AP® Computer Science A Picture Lab Student Guide

The AP Program wishes to acknowledge and thank Barbara Ericson of the Georgia Institute of Technology, who developed this lab and the accompanying documentation.

A6: Mirroring pictures

Car designers at General Motors Research Labs only sculpt half of a car out of clay and then use a vertical mirror to reflect that half to see the whole car. What if we want to see what a picture would look like if we placed a mirror on a vertical line in the center of the width of the picture to reflect the left side (Figure 6)?

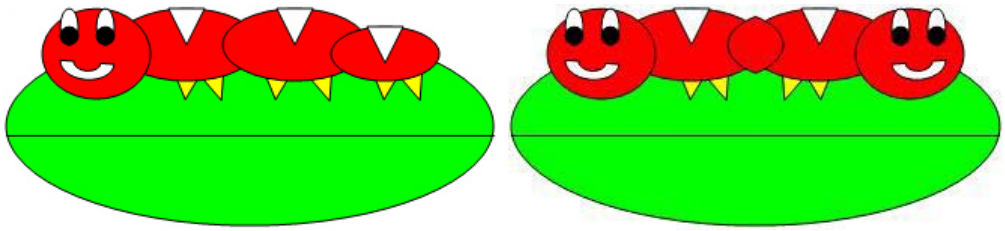

Figure 6: Original picture (left) and picture after mirroring (right)

How can we write a method to mirror a picture in this way? One way to figure out the *algorithm,* which is a description of the steps for solving a problem, is to try it on smaller and simpler data. Figure 7 shows the result of mirroring a two-dimensional array of numbers from left to right vertically.

	0	1	2	3	4
0	1	2	3	4	5
1	6	7	8	9	10
2	11	12	13	14	15

1	2	3	2	1
6	7	8	7	6
11	12	13	12	11

Figure 7: Two-Dimensional array of numbers (left) and mirrored result (right)

Can you figure out the *algorithm* for this process? Test your algorithm on different sizes of two-dimensional arrays of integers. Will it work for 2D arrays with an odd number of columns? Will it work for 2D arrays with an even number of columns?

One algorithm is to loop through all the rows and half the columns. You need to get a pixel from the left side of the picture and a pixel from the right side of the picture, which is the same distance from the right end as the left pixel is from the left end. Set the color of the right pixel to the color of the left pixel. The column number at the right end is the number of columns, also known as the width, minus one. So assuming there are at least 3 pixels in a row, the first left pixel will be at row=0, col=0 and the first right pixel will be at row=0, col=width-1. The second left pixel will be at row=0, col=1 and the corresponding right pixel will be at row=0, col=width-1-1. The third left pixel will be at row=0, col=2 and its right pixel will be at row=0, col=width-1-2. Each time the left pixel is at (current row value, current column value), the corresponding right pixel is at (current row value, width - 1 - (current column value)).

The following method implements this algorithm. Note that, because the method is not looping through all the pixels, it cannot use a nested `for-each` loop.

```
public void mirrorVertical()
{
  Pixel[][] pixels = this.getPixels2D();
  Pixel leftPixel = null;
  Pixel rightPixel = null;
  int width = pixels[0].length;
  for (int row = 0; row < pixels.length; row++)
  {
    for (int col = 0; col < width / 2; col++)
    {
      leftPixel = pixels[row][col];
      rightPixel = pixels[row][width - 1 - col];
      rightPixel.setColor(leftPixel.getColor());
    }
  }
}
```

You can test this with the `testMirrorVertical` method in `PictureTester`.

Exercises

1. Write the method `mirrorVerticalRightToLeft` that mirrors a picture around a mirror placed vertically from right to left. Hint: you can copy the body of `mirrorVertical` and only change one line in the body of the method to accomplish this. Write a class (static) test method called `testMirrorVerticalRightToLeft` in `PictureTester` to test this new method and call it in the `main` method.
2. Write the method `mirrorHorizontal` that mirrors a picture around a mirror placed horizontally at the middle of the height of the picture. Mirror from top to bottom as shown in the pictures below (Figure 8). Write a class (static) test method in `PictureTester` to test this new method and call it in the `main` method.

Figure 8: Original picture (left) and mirrored from top to bottom (right)

3. Write the method `mirrorHorizontalBotToTop` that mirrors the picture around a mirror placed horizontally from bottom to top. Hint: you can copy the body of `mirrorHorizontal` and only change one line to accomplish this. Write a class (static) test method in `PictureTester` to test this new method and call it in the `main` method.
4. Challenge — Work in groups to figure out the algorithm for the method `mirrorDiagonal` that mirrors just a square part of the picture from bottom left to top right around a mirror placed on the diagonal line (the diagonal line is the one where the row index equals the column index). This will copy the triangular area to the left and below the diagonal line as shown below. This is like folding a square piece of paper from the bottom left to the top right, painting just the bottom left triangle and then (while the paint is still wet) folding the paper up to the top right again. The paint would be copied from the bottom left to the top right as shown in the pictures below (Figure 9). Write a class (static) test method in `PictureTester` to test this new method and call it in the `main` method.

Figure 9: Original picture (left) and mirrored around the diagonal line with copying from bottom left to top right (right)

How image processing is related to new scientific breakthroughs

Many of today's important scientific breakthroughs are being made by large, interdisciplinary collaborations of scientists working in geographically widely distributed locations, producing, collecting, and analyzing vast and complex datasets.

One of the computer scientists who works on a large interdisciplinary scientific team is Dr. Cecilia Aragon. She is an associate professor in the Department of Human Centered Design & Engineering and the eScience Institute at the University of Washington, where she directs the Scientific Collaboration and Creativity Lab. Previously, she was a computer scientist in the Computational Research Division at Lawrence Berkeley National Laboratory for six years, after earning her Ph.D. in Computer Science from UC Berkeley in 2004. She earned her B.S. in mathematics from the California Institute of Technology.

Her current research focuses on human-computer interaction (HCI) and computer-supported cooperative work (CSCW) in scientific collaborations, distributed creativity, information visualization, and the visual understanding of very large data sets. She is interested in how social media and new methods of computer-mediated communication are changing scientific practice. She has developed novel visual interfaces for collaborative exploration of very large scientific data sets, and has authored or co-authored many papers in the areas of computer-supported cooperative work, human-computer interaction, visualization, visual analytics, image processing, machine learning, cyberinfrastructure, and astrophysics.

In 2008, she received the Presidential Early Career Award for Scientists and Engineers (PECASE) for her work in collaborative data-intensive science. Her research has been recognized with four Best Paper awards since 2004, and she was named one of the Top 25 Women of 2009 by Hispanic Business Magazine. She was the architect of the Sunfall data visualization and workflow management system for the Nearby Supernova Factory, which helped advance the study of supernovae in order to reduce the statistical uncertainties on key cosmological parameters that categorize dark energy, one of the grand challenges in physics today.

Cecilia Aragon is also one of the most skilled aerobatic pilots flying today. A two-time member of the U.S. Aerobatic Team, she was a medalist at the 1993 U.S. National Championships and the 1994 World Aerobatic Championships, and was the California State Aerobatic Champion.

Glossary

1. Abstract class — You cannot create an object of an abstract class type. But, you can create an object of a subclass of an abstract class (as long as the subclass is not also an abstract class).
2. Abstract method — An abstract method cannot have a method body in the class where the method is declared to be abstract.
3. Algorithm — A step-by-step description of how to solve a problem.
4. AWT — The Abstract Windowing Toolkit. It is the package that contains the original Graphical User Interface (GUI) classes developed for Java.
5. Binary number — A binary number contains only the digits 0 and 1. Each place is a power of 2 starting with 2^0 on the right. The decimal number 6 would be 110 in binary. That would be $0 * 2^0 + 1 * 2^1 + 1 * 2^2 = 6$.
6. Bit — A **b**inary dig**it**, which means that it has a value of either 0 or 1.
7. Byte — A consecutive group of 8 bits.
8. Column-major order — An order for storing two-dimensional array data in a one-dimensional array, so that all the data for the first column is stored before all the data for the second column and so on. In a two-dimensional array represented using an array of arrays (like in Java) this means that the outer array represents the columns and the inner arrays represent the rows.
9. Digital camera — A camera that can take digital pictures.
10. Digital picture — A picture that can be stored on a computer.
11. Inheritance — In Java, a class can specify the parent class from which it inherits instance variables (object fields) and object methods. Even though instance variables may be inherited, if they are declared to be private they cannot be directly accessed using dot notation in the inheriting class. Private methods that are inherited can also not be directly called in an inheriting class.
12. Inner loop — In a nested loop (a loop inside of another loop) the loop that is inside of another loop is considered the inner loop.
13. Interface — A special type of class that can only have public abstract methods in it and/or static constants.
14. Lossy compression — Lossy compression means that the amount of data that is stored is much smaller than the available data, but the part that is not stored is data that humans would not miss.
15. Media computation — A method of teaching programming by having students write programs that manipulate media: pictures, sounds, text, movies. This approach was developed by Dr. Mark Guzdial at Georgia Tech.
16. Megapixel — One million pixels.
17. Nested loop — One loop inside of another loop.
18. Outer loop — In a nested loop (a loop inside of another loop) the loop that is outside of another loop is considered the outer loop.
19. Package — A package in Java is a group of related classes.
20. Pixel — A picture (abbreviated **pix**) **el**ement.
21. RGB model — Represents color as amounts of red, green, and blue light. It sometimes also includes alpha, which is the amount of transparency.

22. Row-major order — An order for storing two-dimensional array data in a one-dimensional array, so that all the data for the first row is stored before all the data for the second row, and so on. In a two-dimensional array represented using an array of arrays (like in Java) this means that the outer array represents the rows and the inner arrays represent the columns.
23. Subclass — A class that has inherited from another class.
24. Superclass — A class that another class has inherited from.
25. UML —Unified Modeling Language. It is a general purpose modeling language used in object-oriented software development.

References

Dann, W., Cooper, S., & Ericson, B. (2009) *Exploring Wonderland: Java Programming Using Alice and Media Computation*. Englewood, NJ: Prentice-Hall.

Guzdial, M., & Ericson B. (2006) *Introduction to Computing and Programming in Java: A Multimedia Approach*. Englewood, NJ: Prentice-Hall.

Guzdial, M., & Ericson, B. (2009) *Introduction to Computing and Programming in Python: A Multimedia Approach*. (2nd ed.). Englewood, NJ: Prentice-Hall.

Guzdial, M., & Ericson, B. (2010) *Problem Solving with Data Structures using Java: A Multimedia Approach*. Englewood, NJ: Prentice-Hall.

Quick Reference

DigitalPicture Interface

```
Pixel[][] getPixels2D()         // implemented in SimplePicture
void explore()                  // implemented in SimplePicture
boolean write(String fileName)  // implemented in SimplePicture
```

SimplePicture Class (implements Digital Picture)

```
public SimplePicture()
public SimplePicture(int width, int height)
public SimplePicture(SimplePicture copyPicture)
public SimplePicture(String fileName)
public Pixel[][] getPixels2D()
public void explore()
public boolean write(String fileName)
```

Picture Class (extends SimplePicture)

```
public Picture()
public Picture(int height, int width)
public Picture(Picture copyPicture)
public Picture(String fileName)
public Pixel[][] getPixels2D()         // from SimplePicture
public void explore()                  // from SimplePicture
public boolean write(String fileName)  // from SimplePicture
```

Pixel Class

```
public double colorDistance(Color testColor)
public double getAverage()
public int getRed()
public int getGreen()
public int getBlue()
public Color getColor()
public int getRow()
public int getCol()
public void setRed(int value)
public void setGreen(int value)
public void setBlue(int value)
public void setColor(Color newColor)
```

java.awt.Color Class

```
public Color(int r, int g, int b)
public int getRed()
public int getGreen()
public int getBlue()
```