### Manipulating Pictures

CS1316: Representing Structure and Behavior

### Contents
- Miscellaneous Java details
- Writing a method
  - Giving a method varying input values
- Method parameters
  - Giving a method varying input values
- Function methods
  - Returning a value or object from a method
- Running a program
  - The static `main` method

### Assignment
- `<Class> <variable> = <expression>;`
- `<variable> = <expression>;`
  - If the variable has already been declared.
  - You can’t declare a variable twice.
  - Note: In DrJava Interactions pane, variables will be declared for you.
- **Style:**
  - Capitalize your classnames
  - Lowercase everything else
  - But can use mixed case to break up combined words

### Java: Expressions and Indentation
- In Java, statements end with `;`.
  - You can use as many lines as you want, insert spaces and returns almost whenever you want. The `semicolon` is the end of the statement.
- Indentation doesn’t matter at all.
  - DrJava will indent for you, but just to make it easier to read.

### Declaring a variable
- `<Classname> <variable>;
- `<Classname>[] <variable>;
- `<Classname> <variable>[];
  - With the square brackets notation, you’re declaring an array. (Turns out either way works.)
  - To access part of an array, you’ll use square brackets, e.g., `myPicturesArray[5]`

### Expressions
- `new <Classname>(<maybe inputs>)`
  - Makes a new instance of the class
- `*, /, +, -`
  - A shortcut:
    - `x = x + 1` is so common that it can be shortened to `x++`
    - `x = x + y` is so common that it can be shortened to `x += y`

### Conditionals
- `if (<logical-expression>) then-statement;`
  - Logical expressions are like you’d expect: `<`, `>`, `<=`, `>=`, `==`
  - Logical “and” is `&&`
  - Logical “or” is `||`
  - BUT then-statement can be a single statement or any number of statements (in curly braces).

### Conditional examples
- `if (thisColor == myColor) setColor(thisPixel, newColor);`
- `if (thisColor == myColor) {setColor(thisPixel, newColor);}`
  - Need this one to end the statement inside the curly braces

### A “Block”
- We call the curly braces and the code within it a block.
  - A Java statement (think “sentence”) can end in a semi-colon or a right-curly-brace (think “- –”, or “| – |” or “¿ – ?”)

### Java: Expressions and Indentation

- **In Java, statements end with `;`**
  - You can use as many lines as you want, insert spaces and returns almost whenever you want. The `semicolon` is the end of the statement.
- **Indentation doesn’t matter at all.**
  - DrJava will indent for you, but just to make it easier to read.

### Declaring a variable

- `<Classname> <variable>;
- `<Classname>[] <variable>;
- `<Classname> <variable>[];
  - With the square brackets notation, you’re declaring an array. (Turns out either way works.)
  - To access part of an array, you’ll use square brackets, e.g., `myPicturesArray[5]`

### Expressions

- `new <Classname>(<maybe inputs>)`
  - Makes a new instance of the class
- `*, /, +, -`
  - A shortcut:
    - `x = x + 1` is so common that it can be shortened to `x++`
    - `x = x + y` is so common that it can be shortened to `x += y`

### Conditionals

- `if (<logical-expression>) then-statement;`
  - Logical expressions are like you’d expect: `<`, `>`, `<=`, `>=`, `==`
  - Logical “and” is `&&`
  - Logical “or” is `||`
  - BUT then-statement can be a single statement or any number of statements (in curly braces).

### Conditional examples

- `if (thisColor == myColor) setColor(thisPixel, newColor);`
- `if (thisColor == myColor) {setColor(thisPixel, newColor);}`
  - Need this one to end the statement inside the curly braces

### A “Block”

- We call the curly braces and the code within it a block.
  - A Java statement (think “sentence”) can end in a semi-colon or a right-curly-brace (think “- –”, or “| – |” or “¿ – ?”)
Public?

- In Java, we can control which parts of our programs other people can access to.
- Think about running a large organization.
  - You want these outside your organization representing your company through pre-written machinations: Press-releases, switchboards, technical support, salespeople.
  - In Java, you can declare what is public and what is private (or protected for just related classes).

In Java, you can declare what is public and what is private.
- For now, we'll make all classes and methods public, and it's probably best to make all data private.

### Contents

- Miscellaneous Java details
  - Writing a method
  - Method parameters
  - Giving a method varying input values
  - Function methods
  - Returning a value or object from a method
  - Running a program
- The static main method

### Examples

#### Example 1: Decreased

```
/**
 * Decrease is the function to decrease the current pixels.
 */
public int decrease(){
  for (int i=0; i<mypixels.length; i++)
    {mypixels[i].setRed(0);}
  return 0;
}
```

#### Example 2: Iteration: For

```
for (int i=0; i<mypixels.length; i++)
  {mypixels[i].setRed(0);}
```
Using this method

```java
Picture mypicture = new Picture(FileChooser.pickAFile());
mypicture.decreaseRed();
mypicture.write("D:/cs1316/less-red-bridge.png");
```

More ways to comment

```java
/**
 * Method to decrease the red by half in the current picture
 */

public void decreaseRed()
{
    // Skipping the insides for a minute.
}
```

A method definition

```java
public void decreaseRed()
{
    // Skipping the insides for a minute.
}
```

Variables we'll need in this method

```java
Pixel pixel = null; // the current pixel
int redValue; // the amount of red
```

More data for the method

```java
// get the array of pixels for this picture object
Pixel[] pixels = this.getPixels();
// start the index at 0
int index = 0;

// loop while the index is less than the length of the pixels array
while (index < pixels.length)
{
    // get the current pixel at this index
    pixel = pixels[index];
    // get the red value at the pixel
    redValue = pixel.getRed();
    // set the red value to half what it was
    redValue = (int) (redValue * 0.5);
    // set the red for this pixel to the new value
    pixel.setRed(redValue);
    // increment the index
    index++;
```

A loop for decreasing red

```java
All arrays know their length
• This is a reference to a variable known only to the object
• We get the pixel, then get the pixel's red value.
• When we multiply by 0.5, we create a float
• We say (int) to turn the value back into an integer to put in redValue.
• Then we set the pixel's red to the new value.
• Finally, we move to the next pixel by incrementing the index.
```

Contents

- Miscellaneous Java details
- Writing a method
  - Method parameters
    - Giving a method varying input values
  - Function methods
    - Returning a value or object from a method
    - Running a program
  - The static main method

Example 2: Decreasing red by an amount

```java
public void decreaseRed(double amount)
{
    Pixel[] pixels = this.getPixels();
    Pixel p = null;
    int value = 0;

    // loop through all the pixels
    for (int i = 0; i < pixels.length; i++)
    {
        // get the current pixel
        p = pixels[i];
        // get the value
        value = p.getRed();
        // set the red value the passed amount time what it was
        p.setRed((int) (value * amount));
    }
```

What do Pictures and Pixels know?

```java
That's what the JavaDoc documentation tells you.
```
Returning a picture

```java
public Picture scale(double factor)
```

- This scaling method returns a new instance of `Picture`.
- It doesn't change the original.
- That will turn out to be an advantage.
- This version takes a factor for how much to scale the target picture (this)

Inheritance

```
public class Picture extends SimplePicture
```

- "But hang on a minute! The class `Picture` doesn't actually know much at all!!"
- Right. `Picture` inherits from `SimplePicture`.

```
public class SimplePicture
```

- That means that much of what `Picture` knows and can do comes from `SimplePicture`.
- We'll talk more about "Why would you want to do that?" later

JavaDoc

- When comments are inserted in a particular format in Java classes and methods, documentation for that class and method can be automatically generated.
- This is called JavaDoc. Java Documentation.

```
/**
 * @return the scaled picture
 */
```

- This version takes a `instance of SimplePicture`
- If you rename the file, it will no longer be a JavaDoc.
- Not all of `Picture`, `Sound`, etc. are in JavaDoc.
- You do need to read the `Picture` and `Sound` classes.
- All of `Picture`, `Sound`, etc. are in `JavaDoc`.

Java users figure out what\*```
And return the new picture at the end:

```
return canvas;
```

Like in Python, anything you create in a method only exists inside that method:
- If you want it to get outside the context (or scope) of that method, you have to return it.

**Why should we want to do that?**
- Manipulation without changing the original: Cascading methods
  - rose.scale(0.5).compose(blank,10,10);
  - This returns a Picture — and rose is not changed!

This is a method that's understood by Picture. Why, that's what scale returns!

BTW, can use `explore()` as well as `show()` to see results or plan our compositions!

**Some of the methods in Picture that are useful in cascades**
- `public Picture scale(double factor)`
- `public void chromakey( Picture target, Color bgcolor, int threshold, int targetx, int targety)`
- `public void bluescreen( Picture target, int targetx, int targety)`
- `public void compose( Picture target, int targetx, int targety)`
- `public Picture flip()`

**How do you use all of those?**
- If you were (say) to build a collage, you'd want to use these methods, but probably not in a method for Picture.
  - Individual picture objects shouldn't necessarily be responsible for assembling lots of pictures.
  - In general: How do you build a program that simply uses other objects?

**public static void main(String[] args)**
- The answer isn't very object-oriented.
- You create a class with one method, with statements as if it were in the Interactions Pane.
  - It's a main method, and it uses the gobbledygook above.
  - It can be run from DrJava with a menu item AND from the Command prompt.

**Example 4: MyPicture.java**

```
public class MyPicture {
    public static void main(String[] args) {
        Picture canvas = new Picture(600,600);
        Picture swan = new Picture("D:/cs1316/MediaSources/swan.jpg");
        Picture rose = new Picture("D:/cs1316/MediaSources/rose.jpg");
        Picture turtle = new Picture("D:/cs1316/MediaSources/turtle.jpg");
        swan.scale(0.5).compose(canvas,10,10);
        swan.scale(0.5).compose(canvas,350,350);
        swan.flip().scale(0.5).compose(canvas,10,350);
        swan.flip().scale(0.5).compose(canvas,350,10);
        rose.scale(0.25).compose(canvas,200,200);
        turtle.scale(2.0).compose(canvas,10,200);
        canvas.show();
    }
}
```

To run it:
Under Tools menu:
- Run from DrJava

Contents
- Miscellaneous Java details
- Writing a method
- Method parameters
- Giving a method varying input values
- Function methods
- A value returned from a method
- Running a program
  - The static main method

To be continued...