Homework 5: More Collage Fun

Due: 5:00pm, Friday, October 28th

Set Up

- Open your home directory (double click on your cs8fXX icon; 2\textsuperscript{nd} icon from the top) and create a new directory in your home directory called \texttt{hw5} (all lowercase).
- Copy the folder \texttt{bookClasses} from your Desktop into the \texttt{hw5} directory.
  - Right click on \texttt{bookClasses} > Copy > Right click on \texttt{hw5} > Paste Into Folder
- When you work in DrJava \textbf{be sure} you are editing files in the \texttt{hw5/bookClasses} directory.

1. Sub-pictures

For this part of the homework, you are going to write methods in the \texttt{Picture.java} class to deal with sub-pictures. Later in the second part of the homework, you will use these methods to create a new collage.

So, what is a sub-picture? A sub-picture is just a rectangular part of another picture. If you have a bigger picture say, 640x480 pixels, you can get a sub-picture by specifying it’s start coordinates and it’s dimensions.

For example, here is a bigger picture, "barbara.jpg" which is 222x294 pixels, and its sub-picture 100x120 pixels at coordinates (100, 150):

Part 1 of HW5 consists of adding two methods to the \texttt{Picture.java} class. These two methods are:

a) \texttt{getSubPicture (int startX, int startY, Picture srcPic)}: This method will "extract" a sub-Picture from location (startX, startY) of srcPic and put it into the calling object. The dimensions of the sub-Picture are the same as that of the calling object.

For this method to work, you should first create a new blank Picture of whatever size you want it to be, like you did for the collage in the previous homework. Then you should create a Picture object for your "source" picture, i.e. the picture from where the pixels would be extracted to get your sub-picture. Finally call \texttt{getSubPicture()} on this new picture. See the code below:

\begin{verbatim}
Picture subPicObj = new Picture (100, 200);  // This is your sub-picture
Picture srcPic = new Picture ("barbara.jpg");  // This is your source picture
subPicObj.getSubPicture(50, 100, srcPic);      // This creates the sub-Picture
\end{verbatim}
To create the Picture of your source picture, you don’t have to use the FileChooser.pickAFile() method. You may copy the images into your hw5/bookClasses folder instead. So if you copy the file barbara.jpg there, then to create a Picture object, you just need to use the path "barbara.jpg" rather than the full absolute path name, like: Picture srcPic = new Picture ("barbara.jpg"). That’s what is shown above. After creating, your sub-picture would look something like this:

![Sub-Picture Example](image1.png)

**b) addSubPicture(int startX, int startY, Picture srcSubPic):** This method will take a sub-Picture as an input and then place it on a bigger picture, starting from the coordinates mentioned by startX and startY. For this method to work, you need to specify the subPicture itself that will get placed (srcSubPic in the parameter list) and then where it will get placed in the bigger picture starting from (int startX and int startY). See the below code, for example:

```java
// Creates blank image, 300 pixels wide and 400 pixels high.
Picture destPic = new Picture (300, 400);

// Adds the sub-Picture created in the previous example code.
destPic.addSubPicture(100, 200, subPicObj);
```

After adding, the picture would look something like this:

![Bigger Picture with Sub-Picture](image2.png)
Notice how in the above picture the sub-picture is placed at the specified coordinates, i.e. (100, 200). In other words, we have successfully, called getSubPicture() to obtain a sub-Picture and then placed it into a new picture using addSubPicture().

Here are some important points to remember about part 1 of this homework:

- **Write your code so that you don’t have to call FileChooser.pickAFile() for any of your images. You can do this by copying the required images into hw5/bookClasses folder.**

- **Make sure your code takes care of bounds overflow for both the methods.**
  
  For example, due to the relative placement of the start of the sub-picture and its size you might encounter situations when accessing the source picture could cause an overflow. Note that this overflow is not only dependent on the size of the sub-picture but also at what (x,y) values you have passed to the getSubPicture() method. Check case 1 below.

  Similarly, you need to provide checks in your addSubPicture() method too so that trying to add new pixels onto the destination image does not cause an overflow. Check case 2 below.

  In other words, you need to make sure that whenever you call getSubPicture(), you don’t overflow the coordinates from the source image while **reading** the pixels. And when you call addSubPicture(), you don’t overflow into the destination image while **writing** the pixels. **Your homework will be hammered by the graders to make sure that you handle all your overflow scenarios. So pay proper attention!**

**Case 1: Overflow occurs while reading from the source picture**

- Region that should be visible in the sub-picture
- Region that should be truncated in the sub-picture
- Finally, this is how the extracted sub-picture should look like.

![Diagram](image-url)
Case 2: Overflow occurs when writing into the destination picture

Again, make sure you follow all style guidelines as in the previous homeworks. Those guidelines are not duplicated here but you still need to follow them. Look up the previous homework write-ups if you have to. Your whole homework should follow the guidelines, including the parts that were copied from the write-up.

Use a nested for loop to read your source pixels in both getSubPicture() and addSubPicture(). You need to figure out given (x,y) in the source picture, what could be the destination pixel in the target picture. For getSubPicture(), the target is the sub-picture image that you newly created. For addSubPicture() the target is the bigger (super-Picture?) that you newly created, where you will place your smaller sub-picture.

Make sure you call show() on the original picture, the sub-Picture and then the resulting picture after you add the sub-Picture. In total 3 pictures should be shown when you run this homework.

Add the getSubPicture() and addSubPicture() methods in the Picture.java file. As mentioned earlier, the getSubPicture() should have the following signature:

```java
public void getSubPicture(int startX, int startY, Picture srcPic)
```

startX and startY specify from where in the srcPic you need to start reading pixels for your sub-Picture. How many pixels you will read from there will be determined by the dimensions of your sub-Picture object.

Similarly, add addSubPicture() within Picture.java file, which should have the following signature:

```java
public void addSubPicture(int startX, int startY, Picture subPic)
```

startX and startY will specify where in the destination picture you need to start adding pixels from your sub-Picture. subPic will be sub-Picture object that you need to read from in order to populate your destination picture.

Here copying the pixels refers to copying the Color object of each source pixel onto the destination pixel. See the following code:

```java
Pixel srcPixel = somePicObj.getPixel(x,y);
Pixel destPixel = anotherPicObj.getPixel(i,j);
destPixel.setColor(srcPixel.getColor());
```

The above is a very general piece of code that simply reads the color from the pixel at x,y from somePicObj and then it writes that color into pixel i,j within anotherPicObj. In effect, you have transferred or copied the pixel from somePicObj into anotherPicObj. Your code for the addSubPicture()and getSubPicture() will have logic similar to the above.
• Use some image other than barbara.jpg for this homework.

• Start with the following template for your HW5A.java class. Use different values for the x,y values in your homework than what is shown below as an example.

File: HW5A.java

/*
 * Filename: HW5A.java
 * Name: Your name and your partner's name
 * Login: cs8fxx, cs8fxx
 * Date: Month, Day, Year
 * Sources of Help: ... (for example: names of people, books, websites, etc.)
 * 
 * Describe what the code does here.
 */

/*
 * Name: Method name
 * Purpose: Briefly describe the purpose of this method
 * Parameters: List all parameters and their types and what they represent.
 * If no parameters, just state None.
 * Return: Specify the return type and what it represents.
 * If no return value, just specify void.
 */

public class HW5A{
    public static void main (String[] args){
        Picture subPicObj = new Picture (100, 200);    // This is your sub-picture
        Picture srcPic = new Picture("barbara.jpg");  // This is your source picture
        srcPic.show();                    // Show the source picture
        subPicObj.getSubPicture(50, 100, srcPic);      // This creates the subpicture
        subPicObj.show();                             // This shows the subpicture

        Picture destPic = new Picture (300, 400);      // Creates blank image, 300 pixels wide and 400 pixels high
        destPic.addSubPicture(100, 200, subPicObj);    // Adds the sub-picture created in the previous example code
        destPic.show();                                // This shows the resulting picture
    }
}

2. Collage using sub-pictures
In this part of the homework, you are going to use the getSubPicture() and addSubPicture() methods above to create a collage of various sub-Pictures. You should choose the sub-picture coordinates appropriately so that the collage looks nice overall. You need to use at least 4 images to get your sub-pictures for the collage. You may use multiple sub-pictures from the same image in order to create your collage. Also, you must apply a filter to your sub-picture before you add it into the collage using addSubPicture(). You may use any of the filters that you created for your previous assignments, but you must add at least 2 new filters and these filters should use an if statement.

These 2 new filters cannot be from the book. This is your chance to showcase your creativity. Extra credit will be based on these 2 new filters. Also, there should be no white space left on the canvas after you are done with the collage.
Here are the steps you will take to accomplish what is said above:

- **Filtering**
  - Add at least **two new filter** methods to Picture.java that **do not** come from the textbook.
  - Your new methods must **use an if statement**. (Think of the cool things you can do with if statements!) You also must say in your comments for the method if your if statement has to do with the **location** (coordinates) of the pixel, the **color** of the pixel, or both.
  - Use at least **two other filters** that are **from chapter 5** in your textbook or can be filters that you wrote for Homework 4.
  - The methods that you use to filter your pictures can either change the calling object or they can take in a Picture and return the modified Picture.
  - You are free to use more pictures and more filters, just make sure it works and there is **no white space** anywhere on your final Picture.

- **Creating the Collage**
  - Creating the collage just involves calling addSubPicture() multiple times, at different locations in the collage picture, with different sub-pictures. Your sub-pictures might even overlap, and that is fine, as long as you cover your entire collage and there is no white space sticking out.
  - For this part, you should start with creating a blank picture of size 640x480.
  - Once the blank picture is created, create as many sub-pictures from the various images present in the mediasources folder or your own as you want. As mentioned earlier, copy these pictures onto your hw5/bookClasses folder for easy access within your code.
  - If you use the original picture (after applying a filter, of course), without getting a sub-picture out of it and then add sub-picture at (0,0) you would have added the original picture in its entirety onto the collage. This can be a nice trick to use if you want to add the complete image onto your collage rather than just a sub-picture of it.

```java
Picture srcPic = new Picture("barbara.jpg");
srcPic.grayscale();
Picture destPic = new Picture (300, 400);
destPic.addSubPicture(0, 0, srcPic);
destPic.show();
```

- You should refer to your HW4 and your textbook (page 143-144) for your loops.

**NOTE: Unlike your HW4, you should NOT choose pictures of all the same size and put them side-by-side.**
You can choose pictures that will make it easy to put them all on one canvas, but make it interesting. There are ways of writing your for loops to design some pretty interesting and good looking pictures. Some people have written for loops so that their pictures came out diagonal, upside down, or in interesting positions around the canvas.

This is your chance to **BE CREATIVE!!!**
In this file you will choose at least four pictures. So that your collage comes out right, you can hard code which pictures and then include them with your submission (see instructions below). If you'd like, you can design your program so that it accepts any picture, but this is not necessary, just something that might be fun to experiment with.

Play around with different coordinates and different sub-picture sizes so that the entire canvas is filled and no white space remains. Use pencil and paper in order to work out the exact placement and sizes of your sub-pictures.

IMPORTANT In order for the grader to grade your submission your code must work with different paths. To obtain each media file, do NOT use the file path or MediaPath functions. Copy the images you want to use for your final submission into the same folder as your code, i.e., inside your hw5/bookClasses folder. Then ensure that the file path is simply "imagename.jpg". For example, if you want to use pic1.jpg make sure pic1.jpg is in your hw5/bookClasses folder and then use,

```
Picture p = new Picture("pic1.jpg");
```

You CAN show the original pictures if you want, but they must also appear with a filter.

In the end, you should call the write() method to save your image. Use a /tmp path like

```
collagePic.write("/tmp/cs8fXX_hw5_collage.jpg");
```

Please see the following image for an example. This example uses only barbara.jpg run through the grayscale() filter and beach.jpg run through the negate() filter. This example also has a lot of white space. However, your homework should use at least 4 pictures with 4 different filters (2 of which should be your own filters and not found in the book) and should not have any white space left on the canvas. Working out the placement with pen and pencil would be a good idea as mentioned earlier.
EXTRA CREDIT:
Just Be Creative!!! You can earn up to 5 extra points for your creativity. You will be graded on the creativity of your filters and on the way you setup pictures on your collage.

How to turnin your homework electronically

Only ONE partner should run the turnin script - do not submit two copies of the same homework.

Make sure the program works correctly in your cs8fxx login on the workstations in the labs in the basement of the CSE building when the workstations are booted under Linux (CentOS).
When you are ready to turn your program in, first open a terminal window by right clicking on the empty area of your computer’s desktop and select "Open a new Terminal."

In addition - you can submit up to 5 pictures of your choosing. PLEASE DONOT SUBMIT MORE THAN 5 PICTURES!!

Make sure all the files that you want to submit are in your hw5/bookClasses folder.

Type in the following command

    turnin hw5

These files are required to be submitted for hw5:

- Picture.java
- HW5A.java
- HW5B.java
- Any additional (no more than 5) pictures you want to submit in the form: pic1.jpg, etc. (where pic1.jpg is the name of your file.)

If you don't have all of these files present or have misspelled the name of one of these files, the turnin script will complain!

You can verify your turnin with the following command

    verify hw5

It is your responsibility to make sure you properly turned in your assignment.

Start Early and Have Fun!