DATA STRUCTURES & ALGORITHMS

LECTURE 8

October 1

ANNOUNCEMENTS

TWO DIMENSIONS

ICLICKER QUESTION
ICLICKER QUESTION

Which nested loops will access the highlighted area?

A. for (int y = 0; y < 4; y++)
   for (int x = 0; x < 2; x++)
     // Access [x,y]

B. for (int x = 0; x < 4; x++)
   for (int y = 0; y < 2; y++)
     // Access [x,y]

C. for (int x = 0; x < 4; x++)
   for (int y = 0; y <= 2; y++)
     // Access [x,y]

D. None of the above

COPYING PIXELS TO A NEW PICTURE

- What if we wanted to copy (part of) one picture into another?
- How would we do that?

We would need to copy each pixel from the source to the target image.
To do this, we would need to track the source picture x and y
  - And the target picture x and y

We can use create a blank picture as the target picture.
COPY PICTURE ALGORITHM

• Copy a picture to a 960 x 720 blank picture
• Create the target picture object
• Invoke the method on the target picture
• Create the source picture object
• Loop through the source picture pixels
  • Get the source and target pixels
  • Set the color of the target pixel to the color of the source pixel

COPY ALGORITHM TO CODE

• Loop through the source pixels
  
  // loop through the columns
  for (int sourceX = 0, targetX = 0;
   sourceX < sourcePicture.getWidth();
   sourceX++, targetX++)
  {
    // loop through the rows
    for (int sourceY = 0, targetY = 0;
     sourceY < sourcePicture.getHeight();
     sourceY++, targetY++)
    {
  
COPY ALGORITHM TO CODE – CONT

• Get the source and target pixels

  sourcePixel = sourcePicture.getPixel(sourceX,sourceY);
  targetPixel = this.getPixel(targetX,targetY);

• Set the color of the target pixel to the color of the source pixel

  targetPixel.setColor(sourcePixel.getColor());

COPYING A SPECIFIC PICTURE

• We can use this technique to copy a specific picture.
• In this case the picture "piperdd.jpg"
public void copySpecific()
{
    Picture sourcePicture = new Picture("/..../piperdd.jpg");
    Pixel sourcePixel = null;
    Pixel targetPixel = null;

    // loop through the columns
    for (int sourceX = 0, targetX = 0;
     sourceX < sourcePicture.getWidth();
     sourceX++, targetX++)
    {
        // loop through the rows
        for (int sourceY = 0, targetY = 0;
         sourceY < sourcePicture.getHeight();
         sourceY++, targetY++)
        {
            // set the target pixel color to the source pixel color
            sourcePixel = sourcePicture.getPixel(sourceX,sourceY);
            targetPixel = this.getPixel(targetX,targetY);
            targetPixel.setColor(sourcePixel.getColor());
        }
    }
}
COPY TO AN UPPER LEFT LOCATION

• How would you copy a picture to a location in another picture (like 100, 100)?
  – Specified as the upper left corner
• You still copy all the source pixels
  – But the target x and y start at the specified location

COPY TO POSITION EXERCISE

• Write a method copyRobot to copy
  – robot.jpg
  – To location
    • 100, 100 in blank picture
• Test with
  Picture p = new Picture(640, 480);
  p.copyRobot();
  p.show();

COPY TO POSITION EXERCISE

• Hint – you have an offset in each direction:
  – offsetX – how far to the right you want to start
  – offsetY – how far down you want to start.
• When copying the pixels, you use the offsets to figure how far over and down to copy the pixels.

CROPPING

• We can copy just part of a picture to a new picture
  – Just change the start and end values for source x and y values to the desired range of pixels.
    • Instead of starting at (0,0) and going to (width-1,height-1).
  – Use pictureObj.explore() to find the x and y values.
  – What are the x and y values to get the face of the dog in piperdd.jpg?
CROPPING

• What are the x and y values to get the face of the dog in piperdd.jpg?

• What is the region?

```
public void copySpecificFace()
{
    Picture sourcePicture = new Picture("../piperdd.jpg");
    Pixel sourcePixel = null;
    Pixel targetPixel = null;

    // loop through the columns
    for(int sourceX = 190, targetX = 100; sourceX < 440; sourceX++, targetX++)
    {
        // loop through the rows
        for(int sourceY = 30, targetY = 100; sourceY < 240; sourceY++, targetY++)
        {
            // set the target pixel color to the source pixel color
            sourcePixel = sourcePicture.getPixel(sourceX,sourceY);
            targetPixel = this.getPixel(targetX,targetY);
            targetPixel.setColor(sourcePixel.getColor());
        }
    }
}
```

TESTING COPY SPECIFIC FACE

• Create a picture object
  - Picture p1 = new Picture(640, 480);

• Show the picture
  - p1.show();

• Invoke the method
  - p1.copySpecificFace();

• Repaint the picture
  - p1.repaint();

WHAT MAKES A GOOD METHOD?

• Ideally, a method should do one and only one thing.
  — That is, accomplish some task.
  — Also, the name should tell you what the method does.
WHAT MAKES A GOOD METHOD?

• A method can call other methods to do some of its work
  – This is known as **procedural decomposition**.
  – In essence, farming out out sub-problems for other methods to handle.
• For example, remember when we re-wrote the sunset method to call methods to
  – Decrease green
  – Decrease blue
  to get the (increased red) sunset effect.

WHAT MAKES A GOOD METHOD?

• We shouldn’t have the same code appearing in several methods.
  – Instead, put the code to do that task in its own method, and call that method when we need that code.
  – Like the methods to change the amount of red, green and blue.
    • Make three methods to do those color changes.
• We should make simple, general methods that are reusable.

WHAT MAKES A GOOD METHOD?

• A method should be in the class that has the data the method is working on.
  – That is methods that work on Picture objects go in Picture.java, *etc.*

ICLICKER QUESTION
ICLICKER QUESTION

Procedural decomposition is:
A. The breakdown of tasks into multiple methods
B. The method by which a Java .class file is created
C. A digital image effect of separating the color values
D. A guideline for indenting code
E. A haunted government office

WAS THE LAST METHOD GENERAL?

• No.
• We wrote in the name of a specific file to copy from in the method
• What if we want to copy from a different picture?
  – We would need to
    • change the method
  or
    • make another method

GENERAL COPY ALGORITHM

• Create a method that copies pixels from any source picture
  – Passing the picture as a parameter
  – Giving a start x and y and end x and y for the source picture
    • If the start x and y and end x and y cover the entire picture then the whole picture will be copied
    • If the start x and y and end x and y are part of the picture then cropping will occur
  – Copying to the current picture object with a target start x and target start y
    • If the start x and y are 0 then it copies to the upper left corner

GENERAL COPY ALGORITHM

• Loop through the x values between xStart and xEnd
• Loop through the y values between yStart and yEnd
• Get the pixel from the source picture for the current x and y values
  • Get the pixel from the target picture for the targetStartX + x and targetStartY + y values
  • Set the color in the target pixel to the color in the source pixel
```java
public void copy(Picture sourcePicture, int startX, int startY,
                   int endX, int endY, int targetStartX, int targetStartY)
{
    Pixel sourcePixel = null;
    Pixel targetPixel = null;
    // loop through the x values
    for(int x = startX, tx = targetStartX;
        x < endX;
        x++, tx++)
    {
        // loop through the y values
        for(int y = startY, ty = targetStartY;
            y < endY;
            y++, ty++)
        {
            // set the target pixel color to the source pixel color
            sourcePixel = sourcePicture.getPixel(x, y);
            targetPixel = this.getPixel(tx, ty);
            targetPixel.setColor(sourcePixel.getColor());
        }
    }
}
```

**Rewrite Methods Exercise**

- Type the copy method in `Picture.java`
- Rewrite `copySpecific()` and `copySpecificFace()` methods to use the new copy method
- Run the methods to make sure they still work
What will be output by the following code?

```java
int [][] someNums = {{1,2,3},{4,5,6},{7,8,9}};
int sum = 0;
for(int i = 0; i < 3; i++)
{
    sum += someNums[i][i];
}
System.out.println(sum);
```

A. 12  
B. 15  
C. 16  
D. 45