# TOPIC 5
# INTRODUCTION TO PICTURES

Notes adapted from Introduction to Computing and Programming with Java: A Multimedia Approach by M. Guzdial and B. Ericson, and instructor materials prepared by B. Ericson.

## Outline

- Pictures
- 1 Dimensional Arrays
- Pixels
- 2 Dimensional Arrays
- Colors

## Pictures

## Picture objects

- Recall: we can create a Picture object by

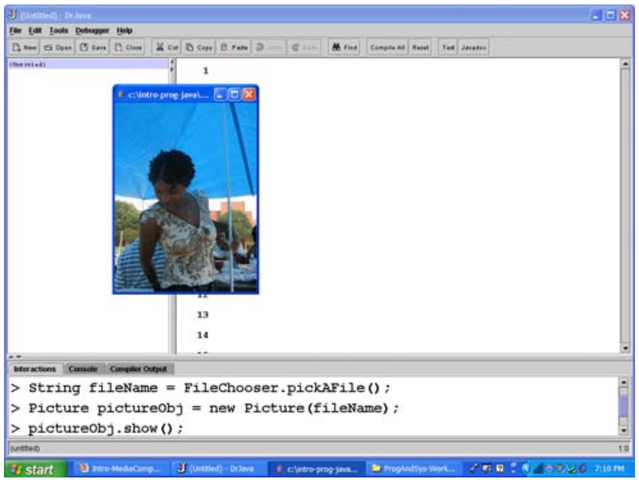
```
String fileName = FileChooser.pickAFile();
Picture pictureObj = new Picture (fileName);
```

- A Picture object has properties width and height
  - We can get the picture width (in pixels) using

    `pictureObj.getWidth()`
  - We can get the picture height (in pixels) using

    `pictureObj.getHeight()`

## Result



```
> String fileName = FileChooser.pickAFile();
> Picture pictureObj = new Picture(fileName);
> pictureObj.show();
```

## Arrays

## Motel California

- Let's say you run a motel
- It is 1 story
- Lily, Marshal, Ted, Robyn, Sheldon and Howard come to get rooms
- To make it easier, you use the letters of their first name to remember them: L, M, T, R, S, H
- You need to pick what rooms from your motel you want to put them in
- Your rooms are numbered from 0 to 5 (you like to be creative)

| Room Number: | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| Person: | L | M | R | T | S | H |

## Motel or Array?

- This motel is just like an array!
- An array is a way of storing items (just like we stored the people in the hotel)
- Each slot in an array is like a hotel room → it can only hold 1 item
- Each slot in an array is numbered like a hotel room → the numbers go from 0 → length-1
- Arrays make it easier to keep track and store lots of objects, just like a hotel makes it easier to store loads of people
- Arrays can be 1 dimensional (1 story, like our motel) or many dimensional

## 1D Arrays

- An **array** is storage for a collection of items of the same type
- You can access the individual items by using an **index**
- The index **starts at 0**
  - The first item is at index 0
  - The last item is at index (length – 1)

| 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 3 | 7 | 9 | 2 | 1 | 5 |

Example:
int array of length 6

## Creating arrays

- Use the Java keyword new
- Example: numbers = new int[6];
  - This creates an array of 6 integers, and has our reference variable numbers refer to it
  - The reference variable numbers refers to the entire collection of integers
  - This does **not** store any data in the array
- We can declare and create in one statement:
  int [ ] numbers = new int[6];
- Once we create an array, its size is fixed and we cannot change it

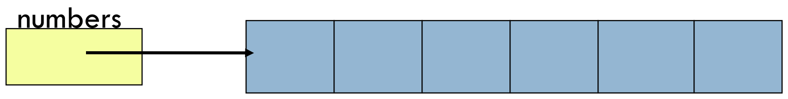
## Arrays

- In Java, arrays behave like **objects**
  - We need to declare a reference variable
  - We need to create the array object
- We **declare array reference variables** by

  type[ ] name;
- Example: int[ ] numbers;
  - This creates a reference variable called numbers that can be used to refer to an array of integers
  - But this does not actually create the array

## Creating arrays

int[ ] numbers = new int[6];

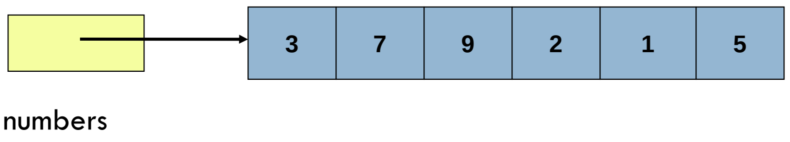


## Array indexing

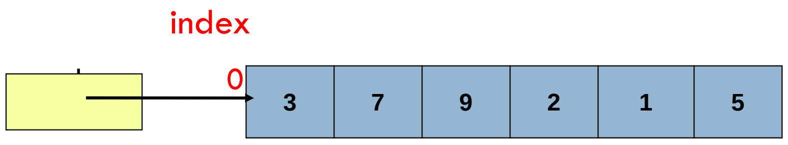
- Each item of an array can be accessed individually, using **indexing (like looking up a person by looking into their room number)**
- Examples:
  - numbers[0] refers to the first element of the array (the element at position 0)
  - numbers[1] refers to the second element (the element at position 1)
  - …
  - numbers[i] refers to the (i+1)th element (the element at position i)

## Storing in arrays

```
numbers[0] = 3;
numbers[1] = 7;
numbers[2] = 9;
numbers[3] = 2;
numbers[4] = 1;
numbers[5] = 5;
```



## Array indexing



What would be printed by
`System.out.println(numbers[0]);`

How would we print the last item in the array?

## Initializing arrays

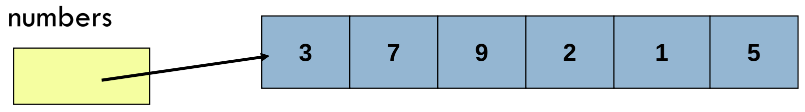
- We can declare, create, and **initialize** an array in one statement
- Example:

  `int[ ] numbers = { 3, 7, 9, 2, 1, 5 };`

  - This creates an array of 6 integers, and sets the initial values of the integers in the array according to the values specified
  - The length of the array is determined by the number of items listed

## Initializing arrays

int[ ] numbers = { 3, 7, 9, 2, 1, 5 };



## Array size

- Java remembers the size of arrays
  - An array object has a special attribute called length, which stores the size of the array
  - Note: length is a variable, not a method!
    - So there are no parentheses after length
- Example: int arraySize = numbers.length;
- Useful: to get the last item in an array, for example

  int lastNumber = numbers[numbers.length – 1];

## Array of Pictures

- Lets practice using arrays by making an array of pictures
- `Picture[] myPicArray = new Picture[3];`
- Now lets make some picture objects:

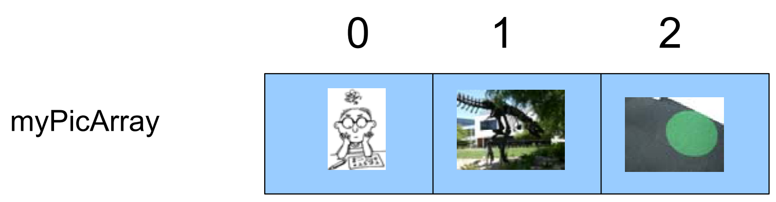
```
String fileName1 = FileChooser.pickAFile();
Picture picture1 = new Picture(fileName1);
String fileName2 = FileChooser.pickAFile();
Picture picture2 = new Picture(fileName2);
String fileName3 = FileChooser.pickAFile();
Picture picture3 = new Picture (fileName3);
```

**What is the length of this array? Does it have anything stored in it?**

## Array of Pictures

- Lets fill up our array!

```
myPicArray[0] = picture1;
myPicArray[1] = picture2;
myPicArray[2] = picture3;
```



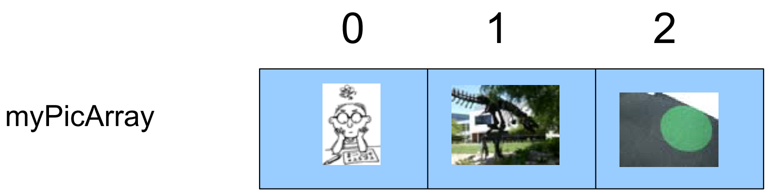
## Array of Pictures

- How would I access the picture of the green circle?

myPicArray[2]

- How would I see how long my array is?

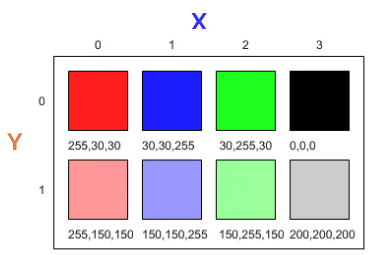
myPicArray.length;



## Pixels

## Picture as a grid of pixels

- A picture is organized as a **grid (matrix) of pixels**
- The grid has **columns** and **rows**
- Each pixel has an **(x, y) position** in the grid
  - x specifies the column, starting at 0
  - y specifies the row, starting at 0



## The Pixel class

- We have a class called Pixel that models individual pixels
- Each object of the Pixel class has
  - An (x,y) position in a picture
    - x specifies the column, starting at 0
    - y specifies the row, starting at 0
  - A red, green, and blue value
    - Each is an integer between 0 and 255

## Creating Pixel objects

- We can get a pixel at a specific location in a picture by using the getPixel method of the Picture class
- Example:
  `Pixel pixel1 = pictureObj.getPixel(0,0);`
  - This will create a Pixel object from the pixel in the picture at position 0,0
  - This is the top left-hand corner of the picture
  - It will store a reference to this Pixel object in the variable pixel1

## Manipulating Pixel objects

- We can get and set the red, green and blue values of a Pixel object individually, using methods of the Pixel class
- Example of getting a pixel's color values:

```
int red = pixel1.getRed();
int green = pixel1.getGreen();
int blue = pixel1.getBlue();
```

- Example of setting a pixel's color values:

```
pixel1.setRed(red+10);
pixel1.setGreen(0);
pixel1.setBlue(blue-10);
```

## Pixel location in a picture

- We can get the pixel's location in the grid of pixels that make up the Picture object:

  getX()
  - Returns its x position (the column)

  getY()
  - Returns its y position (the row)

- Example: what will be printed here?

```
System.out.println(pixel1.getX() + "," +
                    pixel1.getY());
```

## Two Dimension Arrays

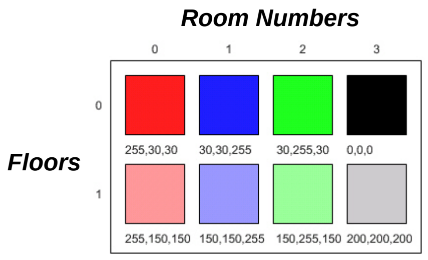
## 2D Arrays

- Early we saw 1D arrays as an example of a motel
- Now picture instead, the grid of pixels
- Its like a hotel for pixels, with multiple floors

Each pixel is in a different room that is specified both by the room and the floor

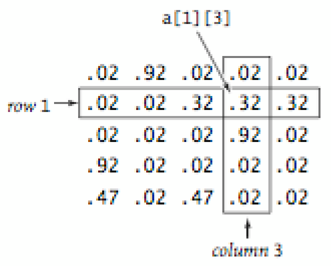
For example, the deep blue is in room 1, on floor 0

The gray is room 3 on floor 1



## 2D Arrays

- A 2D array is just like this hotel
- It is like a many arrays stacked on top of each other
- The picture to the side has a 2D array called "a"
- It has 5 arrays of numbers that are each 5 long
- You can still index to a particular number in the grid, but you have to say not just the column, but also the row it is in
- This is a great way to store pixels or objects represented in a matrix



*Anatomy of a two-dimensional array*

## Initializing 2D Arrays

- Instead of one set of square brackets, you use two to make a 2D array
- `int[][] twoDArray = new int[4][4];`

- This would make an array of 4 “rooms” and 4 “floors”
- In other words, a 4x4 matrix – 16 integer slots in total
- If you picture the grid of pixels, it makes sense to store it in a 2D array
- However in memory, 2D and 1D are very similar....
- (Copy down drawing from the board)

## 2D Array of Pixels

- If we were to make a 2D array of pixels, it might look like this:
- `Pixel[][] 2DPixel = new Pixel[200][200];`
- Then, after we put pixels into the array, we could access the pixel in a certain location (say column 4, row 19) by using:
- `Pixel pixelObject = 2DPixel[4][19];`
- More on how to best use 2D arrays later…

## 1D Arrays of pixels

- The pixels can also be stored as one long sequence, in a 1D array
- In this case, at the end of a row the next row just gets tagged on, and so on, and so on until each pixel is in the array
- In other words:
  - The pixes from the first row of the grid go in the array
  - Followed by the pixels from the second row
  - Etc.

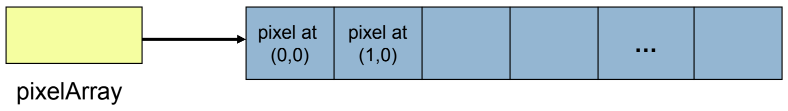
  To get the pixels from the picture in a 1D array:

```
Pixel[ ] pixelArray = pictureObj.getPixels();
```

## Arrays of pixels

This code would look like:

```
Pixel[ ] pixelArray = pictureObj.getPixels();
```



This creates an array of Pixel objects.
Note that each element of the array is actually a **reference** to a Pixel object.

## Pixel objects from a Pixel array

- Just like before, we use **indexing** to get Pixel objects from the 1D pixel array
- For example, to get the first pixel:
- `Pixel pixelObj = pixelArray[0];`

- To get the nth pixel:
  - `Pixel pixelObjn = pixelArray[n];`

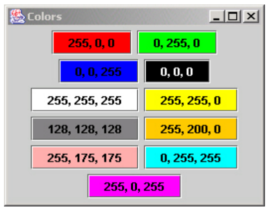
## Colors

## The Color class

- Recall the class defined in Java that represents color:
  - The Color class in the **package** java.awt
    - A package is a group of related classes
  - To use the class, you must either use
    - The full name java.awt.Color
    - Or, much easier, use the import statement import java.awt.Color;
      - Then you can just use the class name Color (without needing the name of the package as well)
      - In a Java program, import statements go at the beginning of the source file

## 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, …



## Color objects

- You can create a Color object by giving the red, green, and blue values
- Example:

```
Color colorObj = new Color(255,10,125);
```

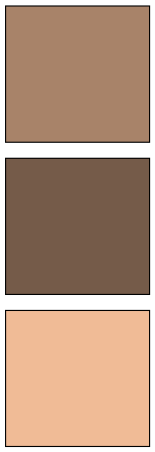
## Making colors lighter or darker

- The Color class has methods for making a Color object lighter or darker:

```
colorObj.brighter();
colorObj.darker();
```

- Example in Interactions pane:

```
> import java.awt.Color;
> Color testColor = new Color(168,131,105);
> System.out.println(testColor);
> Color darkColor = testColor.darker();
> System.out.println(darkColor);
> Color brightColor = testColor.brighter();
> System.out.println(brightColor);
```



## 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();
```

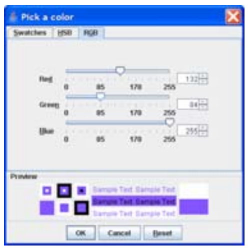
- To set a Pixel's color using a new Color object:

```
Color color2 = new Color(red+10, 0, blue-10);
pixelObj.setColor(color2);
```

## Choosing a Color

- You can also get a color by using the following method:
  ColorChooser.pickAColor()
- You can use this anywhere you would have used a Color object
- Example:

```
Color pickedColor = ColorChooser.pickAColor();
pixelObj.setColor(pickedColor);
```



Red 0 85 170 255 132
Green 0 85 170 255 84
Blue 0 85 170 255 255

## Pixel recap

```
import java.awt.Color;
String fileName = FileChooser.pickAFile();
Picture pictureObj = new Picture(fileName);
pictureObj.show();
Pixel [] pixelArray = pictureObj.getPixels();
Pixel pixelObj = pixelArray[0];
int red = pixelObj.getRed();
int green = pixelObj.getGreen();
int blue = pixelObj.getBlue();
System.out.println("r = "  + red + ", g = "  + green
                                            + ",  b = "  + blue);
```

## Pixel recap

```
Color colorObj = pixelObj.getColor();
red = colorObj.getRed();
green = colorObj.getGreen();
blue = colorObj.getBlue();
System.out.println("r = "  + red + ", g = "  + green
                                            + ",  b = "  + blue);
```

- In what class are these methods getRed, getGreen, getBlue defined?
- In what class are the methods getRed, getGreen, getBlue on the previous slide defined?

## Changing colors in a picture

- We have seen how to change the color of a pixel in a picture
- But you won't see any change in the picture until you **repaint the picture** by
  pictureObj.repaint();
- Another way to do this is by pictureObj.show();

## Changing a Picture Exercise

```
import java.awt.Color;
String fileName = FileChooser.pickAFile();
Picture pictureObj = new Picture(fileName);
pictureObj.show();
pictureObj.getPixel(10,100).setColor(Color.black);
pictureObj.getPixel(11,100).setColor(Color.black);
pictureObj.getPixel(12,100).setColor(Color.black);
pictureObj.getPixel(13,100).setColor(Color.black);
pictureObj.getPixel(14,100).setColor(Color.black);
pictureObj.getPixel(15,100).setColor(Color.black);
pictureObj.getPixel(16,100).setColor(Color.black);
pictureObj.getPixel(17,100).setColor(Color.black);
pictureObj.getPixel(18,100).setColor(Color.black);
pictureObj.getPixel(19,100).setColor(Color.black);
pictureObj.repaint();
```

## Saving changes to pictures

- After manipulating a picture, we can save our results to a file:

  ```
  pictureObj.write("newPicture.jpg");
  ```

- You can specify a full path so you know exactly where it is saved, for example:

  ```
  pictureObj.write("Z:/jane/MyPictures/newPicture.jpg");
  ```

- Or you can use the FileChooser here too:

  ```
  String fileName = FileChooser.pickAFile();
  pictureObj.write(fileName);
  ```

## Summary

- Java
  - Arrays
- Pictures
  - Picture as an array of Pixel objects
- Pixels
  - Getting/setting  red, green, blue values of pixel
  - Getting/setting color of pixel as a Color object

## Key Notes

- 1D arrays
  - Go from 0 → length-1
  - Created using: type[] varName = new type[length]
  - Can store many things in an array → integers, strings, pixels, pictures, etc.
- Pixels in a grid make up a picture
  - They have a row and a column #
  - Pixel pixelName = pictureObj.getPixel();
- 2D arrays
  - type[][] varName = new type[#][#]