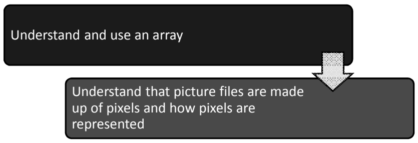
Pictures and Arrays



Understand and use an array

Understand that picture files are made up of pixels and how pixels are represented

Digital Pictures

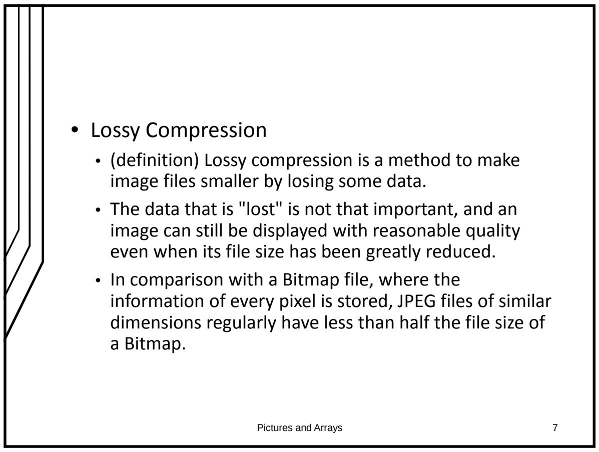
- pixel
  - (definition) A pixel (or picture element) is a single point in an image and represents the smallest element of an image that can be manipulated. Think of them as the atoms making up the image.
- Represented by pixels
  - With a red, green, and blue value stored for each pixel

- An image may be made up of thousands or even millions of pixels.
- New technical advances frequently lead to increases in the amount of storage and speed of computers.
- The total amount of media online is growing at a fast pace and making pictures smaller can make them faster to download and easier to store.
- As a result, digital pictures are commonly compressed when they are stored.

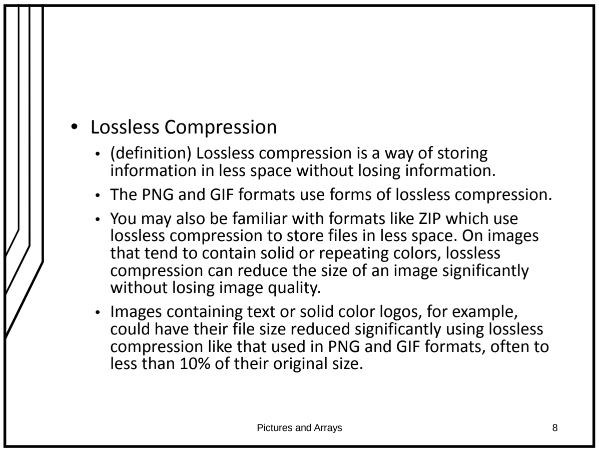
Types of images

- JPEG
  - (definition) JPEG (Joint Photographic Expert Group) is a commonly used format for encoding digital images. JPEG files can be adjusted for either quality or space saving.
  - The most popular file format for distributing images over the Internet
  - International standard
  - With lossy compression

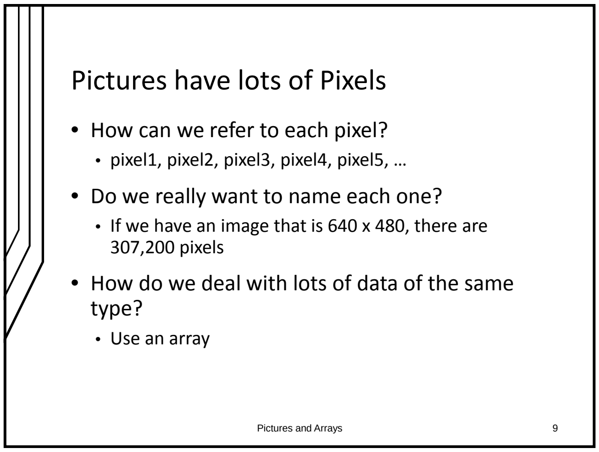
- PNG
  - (definition) PNG (Portable Network Graphics) is another popular image format for storing images using lossless compression, meaning the image information is not lost. Usually PNG is used in situations in which we care less about image size, or our image contains lots of the same color information.



- Lossy Compression
  - (definition) Lossy compression is a method to make image files smaller by losing some data.
  - The data that is "lost" is not that important, and an image can still be displayed with reasonable quality even when its file size has been greatly reduced.
  - In comparison with a Bitmap file, where the information of every pixel is stored, JPEG files of similar dimensions regularly have less than half the file size of a Bitmap.

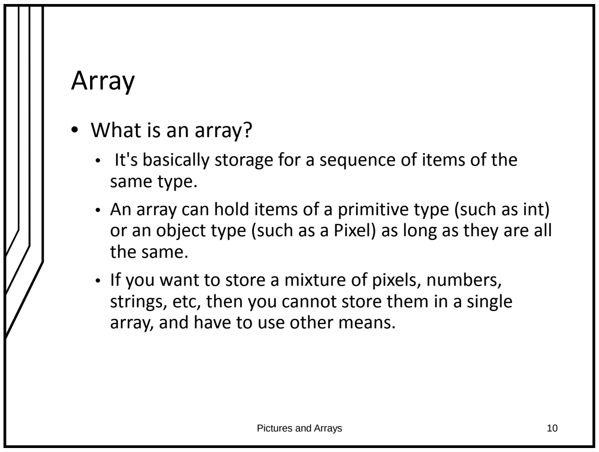


- Lossless Compression
  - (definition) Lossless compression is a way of storing information in less space without losing information.
  - The PNG and GIF formats use forms of lossless compression.
  - You may also be familiar with formats like ZIP which use lossless compression to store files in less space. On images that tend to contain solid or repeating colors, lossless compression can reduce the size of an image significantly without losing image quality.
  - Images containing text or solid color logos, for example, could have their file size reduced significantly using lossless compression like that used in PNG and GIF formats, often to less than 10% of their original size.



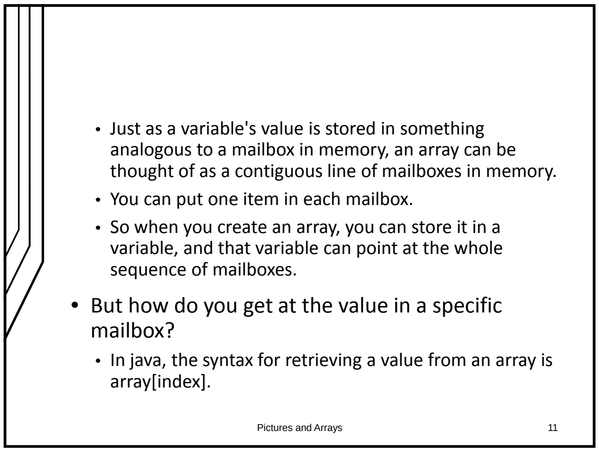
Pictures have lots of Pixels

- How can we refer to each pixel?
  - pixel1, pixel2, pixel3, pixel4, pixel5, …
- Do we really want to name each one?
  - If we have an image that is 640 x 480, there are 307,200 pixels
- How do we deal with lots of data of the same type?
  - Use an array



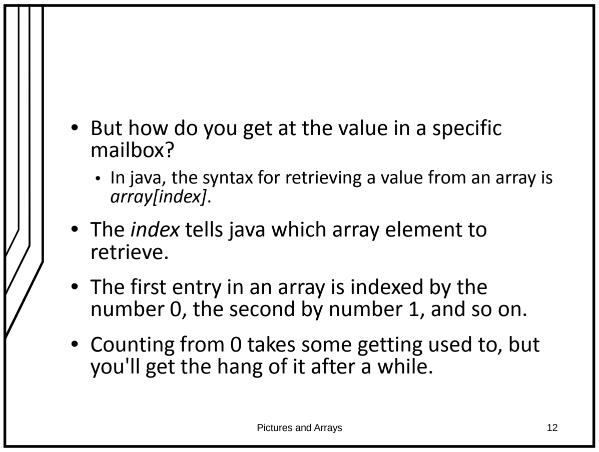
Array

- What is an array?
  - It's basically storage for a sequence of items of the same type.
  - An array can hold items of a primitive type (such as int) or an object type (such as a Pixel) as long as they are all the same.
  - If you want to store a mixture of pixels, numbers, strings, etc, then you cannot store them in a single array, and have to use other means.



- Just as a variable's value is stored in something analogous to a mailbox in memory, an array can be thought of as a contiguous line of mailboxes in memory.
- You can put one item in each mailbox.
- So when you create an array, you can store it in a variable, and that variable can point at the whole sequence of mailboxes.

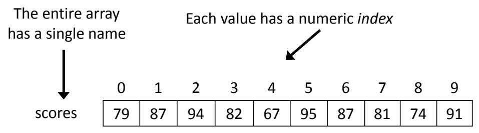
- But how do you get at the value in a specific mailbox?
  - In java, the syntax for retrieving a value from an array is array[index].



- But how do you get at the value in a specific mailbox?
  - In java, the syntax for retrieving a value from an array is *array[index]*.
- The *index* tells java which array element to retrieve.
- The first entry in an array is indexed by the number 0, the second by number 1, and so on.
- Counting from 0 takes some getting used to, but you'll get the hang of it after a while.

Arrays

- An array is an ordered list of values



The entire array has a single name

Each value has a numeric *index*

	0	1	2	3	4	5	6	7	8	9
scores	79	87	94	82	67	95	87	81	74	91

- An array of size N is indexed from zero to N-1
- This array holds 10 values that are indexed from 0 to 9

0	1	2	3	4	5	6	7	8	9
79	87	94	82	67	95	87	81	74	91

Arrays

- A particular value in an array is referenced using the array name followed by the index in brackets
- For example, the expression

  scores[2] refers to the value 94 (which is the 3rd value in the array)

- That expression represents a place to store a single integer, and can be used wherever an integer variable can.
- For example, it can be assigned a value, printed, or used in a calculation.

Declaring Arrays

- The scores array can be declared as follows:

  int[ ] scores = new int[10];

- Note that the type of the array does not specify its size, but each object of that type has a specific size
- The type of the variable scores is int[] (an array of integers)
- It is set to a new array object that can hold 10 integers

Declaring Arrays

Some examples of array declarations:

```
float[] prices = new float[500];

boolean[] flags;
flags = new boolean[20];

int size = 50;
char[] codes = new char[size];
```

Bounds Checking

- Once an array is created, it has a fixed size
- An index used in an array reference must specify a valid element
- That is, the index value must be in bounds (0 to N-1)
- The Java interpreter will throw an exception if an array index is out of bounds
- This is called automatic bounds checking

Bounds Checking

- If the array can hold 100 values, it can only be indexed using the numbers 0 to 99.
- If `count` has the value 100, then the reference will cause an `ArrayOutOfBoundsException`:

```
System.out.println (codes[count]);
```

- It's common to introduce *off-by-one* errors when using arrays

```
for (int index=0; index <= 100; index++)
      codes[index] = index*50;
```

Bounds Checking

- Each array object has a public constant called `length` that stores the size of the array
- It is referenced using the array name (just like any other object):

```
scores.length
```

- Note that length holds the number of elements, not the largest index

Array Declarations Revisited

- The brackets of the array type can be associated with the element type or with the name of the array
- Therefore the following declarations are equivalent:

```
float[] prices;

float prices[];
```

- The first format is generally more readable

Initializer Lists

- An initializer list can be used to instantiate and initialize an array in one step
- The values are delimited by braces and separated by commas
- Examples:

```
int[] units = {147, 323, 89, 933,
               540, 269, 97, 114,
               298, 476};

char[] letterGrades = {'A', 'B', 'C',
                       'D', 'F'};
```

Initializer Lists

- Note that when an *initializer* list is used:
  - the `new` operator is not used
  - no size value is specified
- The size of the array is determined by the number of items in the initializer list
- An initializer list can only be used in the declaration of an array

Arrays as Parameters

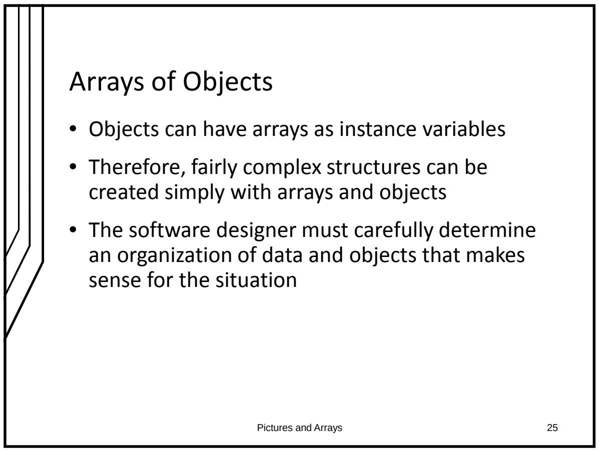
- An entire array can be passed to a method as a parameter
- Like any other object, the reference to the array is passed, making the formal and actual parameters aliases of each other
- Changing an array element in the method changes the original
- An array element can be passed to a method as well, and will follow the parameter passing rules of that element's type

Arrays of Objects

- The elements of an array can be object references
- The following declaration reserves space to store 25 references to `String` object

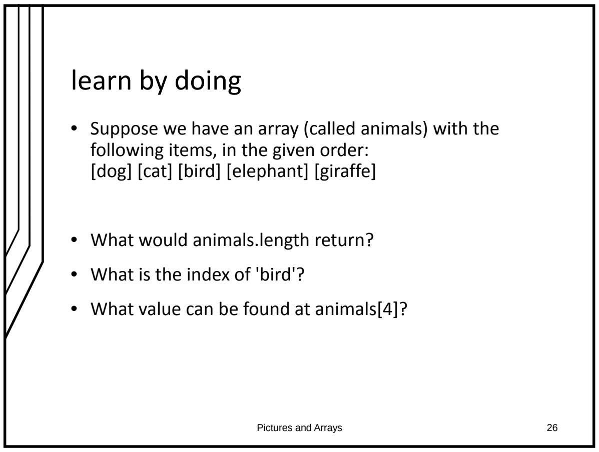
```
String[] words = new String[25];
```

- It does NOT create the `String` objects themselves
- Each object stored in an array must be instantiated separately



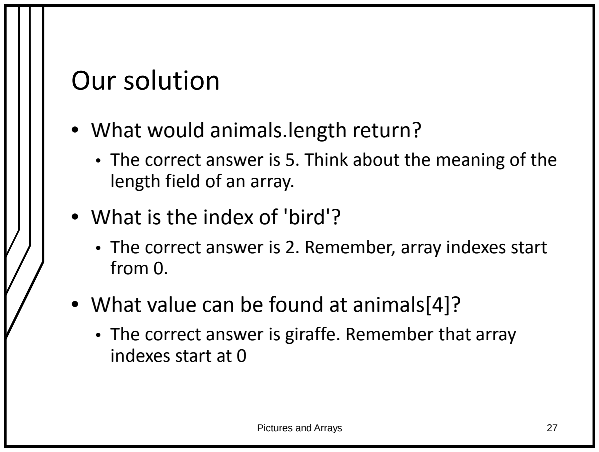
Arrays of Objects

- Objects can have arrays as instance variables
- Therefore, fairly complex structures can be created simply with arrays and objects
- The software designer must carefully determine an organization of data and objects that makes sense for the situation



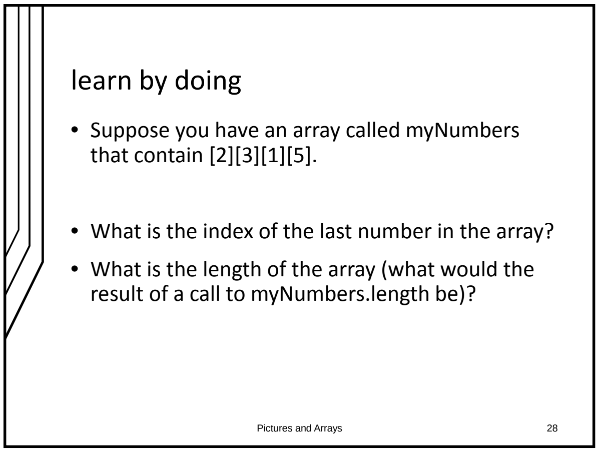
learn by doing

- Suppose we have an array (called animals) with the following items, in the given order:
  [dog] [cat] [bird] [elephant] [giraffe]
- What would animals.length return?
- What is the index of 'bird'?
- What value can be found at animals[4]?



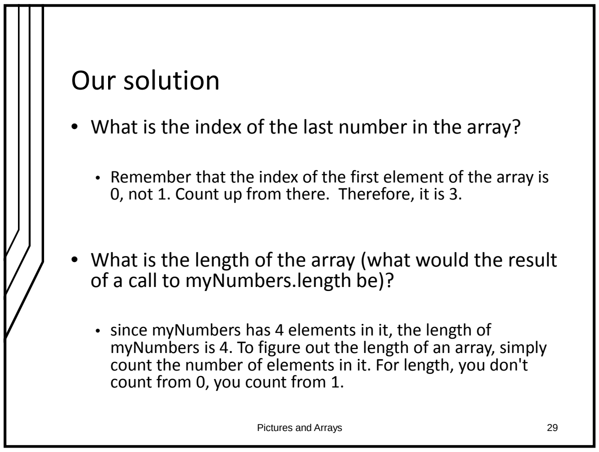
Our solution

- What would animals.length return?
  - The correct answer is 5. Think about the meaning of the length field of an array.
- What is the index of 'bird'?
  - The correct answer is 2. Remember, array indexes start from 0.
- What value can be found at animals[4]?
  - The correct answer is giraffe. Remember that array indexes start at 0



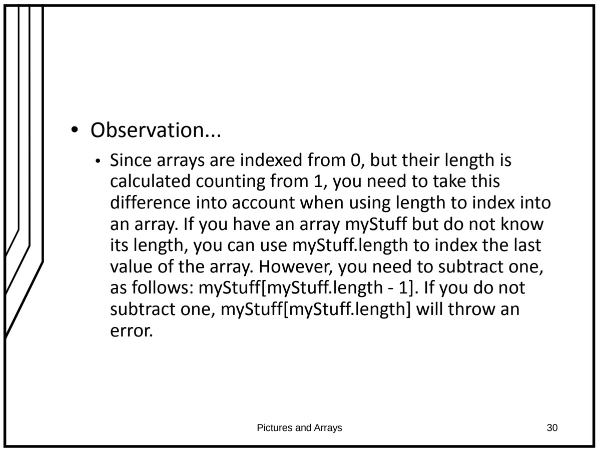
learn by doing

- Suppose you have an array called myNumbers that contain [2][3][1][5].
- What is the index of the last number in the array?
- What is the length of the array (what would the result of a call to myNumbers.length be)?

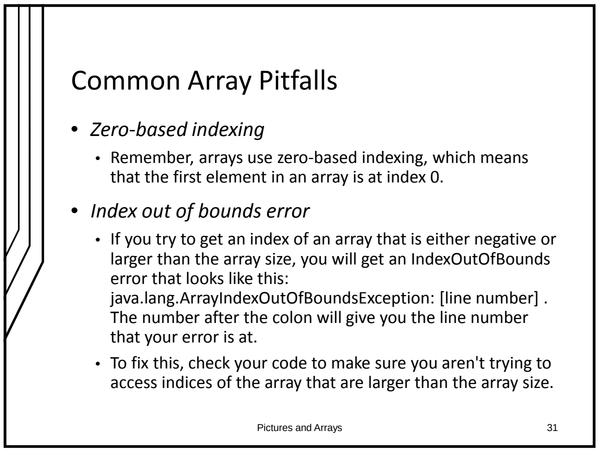


Our solution

- What is the index of the last number in the array?
  - Remember that the index of the first element of the array is 0, not 1. Count up from there. Therefore, it is 3.
- What is the length of the array (what would the result of a call to myNumbers.length be)?
  - since myNumbers has 4 elements in it, the length of myNumbers is 4. To figure out the length of an array, simply count the number of elements in it. For length, you don't count from 0, you count from 1.

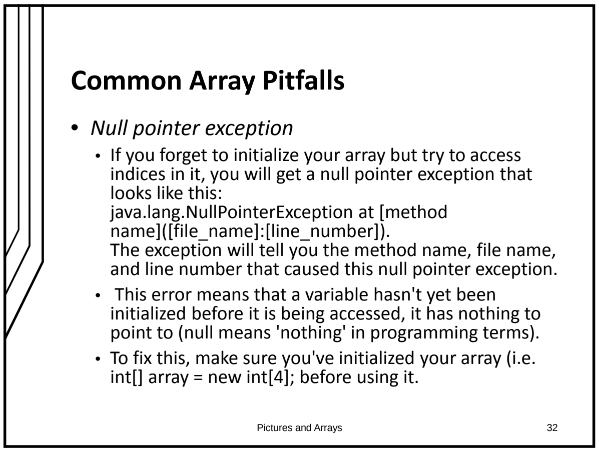


- Observation...
  - Since arrays are indexed from 0, but their length is calculated counting from 1, you need to take this difference into account when using length to index into an array. If you have an array myStuff but do not know its length, you can use myStuff.length to index the last value of the array. However, you need to subtract one, as follows: myStuff[myStuff.length - 1]. If you do not subtract one, myStuff[myStuff.length] will throw an error.



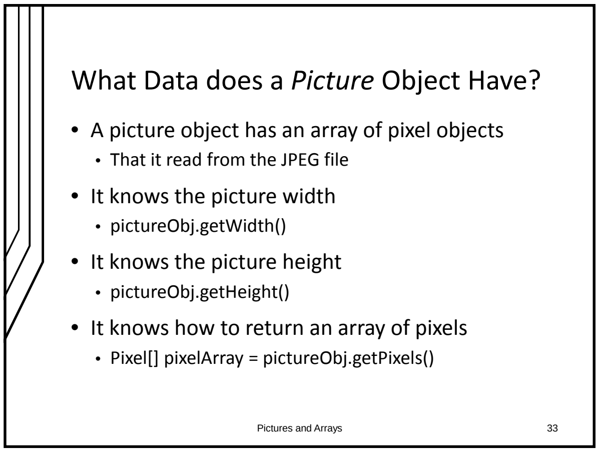
Common Array Pitfalls

- *Zero-based indexing*
  - Remember, arrays use zero-based indexing, which means that the first element in an array is at index 0.
- *Index out of bounds error*
  - If you try to get an index of an array that is either negative or larger than the array size, you will get an IndexOutOfBounds error that looks like this: java.lang.ArrayIndexOutOfBoundsException: [line number] . The number after the colon will give you the line number that your error is at.
  - To fix this, check your code to make sure you aren't trying to access indices of the array that are larger than the array size.



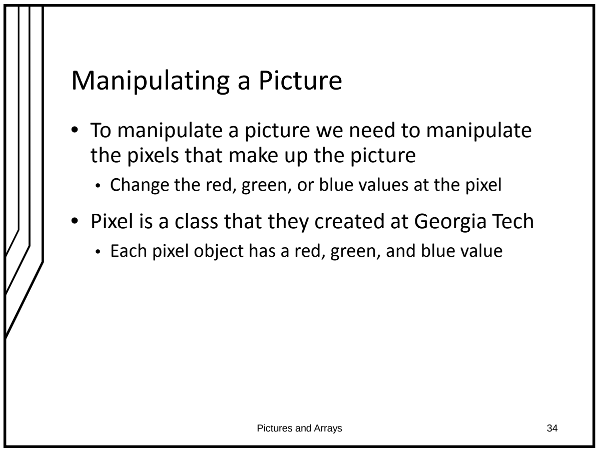
Common Array Pitfalls

- *Null pointer exception*
  - If you forget to initialize your array but try to access indices in it, you will get a null pointer exception that looks like this: java.lang.NullPointerException at [method name]([file_name]:[line_number]). The exception will tell you the method name, file name, and line number that caused this null pointer exception.
  - This error means that a variable hasn't yet been initialized before it is being accessed, it has nothing to point to (null means 'nothing' in programming terms).
  - To fix this, make sure you've initialized your array (i.e. int[] array = new int[4]; before using it.



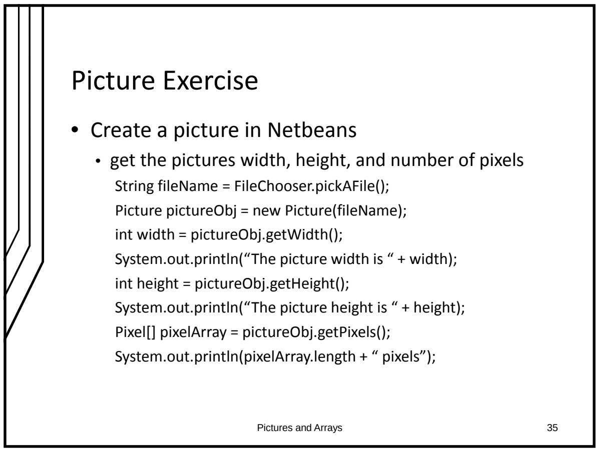
What Data does a *Picture* Object Have?

- A picture object has an array of pixel objects
  - That it read from the JPEG file
- It knows the picture width
  - pictureObj.getWidth()
- It knows the picture height
  - pictureObj.getHeight()
- It knows how to return an array of pixels
  - Pixel[] pixelArray = pictureObj.getPixels()



Manipulating a Picture

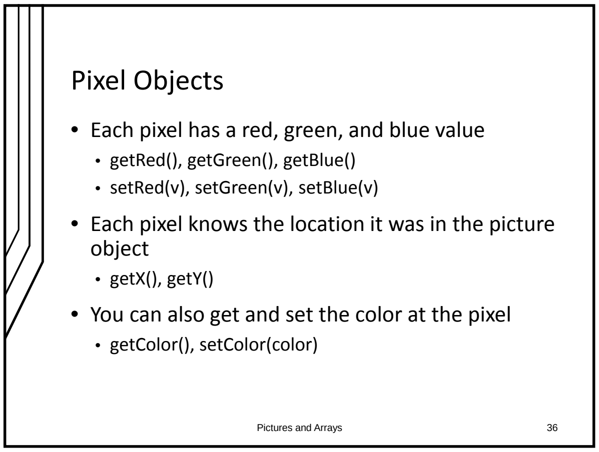
- To manipulate a picture we need to manipulate the pixels that make up the picture
  - Change the red, green, or blue values at the pixel
- Pixel is a class that they created at Georgia Tech
  - Each pixel object has a red, green, and blue value



Picture Exercise

- Create a picture in Netbeans
  - get the pictures width, height, and number of pixels

```
String fileName = FileChooser.pickAFile();
Picture pictureObj = new Picture(fileName);
int width = pictureObj.getWidth();
System.out.println(“The picture width is “ + width);
int height = pictureObj.getHeight();
System.out.println(“The picture height is “ + height);
Pixel[] pixelArray = pictureObj.getPixels();
System.out.println(pixelArray.length + “ pixels”);
```



Pixel Objects

- Each pixel has a red, green, and blue value
  - getRed(), getGreen(), getBlue()
  - setRed(v), setGreen(v), setBlue(v)
- Each pixel knows the location it was in the picture object
  - getX(), getY()
- You can also get and set the color at the pixel
  - getColor(), setColor(color)

Colours and the Pixel Class

- Instead of setting the red, green and blue values individually, the Pixel class allows you to use a Color object to represent the current color values of pixel objects.
- This Color object is found in the java.awt package.
- In order to use it, you should:

  import java.awt.Color;

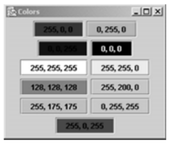
- so that java will know about the Color object.

Color Objects

- There is a class defined in Java that represents color
  - The Color class in the package java.awt
  - To use the class you must either
    - import java.awt.Color;
    - Use the full name java.awt.Color
- You can create a color object by giving the red, green, and blue values for it
  - Color colorObj = new Color(255,10,125);

Predefined Colors

- The Color class has defined class constants for many colors
  - Color.red, Color.green, Color.blue, Color.black, Color.white, Color.yellow, Color.gray, Color.orange, Color.pink, Color.cyan, Color.magenta
  - Or you can use all uppercase names
    - Color.RED, Color.BLUE, Color.BLACK, …



Getting and Setting Pixel Colors

- To get a pixel's color as a color object

  Color color1 = *pixelObj*.getColor();

  int red = color1.getRed();

  int green = color1.getGreen();

  int blue = color1.getBlue();

- To set a pixel's color using a new color object

  red = 20;

  green = 30;

  blue = 100;

  Color color2 = new Color(red,green,blue);

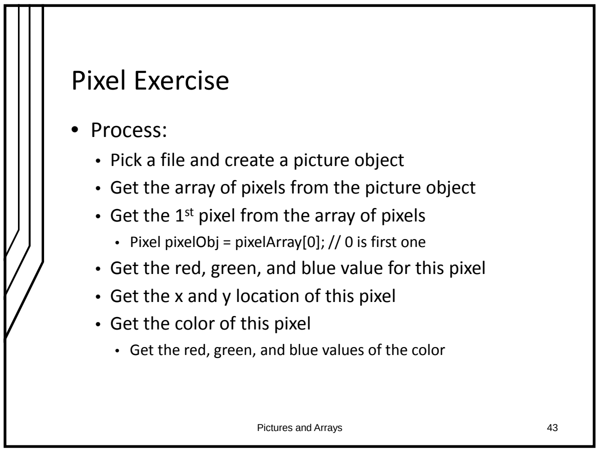
  pixelObj.setColor(color2);

Using Classes in Packages

- All classes in the Java language are in packages
  - You can use any class in java.lang
    - System, Math, Object
- For classes in other packages you need to import them
  - import java.awt.Color;
  - Import java.awt.*; //import all classes in this package
    - To use the short name: Color
- Or use the fully qualified name
  - *packageName.ClassName*
  - java.awt.Color

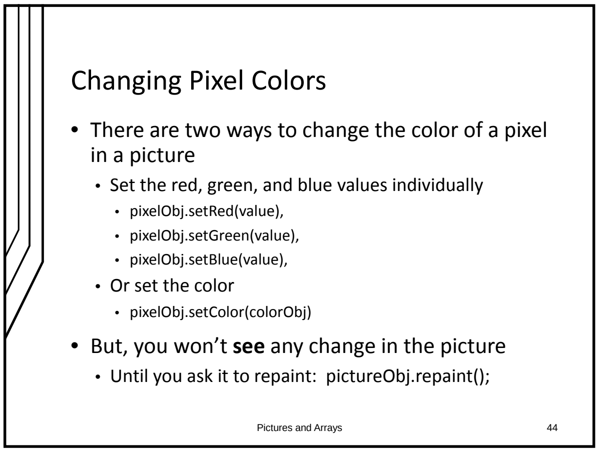
Undefined Class Error

- If you forget to import a class
  - Yet, you use the short name for the class
  - It won't compile
    - Undefined class error
- Undefined class errors mean
  - You need to import the class
  - Or you misspelled the class
  - Or used the wrong case



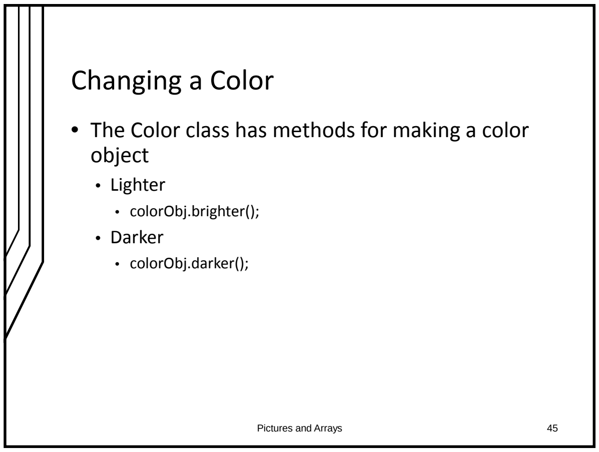
Pixel Exercise

- Process:
  - Pick a file and create a picture object
  - Get the array of pixels from the picture object
  - Get the 1st pixel from the array of pixels
    - Pixel pixelObj = pixelArray[0]; // 0 is first one
  - Get the red, green, and blue value for this pixel
  - Get the x and y location of this pixel
  - Get the color of this pixel
    - Get the red, green, and blue values of the color



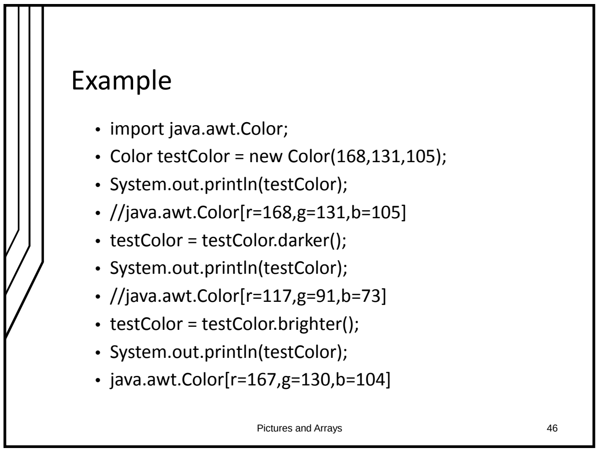
Changing Pixel Colors

- There are two ways to change the color of a pixel in a picture
  - Set the red, green, and blue values individually
    - pixelObj.setRed(value),
    - pixelObj.setGreen(value),
    - pixelObj.setBlue(value),
  - Or set the color
    - pixelObj.setColor(colorObj)
- But, you won’t **see** any change in the picture
  - Until you ask it to repaint: pictureObj.repaint();



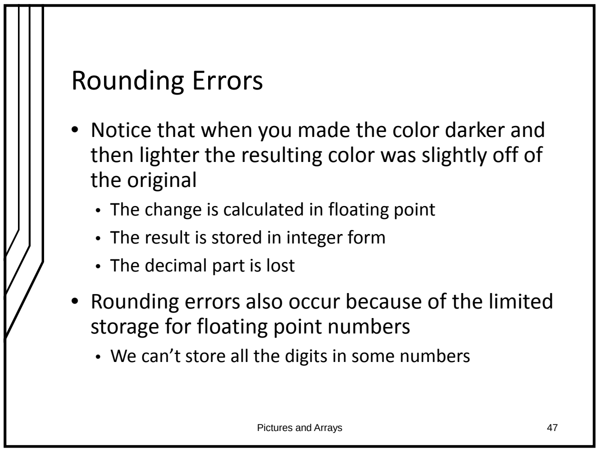
Changing a Color

- The Color class has methods for making a color object
  - Lighter
    - colorObj.brighter();
  - Darker
    - colorObj.darker();



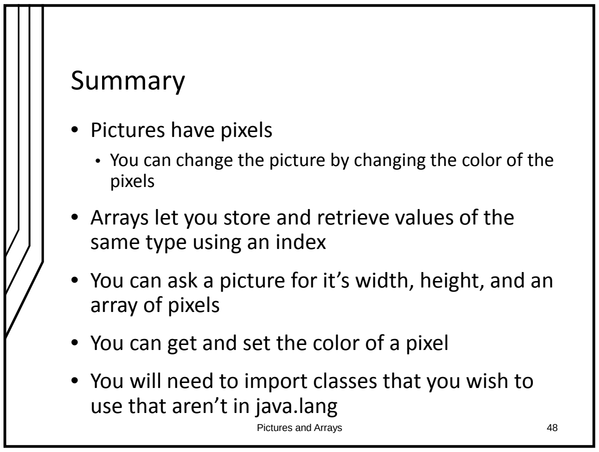
Example

- import java.awt.Color;
- Color testColor = new Color(168,131,105);
- System.out.println(testColor);
- //java.awt.Color[r=168,g=131,b=105]
- testColor = testColor.darker();
- System.out.println(testColor);
- //java.awt.Color[r=117,g=91,b=73]
- testColor = testColor.brighter();
- System.out.println(testColor);
- java.awt.Color[r=167,g=130,b=104]



Rounding Errors

- Notice that when you made the color darker and then lighter the resulting color was slightly off of the original
  - The change is calculated in floating point
  - The result is stored in integer form
  - The decimal part is lost
- Rounding errors also occur because of the limited storage for floating point numbers
  - We can’t store all the digits in some numbers



Summary

- Pictures have pixels
  - You can change the picture by changing the color of the pixels
- Arrays let you store and retrieve values of the same type using an index
- You can ask a picture for it’s width, height, and an array of pixels
- You can get and set the color of a pixel
- You will need to import classes that you wish to use that aren’t in java.lang