print statement will print `false`.

```
    //add code below this line

    Dog dog1 = new Dog("Marceline", "German Shepherd");
    Dog dog2 = new Dog(dog1);
    dog2.name = "Cajun";
    dog2.breed = "Belgian Malinois";

    System.out.println(dog1.name + " " + dog1.breed);
    System.out.println(dog2.name + " " + dog2.breed);
    System.out.println(dog2 == dog1);

    //add code above this line
```