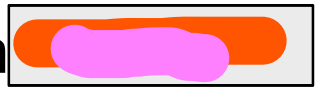
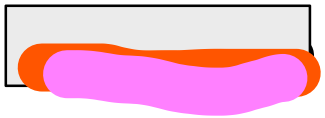
2.4 Documentation and Style-Outline

- Meaningful Names
- Self-Documentation and Comments
- Indentation
- Named Constants

- Most programs are modified over time to respond to new requirements.
- Programs which are easy to read and understand are easy to modify.
- Even if it will be used only once, you have to read it in order to debug it .

Meaningful Names for Variables

- A variable's name should suggest its use.
- Observe conventions in choosing names for variables.
  - » Use only letters and digits.
  - »  using uppercase letters at word boundaries (e.g. `taxRate`).
  - » Start variables with  letters.
  - » Start class names with  letters

- Use meaningful names for variables, classes, etc.
- Use indentation and line spacing as shown in the examples in the text
- Always include a "[redacted]" (an brief explanation of the program at the beginning of the file)
- Use all lower case for variables, except [redacted] internal words (**`eggsPerBasket`**)
- Use [redacted] for variables that have [redacted] value, **`PI`** for the value of pi (3.14159…) (see text for more examples)

Documentation and Comments

- The best programs are self-documenting.
  - » clean style
  - » well-chosen names
- Comments are written into a program as needed explain the program.
  - » They are useful to the programmer, but they are ignored by the compiler.

Comments & indenting

- —text in a program that the compiler ignores
- Does not change what the program does, only explains the program
- Write meaningful and useful comments
- Comment the obvious
- Assume a *reasonably* knowledgeable reader
- `//` for comments
- `/* … */` for comments
- /** …. */ for

- A comment can begin with //.
  - » Everything after these symbols and to the end of the line is treated as a comment and is ignored by the compiler.

```
double radius; //in centimeters
```

- A comment can begin with /* and end with */
  - » Everything between these symbols is treated as a comment and is ignored by the compiler.

```
/* the simplex method is used to
   calculate the answer*/
```

- A [redacted] comment, begins with /** and ends with */.
  - » It can be extracted automatically from Java software.

```
/** method change requires the number of coins to be
  nonnegative */
```

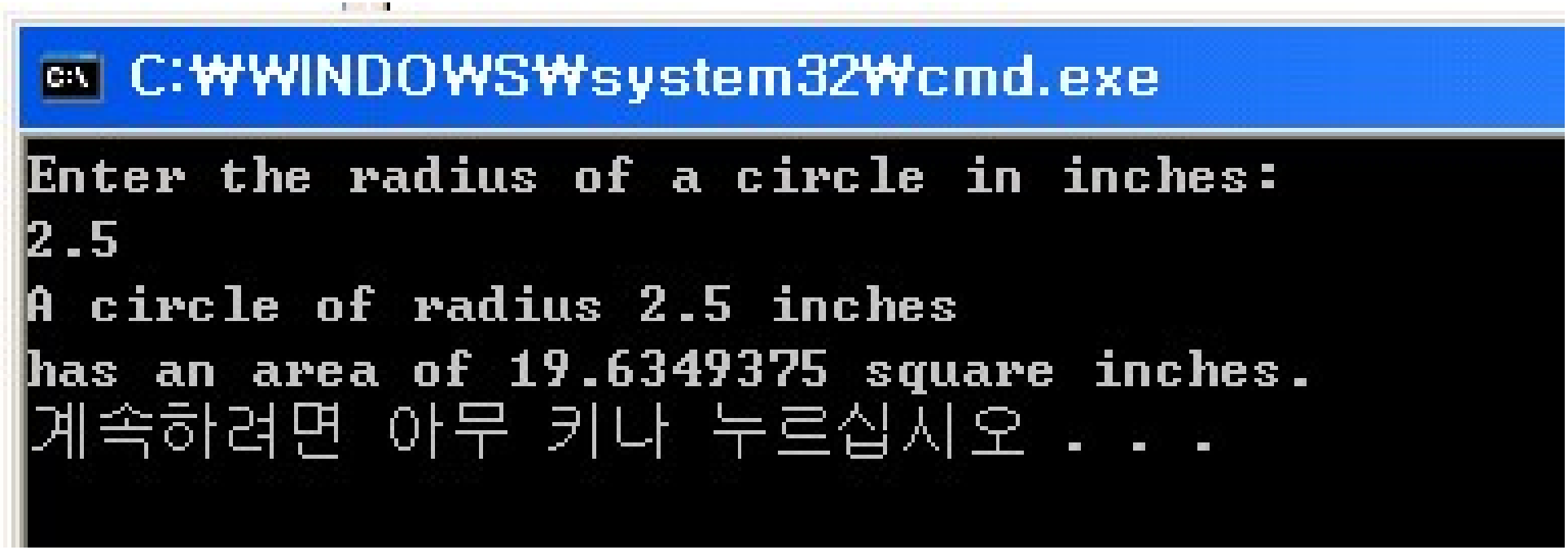
When to Use Comments

- Begin each program file with an explanatory comment
  - » what the program does
  - » the name of the author
  - » contact information for the author
  - » date of the last modification.
- Provide only those comments which the expected reader of the program file will need in order to understand it.

```
// Listing 2.7 Comments and Indenting

import java.util.Scanner;
/**
 Program to compute area of a circle.
 Author: Jane Q. Programmer.
 E-mail Address: janeq@somemachine.etc.etc.
 Programming Assignment 2.
 Last Changed: October 7, 2013.
*/

public class CircleCalculation
{
   public static void main(String[] args)
   {
      double radius; //in inches
      double area;   //in square inches
      Scanner keyboard = new Scanner(System.in);

      System.out.println("Enter the radius of a circle in inches:");
      radius = keyboard.nextDouble( );
      area = 3.14159 * radius * radius;
      System.out.println("A circle of radius " + radius + " inches");
      System.out.println("has an area of " + area + " square inches.");
   }
}
```

Listing 2.7

- **Listing 2.7** . Comments and Indenting.
  - » CircleCalculation.java



Indentation

- Indentation should communicate nesting clearly.
- I good choice is four spaces for each level of indentation.
- Indentation should be consistent.
- Indentation should be used for second and subsequent lines of statements which do not fit on a single line.
- Indentation does not change the behavior of the program.
- Improper indentation can miscommunicate the behavior of the program.

Named Constants

- [redacted] *constant*—using a name instead of a value
- Example: use **`MORTGAGE_INTEREST_RATE`** instead of **`8.5`**
- Advantages of using named constants (??)
  - » [redacted] because reader can tell how the value is being used
  - » [redacted] because value can be changed in one place (the definition) instead of being changed everywhere in the program.
  - » [redacted] used for a different purpose

Defining Named Constants

```
public static final double PI = 3.14159;
```

public—no restrictions on where this name can be used

static—must be included, but explanation has to wait

final—the program is not allowed

- The remainder of the definition is similar to a variable declaration and gives the type, name, and initial value.
- A declaration like this is usually at the beginning of the file and is not the main method definition.

```
// Listing 2.8 Naming a Constant
import java.util.Scanner;

/**
 Program to compute area of a circle.
 Author: Jane Q. Programmer.
 E-mail Address: janeq@somemachine.etc.etc.
 Programming Assignment 2.
 Last Changed: October 7, 2013.
*/

public class CircleCalculation2
{
   public static final double PI = 3.14159;

   public static void main(String[] args)
   {
      double radius; //in inches
      double area;   //in square inches
      Scanner keyboard = new Scanner(System.in);

      System.out.println("Enter the radius of a circle in inches:");
      radius = keyboard.nextDouble( );
      area = PI * radius * radius;
      System.out.println("A circle of radius " + radius + " inches");
      System.out.println("has an area of " + area + " square inches.");
   }
}
```

Listing 2.8

- Listing 2.8. Naming a Constant
  - » CircleCalculatioin2.java