RETURNING VALUES FROM METHODS

PICTURE TRANSFORMATIONS

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

- How to return a value from a method
- How to transform a picture into one of a different size
  - Rotation
  - (Scaling up and scaling down)
- Returning a Picture object from a method

Return Values from Methods

- Recall that methods can **return** values
- We have invoked some methods that returned something, for example:
  - getWidth() returns an int

    ```
    for (int x = 0; x < pictureObj.getWidth(); x ++) …
    ```
  - getPixel() returns a reference to a Pixel object

    ```
    Pixel pixelObj = pictureObj.getPixel(x,y);
    ```
  - getPixels() returns a reference to an array of pixels

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

Return Values: Count White Pixels

```
 // this is an object method
public int countWhitePixels()
{
  int counter = 0;
  // loop through the columns (x direction)
  for (int x = 0; x < this.getWidth(); x++)
  {
     // loop through the rows (y direction)
     for (int y = 0; y < this.getHeight(); y++)
       {
```

Counting White Pixels

```
    // get the pixel at the x and y location
    Pixel pixelObj = this.getPixel(x,y);

    // if the pixel is white increment the counter
    if (pixelObj.getRed()==255 && pixelObj.getGreen()==255
    && pixelObj.getBlue() == 255)
          counter = counter + 1;
    }
  }
  return counter;
}
```

Rules

- The return type **must** be specified in the method header, for example
  public int countWhitePixels()
- The **return** statement sends a value back to where the method was invoked
  - The returned value can be stored in a variable:
    int numWhitePixels = pictureObj.countWhitePixels();
  - Or it can be used directly:
    System.out.println(pictureObj.countWhitePixels());

Warning

The method must **always** return a value of the **correct type**

```
public int badMethod1(){
  return 1.1;
}
public int badMethod2(int x){
  if (x < 10)
    return 1;
}
```

Warning

The method must **always** return a value of the **correct type**

```
public int badMethod3(int x){
  if (x < 10)
    return 1;
  if (x >= 10)
    return 2;
}
```

Example

- A return value can be a boolean value, i.e. true or false
- Exercise: Write a method for the Picture class that checks whether two pictures are of the same size
  public boolean equalSize(Picture otherPic)
  - It will be invoked on a Picture object
  - If this Picture object is of the same size as the parameter picture, the method returns true, otherwise it returns false

Returning a Picture Object

- So far, we have invoked our picture methods on a **target** Picture object
- We will now write a new version of decreaseRed
  - It will create a new target Picture object inside the method
  - It will return this Picture object as the result of the method

Decrease Red Method

- The new decreaseRed method will
  - be invoked on the source picture
  - return a target picture that has the same dimensions as the source picture
- So we don't need to pass the source picture as a parameter
- Example of a call to the new decreaseRed() :

```
Picture sourcePic = new Picture(…);
Picture targetPic = sourcePic.decreaseRed();
targetPic.show();
```

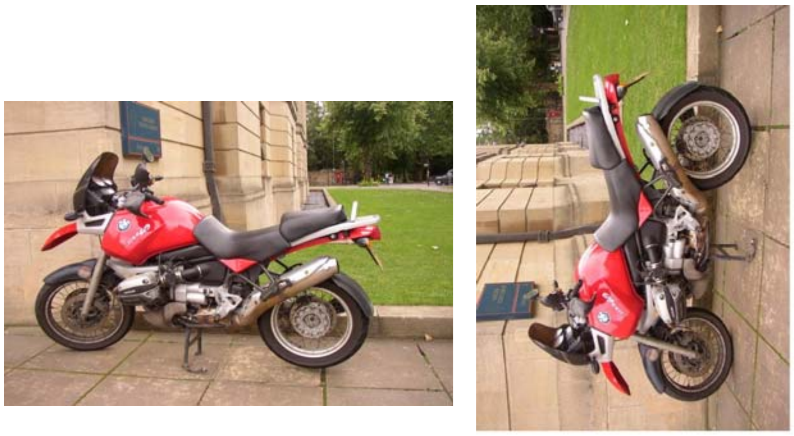
Decrease Red Method

```
public Picture decreaseRed()
 {
   Picture targetPicture = new Picture(this.getWidth(), this.getHeight());
  // loop through the columns
  for (int x = 0; x < this.getWidth(); x++)
  {
    // loop through the rows
    for (int y = 0; y < this.getHeight(); y++)
    {
      sourcePixel = this.getPixel(x, y);
      int redValue = sourcePixel.getRed();
      int greenValue = sourcePixel.getGreen();
      int blueValue = sourcePixel.getBlue();

     // decrease the red value
     redValue = redValue / 2;
```

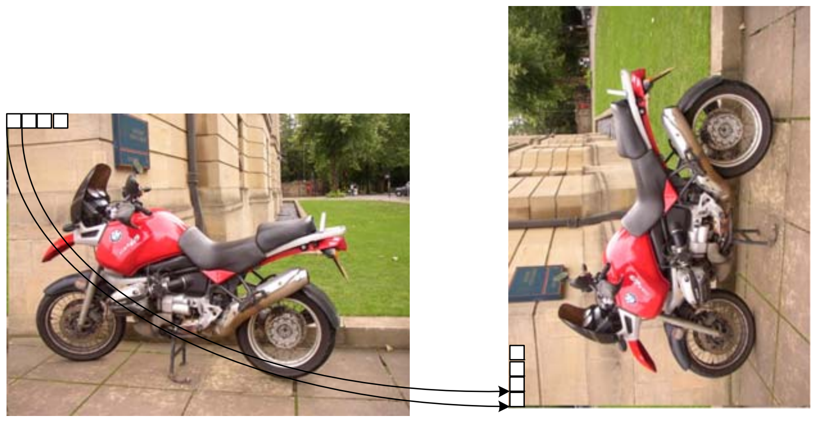
Decrease Red Method (continued)

```
    // assign target picture values
    Pixel targetPixel = targetPicture.getPixel(x, y);
    targetPixel.setGreen(greenValue);
    targetPixel.setBlue(greenValue);
    targetPixel.setRed(redValue);

    }
   }
 return targetPicture;
}
```

Rotating Pictures



Rotating Pictures



Left Rotation

- To rotate an image 90 degrees to the left, we copy all the pixels, but they go to *different locations* in the target
- (0,0) goes to (0,2)
  (1,0) goes to (0,1)
  (2,0) goes to (0,0)
  (0,1) goes to (1,2)
  (1,1) goes to (1,1)
  (2,1) goes to (1,0)
- What happens to the source row (y) cordinates?
- They go to the target column (x) coordinates :

  target x = source y

(0,0)	(1,0)	(2,0)
(0,1)	(1,1)	(2,1)

(2,0) goes here	(2,1) goes here
(1,0) goes here	(1,1) goes here
(0,0) goes here	(0,1) goes here

Left Rotation

- What happens to the source column (x) coordinates?
  (0,0) goes to (0,2)
  (1,0) goes to (0,1)
  (2,0) goes to (0,0)
  (0,1) goes to (1,2)
  (1,1) goes to (1,1)
  (2,1) goes to (1,0)

- They go to the target row (y) coordinates that are calculated by:

  target y =
  (source width –1) – source x

	0	1	2
0	(0,0)	(1,0)	(2,0)
1	(0,1)	(1,1)	(2,1)

	0	1
0	(2,0) goes here	(2,1) goes here
1	(1,0) goes here	(1,1) goes here
2	(0,0) goes here	(0,1) goes here

Left Rotation Method

- The copyLeftRotation method will
  - be invoked on the source picture
  - return a target picture that is the source picture rotated left
- So we don't need to pass the source picture as a parameter
- Example of a call to the new copyLeftRotation() :

```
Picture sourcePic = new Picture(...);
Picture targetPic = sourcePic.copyLeftRotation();
targetPic.show();
```

Left Rotation Method

```
public Picture copyLeftRotation(){
   Picture targetPicture = new Picture(this.getHeight(),
   this.getWidth());
   for (int sourceX = 0;  sourceX < this.getWidth();  sourceX++){
     for (int sourceY = 0;  sourceY < this.getHeight();  sourceY++){

       int targetX = sourceY;
       int targetY = (this.getWidth() – 1) – sourceX;
       Pixel sourcePixel = this.getPixel(sourceX,  sourceY);
       Pixel targetPixel = targetPicture.getPixel(targetX,  targetY);
       targetPixel.setColor(sourcePixel.getColor());
     }
   }
   return targetPicture;
 }
```

Summary

- Returning values from Methods
- Returning Pictures
- Picture Algorithms that Return Pictures