CS162 – summer 2015 Project #1 - Digital Artists

Due Date

Thursday, May 21, 2015 at 6pm

Preparation

- Watch the first 5 minutes of following YouTube video: https://www.youtube.com/watch?v=HGVNjrIyw74 (the last 2:30 minutes of the video can be skipped)
- Know how to compile and run Java Applets on BlueJ
- Read this entire document before starting the project
- Track the amount of time you spend on the project (you are required to turn in your "time card")

Learning Objectives

After completing this project you should be able to:

- write, compile, and run a simple Java applet
- declare variables to store integers and Strings
- assign values to variables
- use arithmetic expressions as arguments in a method call
- describe the difference between a Java application and applet

Project Requirements

Recreate your favorite childhood drawing on a Java applet. Your graphics artwork must satisfy the following requirements:

- (5 pts) Overall canvas size is approximately 500x300
- (5 pts) Draw a 500x300 rectangle in a color of your choice as the "background"
- (20 pts) Include all of the following drawing elements:

- Lines
- Rectangle (or rounded rectangles)
- Circles or ovals
- Text (your name should be one of the text elements)

- (10 pts) Use at least two colors. Refer to "Using Colors" section below.
- (10 pts) Render the text using at least two different font sizes (or styles). Refer to "Changing Font" section below)
- (10 pts) Use variables with meaningful names to hold different quantities related to the drawing (refer to "Using Variables" section below)
- (15 pts) Use arithmetic expressions in method call arguments (refer to "Using Variables" section below)
- (5 pts) The overall artwork should be attractive and well-balanced
- (10 pts) Include a rough sketch of your artwork (use your phone, a digital camera, or a scanner to produce a digital image). Upload a digital copy to Bb

- (10 pts) Well documented and elegant source code that follows Java Style Guide. Your paint() method shall include sufficient comments that explain what each block of code does

Using Colors

The setColor() method takes a single parameter of type Color and you have at least two options for using colors

- Use one of the predefined Java colors: Color.RED, Color.BLUE, Color.WHITE … (The complete list of these colors can be found here).
  g.setColor (Color.MAGENTA);
- Enter your own RGB values. The following example creates a color of RGB (54, 106, 184):
  Color myFavoriteBlue = new Color (54, 106, 184);
  g.setColor (myFavoriteBlue);

  A Google search for "color generator" should find online interactive tools that allow you to use a color palette.

Using Variables

Professional code uses variables to store values. Variables are typically declared at the top of a method and then assigned values as needed. Compare the following two techniques for drawing two black wheels. Obviously, the code on the right is easier to understand (and easier to modify).

Also observe how arithmetic expression (wheelY + wheelDistance) is used as an argument to the method invocation.

With no variables	Using variables

g.setColor (Color.BLACK);
/* draw the front wheel */
g.fillOval (30, 40, 60, 60);
/* draw the rear wheel */
g.fillOval (30, 130, 60, 60);
	int wheelX = 30;
	int wheelY = 40;
	int wheelDistance = 90;
	int wheelDia = 60;
	g.setColor (Color.BLACK);
	/* draw the front wheel */
	g.fillOval (wheelX, wheelY, wheelDia, wheelDia);
	/* draw the rear wheel */
	g.fillOval (wheelX, wheelY + wheelDistance,
	wheelDia, wheelDia);

Changing Font

The steps for changing colors can be applied to changing fonts.
Font myChoiceOfFont = new Font ("serif", Font.ITALIC, 20);

g.setFont (myChoiceOfFont);

g.drawString ("Hello", _____, _____);

The above example renders the text "Hello" using a 20-point serif (font name) in italic (font style).

- Other font names are: "sanserif", "monospace", "Times", "Helvetica", "Courier", etc.
- Font styles include Font.PLAIN, Font.BOLD (or combination of these styles: Font.BOLD+Font.ITALIC)

Extra Credits

The following should only be attempted after all of the above requirements have been completed. Challenge exercises demonstrate that you have the initiative to investigate problems and identify solutions with minimal help from your instructor.

- (10 pts) Add a photo of yourself

To include images you must import the following additional packages:

```
import java.awt.image.*;
import java.net.*;
import javax.imageio.*;
import java.io.*;
```

The following code assumes you have an image "MyPhoto.jpg" in the directory of your BlueJ project:

```
BufferedImage photo = null;

try {
  URL photoUrl = new URL (getCodeBase(), "MyPhoto.jpg");
  photo = ImageIO.read (photoUrl);
  /* render the image into a 150x200 rectangle */
  g.drawImage (photo, 50, 80, 150, 200, null);
}

catch (IOException e) {
  g.drawString ("Problem rendering photo", 50, 80);
}
```

Late Policy

Project should be turned in on time at the START of the class period. Projects submitted after the deadline will be penalized according to the following policy:

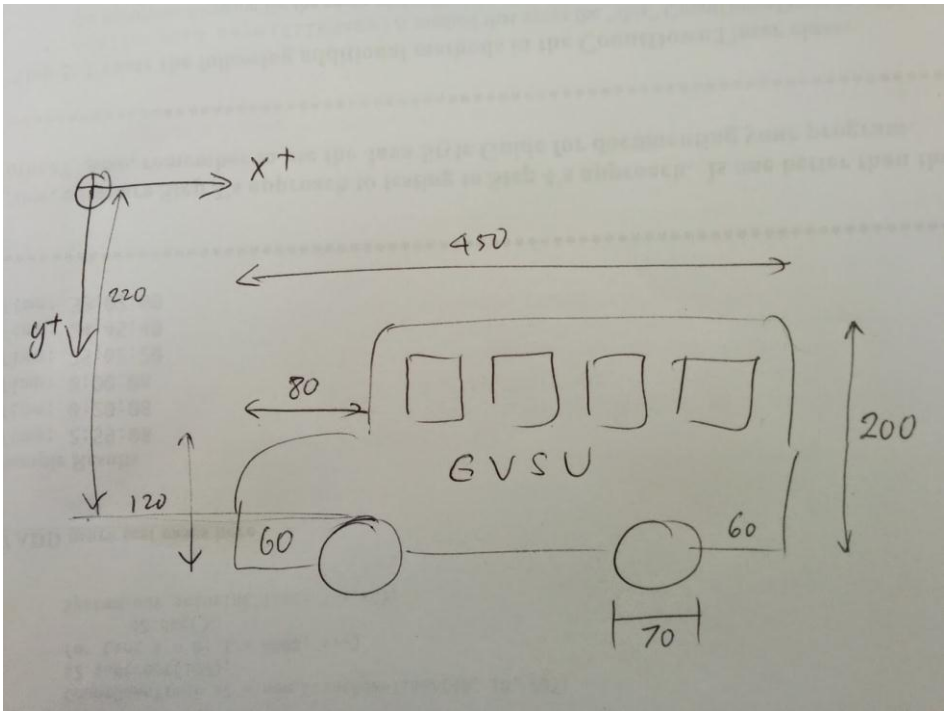
- First day (-20 points)
- Each subsequent weekday in an additional (-10 pts)
- Weekends and university holidays are free days

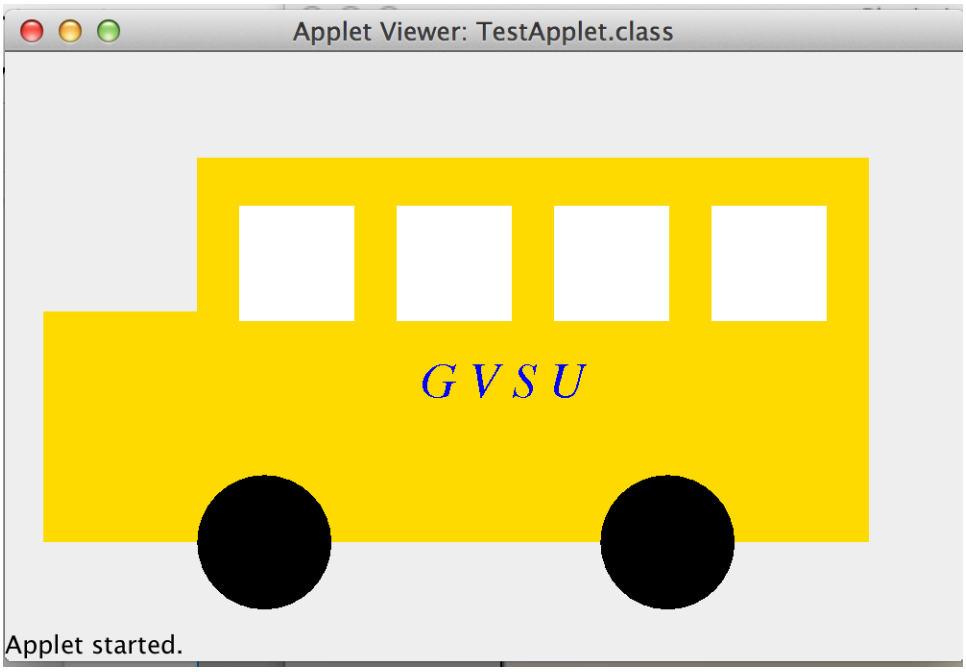
Turn in

A professional document that is stapled:

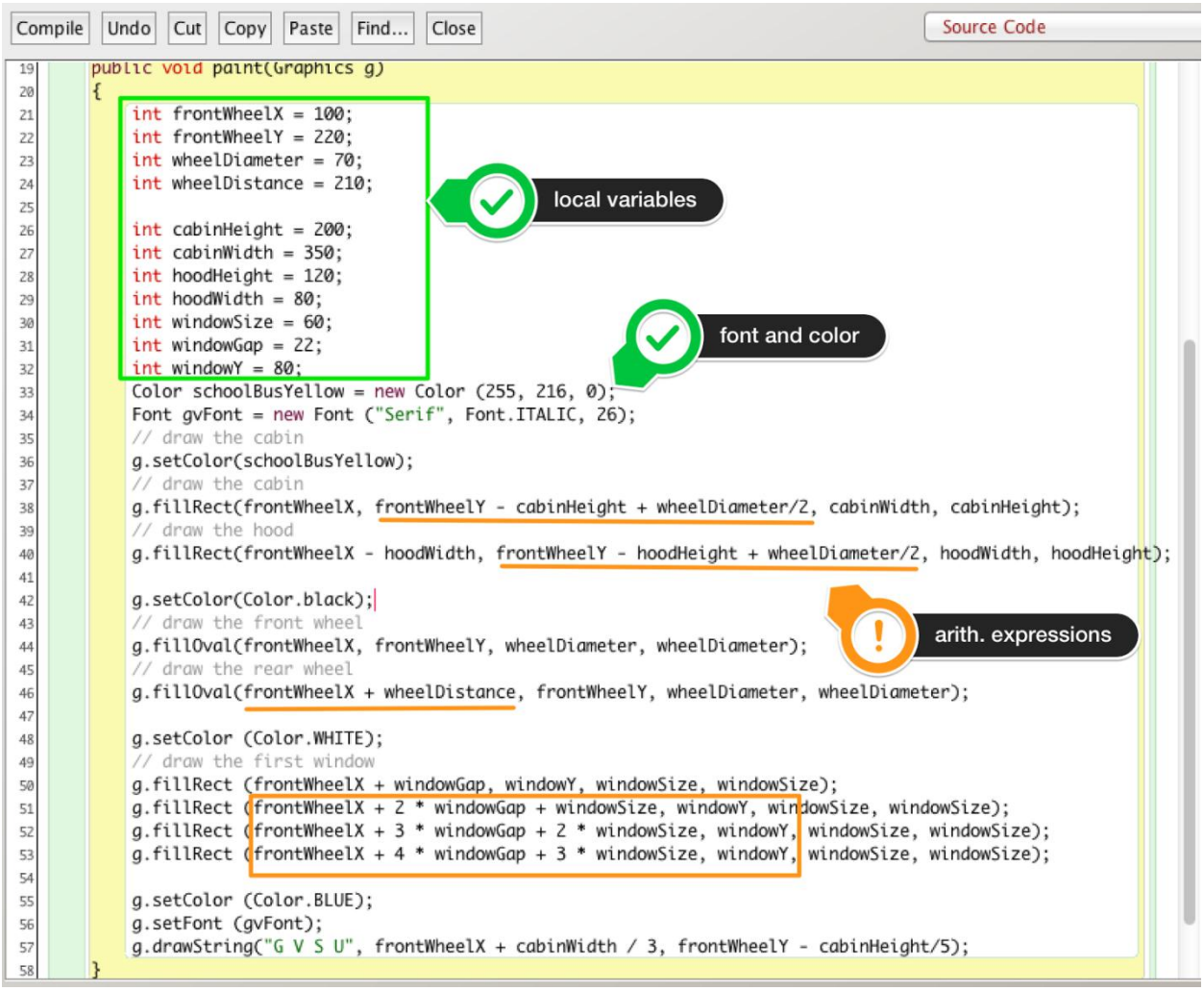
- Cover page that includes your name, a title, a signed pledge, time card, and a screenshot of the applet
- Signed pledge: "I pledge that this work is entirely mine, and mine alone (except for any code provided by my instructor)"
- Time Card a brief statement of how much time you spent on the project. For example "I spent 7 hours on this project from Jan 28 to Feb 3 reading book, designing a solution, writing code, fixing errors, and putting together the printed document"
- Your sketch – can be done by hand.
- Source code - a printout of your elegant source code (with your name)

Sample Instructor's Work (Sketch, Finished Applet, and Java Code)[1]





[1] Created by Professor Hans Dulimarta, School of CIS - GVSU.




```
public void paint(Graphics g)
{
    int frontWheelX = 100;
    int frontWheelY = 220;
    int wheelDiameter = 70;
    int wheelDistance = 210;

    int cabinHeight = 200;
    int cabinWidth = 350;
    int hoodHeight = 120;
    int hoodWidth = 80;
    int windowSize = 60;
    int windowGap = 22;
    int windowY = 80;
    Color schoolBusYellow = new Color (255, 216, 0);
    Font gvFont = new Font ("Serif", Font.ITALIC, 26);
    // draw the cabin
    g.setColor(schoolBusYellow);
    // draw the cabin
    g.fillRect(frontWheelX, frontWheelY - cabinHeight + wheelDiameter/2, cabinWidth, cabinHeight);
    // draw the hood
    g.fillRect(frontWheelX - hoodWidth, frontWheelY - hoodHeight + wheelDiameter/2, hoodWidth, hoodHeight);

    g.setColor(Color.black);
    // draw the front wheel
    g.fillOval(frontWheelX, frontWheelY, wheelDiameter, wheelDiameter);
    // draw the rear wheel
    g.fillOval(frontWheelX + wheelDistance, frontWheelY, wheelDiameter, wheelDiameter);

    g.setColor (Color.WHITE);
    // draw the first window
    g.fillRect (frontWheelX + windowGap, windowY, windowSize, windowSize);
    g.fillRect (frontWheelX + 2 * windowGap + windowSize, windowY, windowSize, windowSize);
    g.fillRect (frontWheelX + 3 * windowGap + 2 * windowSize, windowY, windowSize, windowSize);
    g.fillRect (frontWheelX + 4 * windowGap + 3 * windowSize, windowY, windowSize, windowSize);

    g.setColor (Color.BLUE);
    g.setFont (gvFont);
    g.drawString("G V S U", frontWheelX + cabinWidth / 3, frontWheelY - cabinHeight/5);
}
```