The objective is to start studying a Java application ("app") with some missing code and, in lab, write in the missing code. Also, demonstrate the application running on your own chromakeyed digital picture. Methods getPixel and setColor from G&E's API to change the color of one pixel are briefly explained and applied on p157-8, 197-8 of the text, available in the lab.

**IT steps:** Get this done as fast as possible, and keep notes about the underlined Unix commands because you will need to use similar ones in future labs and courses, and to know them for our exams.

1. (In HU-25) Log in as usual (see old lab sheets if necessary, or, better, ask someone.)
2. (In HU-25) Command: `cd CSI201` (which means change directory to CSI201)

   This assumes you already made your CSI201 dir. GET HELP RIGHT AWAY if you see any message. Only the prompt is printed again when a Unix shell command is successful.

3. (In HU-25) Command: `mkdir Lab5` (which means make directory Lab5)

   Outside of HU-25, use your favorite way to get into your CSI201 directory/folder and make a sub-directory/folder inside it named Lab5. In the library, stuff inside your S: drive is available both in the Unix systems we access in HU-25 and in all the library computers. DON'T USE a library C: drive or MyDocuments or Desktop because stuff there will just disappear.

4. (In HU-25) Command: `cd Lab5` (which means change directory to Lab5)


   There is an asterisk after the last slash, then a SPACE, and FINALLY a PERIOD. (This means copy all files from directory /usr/local/depts/cs/geintro/Lab5/ into the current working directory which step 4 set as Lab5. The asterisk means match everything.)

   Outside of HU-25, download the three files into your Lab5 directory/folder from the Web URL: [http://www.cs.albany.edu/~sdc/CSI201/Spr14/Lab5](http://www.cs.albany.edu/~sdc/CSI201/Spr14/Lab5)

   Copies of the .java files for this lab EditablePicture.java and Lab5App.java are printed on the attached sheet. (Do NOT copy/paste from the .pdf!! You'll get messed up chars.)

6. Start DrJava as usual, use the Open function to navigate to the Lab5 directory/folder where you copied the files of step 5, and open both. (In the Library, remember to set the Extra Classpath preference to locate the bookClasses dir/folder; you must do that every time.) Compile both files and Run Lab5App. GET HELP if this fails: they should work as given. TIP: When a program run "gets stuck", click the Reset item in DrJava's upper menu bar. We describe the functionality of this app, indicating which functional item doesn't function yet because the code is missing. Your job in lab is to write the missing code to make it fully functional. Functional items in the sequence that you (the user or tester) observes:

   1. G&E's Folder Chooser window pops up and the computer waits until you navigate to and pick a directory (aka folder). You should pick the directory in which you have copied your digital picture. (If you don't have your picture, use Prof. Chaiken's to get started.) After picking, that window vanishes.

   2. G&E's File Chooser window pops up showing the directory (folder) you had picked in step 1, and waits until you have picked a file. After picking, that window vanishes.

   3. G&E's Explorer window pops up showing the picture you picked in step 2. G&E programmed their Explorer so people can see the image magnified up to 5X, and can pick a Pixel either by point and click or by entering an x-y location. The picked Pixel is marked by a cross-hair, and its color is displayed both by a color square and by the color's three Red, Green and Blue numeric intensity values. (Try it! It's easier to understand by seeing it than by reading this.)
4. G&E's Color Chooser window pops up. When you choose a **Color**, that window vanishes.

5. Four lines of text are printed. One line includes a printed description of the **Color** you had chosen in step 4. The last line asks you to type in two coordinate values.

6. The computer waits for you to type two integers separated by spaces and then press enter. If you type stuff invalid as integers, the program crashes. (That's fine in an intro. course.)

7. A line of text is printed about being in a method call. That line includes the two numbers you had typed in step 6.

8. Another instance of G&E's File Chooser window pops up, showing (almost) the same picture as before. When you have correctly added the missing code and are running your recompiled app (but NOT BEFORE!) you will find that the **Pixel** located by the two coordinates you typed in step 6 has CHANGED COLOR to the **Color** you had picked in step 4.

**Your first Lab5 job:** Study and discuss (with TAs and/or classmates) the comments within **EditablePicture.java** and apply that knowledge to write under each **//Purpose:** comment the line of code whose purpose is described, so that the color change of step 8 above actually happens when you demonstrate the app. This is easy, only 3 lines. But you have to know what they are. BOTH write them by hand onto the attached sheet AND code them in DrJava. Get clues from p157-158 of the textbook, but it will not work to just copy from there because the names and contexts are different.

**Getting your own chromakey picture:** Do this sometime during lab and then try the lab on your own picture. It's more fun! (1) The TA will have a web browser open to the page of pictures. When only a few classmates are waiting to use it, take your turn. Simply type your two-letter and digits netid in the box WITHOUT PRESSING ENTER, find your picture below and finally click on the button just below it. (2) After step (1), go back to your desk, go into the shell window, and command (use lower case letters in your net id):

```
  cp /usr/local/depts/cs/geintro/Spring14Photos/yournetid.JPG
```

(Outside of lab: You can download your picture with your browser's Save As function from the page of pictures with the URL below. Use user name csi201 and get the password from Blackboard)


**Your second Lab5 job:** Write your name and your TA's name on the attached sheet of code. Encircle and clearly mark with 1-8 the lines of code that directly cause the computer to do EACH of the functions listed above which the user sees. Now, below are a few crucial computer operations that are completely invisible to users. Encircle and mark with A-D the lines of code that cause the computer to do EACH of these HIDDEN operations: Give that sheet of paper to your TA for credit.

A) Make the computer set up the variable named **myP** so that it can be used three more times to refer to the **EditablePicture** object for calling methods on that object. The technical term for "set up a variable" is "declare". Declaring also gives the variable its name and type.

B) CALL the method named **changeXYLocationsColor** on the **EditablePicture** referred to by (the value of) **myP**. In other words, DO that METHOD CALL.

C) Actually change or **set** to new values the **Color** information stored in computer memory for the **Pixel** whose color is changed.

D) Make the computer stop following the instructions coded in the body of the method named **changeXY LocationsColor** and **return** to following the instructions at the spot after the METHOD CALL of step B). (Yes, the question tells you the answer!)

**Your third Lab5 job:** Upload check: After testing that your code actually changes a **Pixel**'s color, upload the 2 **.java** files comprising the app plus the two corresponding **.class** files to Blackboard.
Your Name________________________________________________

Your TAs Name____________________________________________

Grading: Lines of code for all 8 visible functions 1-8 correctly labelled______________
Lines of code for all 4 invisible functions A-D correctly labelled___________

CSI201 course expectations: This is one of the very last "fill in the blanks" kind of
programming questions that will be asked in this course. CSI201 is NOT a fill-in-the-blanks
kind of course! The current project, future projects and future labs require writing code into
empty areas and blank .java files according to directions in English.

Example: Given what you learned in this lab, add to EditablePicture the method below
which changes to the Color given by the parameter cParam the 9 Pixels of this
Picture that include and surround the Pixel located by the parameters xParam and
yParam; and modify the app so it actually demonstrates your added work.

```java
public void change9XYLocationsColor(int xParam, int yParam, java.awt.Color cParam)
```

(hint: Code like xParam+1, yParam-1, xParam-1, etc. will make the computer to the right
math to compute the parameter values you need when calling getPixel.)

Lab5App.java is reproduced here and EditablePicture.java is reproduced on the reverse
of this sheet.

```java
public class Lab5App
{
    public static void main(String[]a)
    {
        FileChooser.pickMediaPath();
        String digPhotoFileName;
        digPhotoFileName = FileChooser.pickAFile();
        EditablePicture myP;
        myP = new EditablePicture( digPhotoFileName );
        myP.explore( );
        java.awt.Color cRef;
        cRef = ColorChooser.pickAColor();
        System.out.println("You just picked Color " + cRef);
        System.out.println("Please magnify your picture in the explorer window.");
        System.out.println("Then, decide which coordinates locate a spot on your nose.");
        System.out.println("Type in those coordinates now in x y order, separated by spaces.");
        java.util.Scanner sc = new java.util.Scanner(System.in);
        int xIn, yIn;
        xIn = sc.nextInt();
        yIn = sc.nextInt();
        myP.change9XYLocationsColor( xIn, yIn, cRef );
        myP.explore();
    }
}
```
public class EditablePicture extends Picture
{
    final boolean DEBUG = true;

    public EditablePicture(String filenameParam )
    {
        super(filenameParam);
    }

    public void changeXYLocationsColor( int xParam, int yParam, java.awt.Color cParam )
    {
        if( DEBUG )
        {
            System.out.println("In method call changeXYLocationsColor("
                        + xParam + "," + yParam + "," + cParam + ")");
        }
        //Here's some code to you'll need (NOT written in the right sequence!)
        //
        // this.getPixel(xParam, yParam)
        // (Here is a method call that returns a value that should be
        // copied with = into a declared variable.
        // The getPixel method belongs to the Picture on which the
        // method was called, so we refer to such a Picture by this.)
        //
        // Pixel pixToChangeRef (Here is a declaration of a variable.)
        //
        // pixToChangeRef.setColor( cParam ) (Here is a method call.)
        //
        // pixToChangeRef = ????
        // (= is Java's assignment or copy operator.)
        //
        //Write the one line of code to achieve each purpose already written.
        //Purpose: Set up a Pixel reference variable with a declaration.
        //
        //Purpose: Call the getPixel method to get a reference to the Pixel to change,
        // and assign or copy that reference into a variable.
        //
        //Purpose: Set the Color of the Pixel to change to the value of the Color parameter
        //
        //Purpose: Return control to the spot where the method call was. (Written for you.)
        return;
    }

    public static void main(String[]a)
    {
        throw new java.util.concurrent.RejectedExecutionException
            ("DONT Run EditablePicture because it is not an application container!");
    }
}