Chapter 2
Data and Expressions

- Character strings
- Primitive data
- The declaration and use of variables
- Expressions and operator precedence
- Data conversions
- Accepting input from the user
- Java applets
- Introduction to graphics

Character Strings

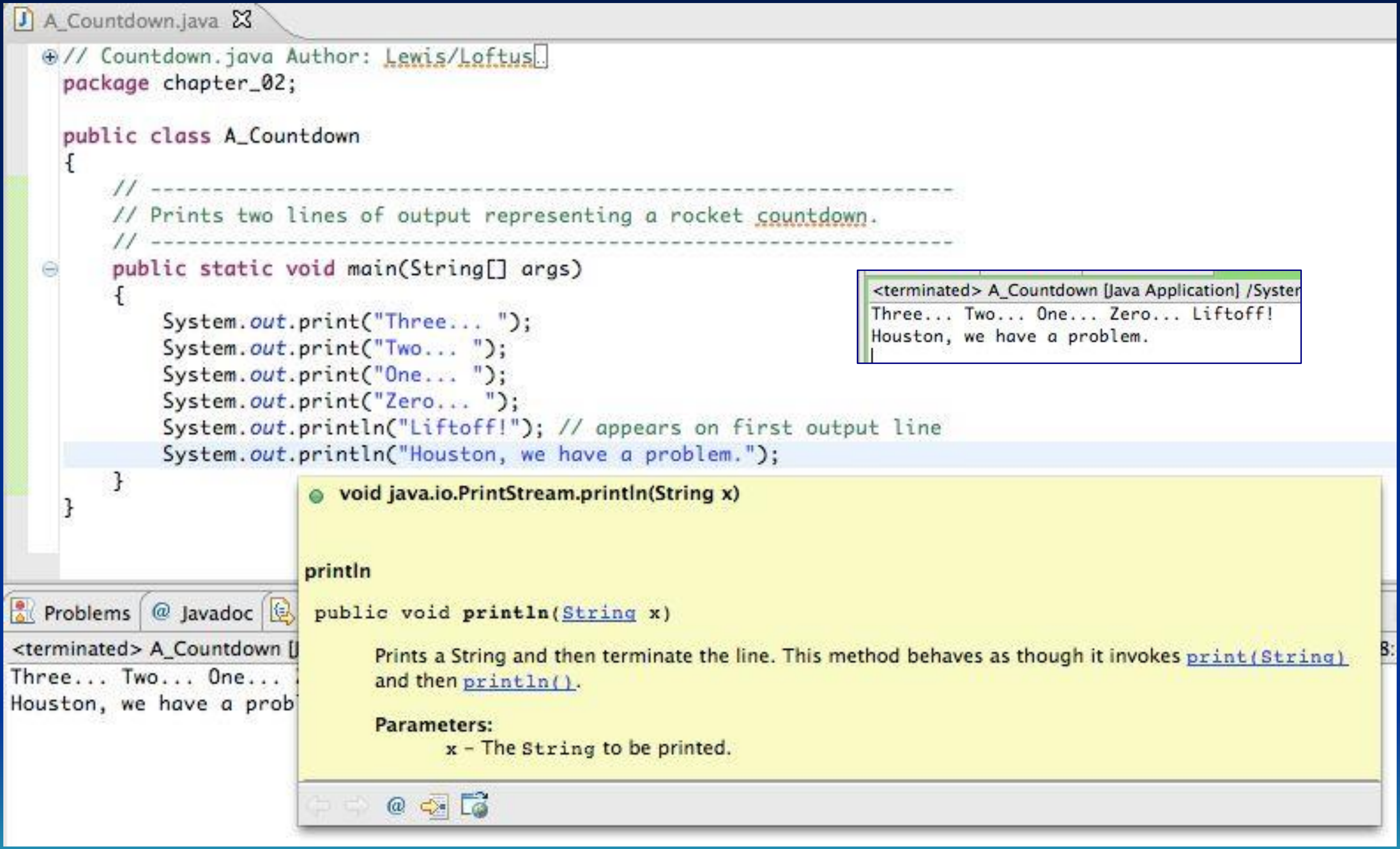
- A String is a sequence of characters.
- A string literal is represented by putting double quotes around the text
    - "This is a string literal."
    - "123 Main Street"
    - "X"
- Every character string is an object in Java, defined by the String class.

The println Method

- In the Lincoln program from Chapter 1, we invoked the println method to print a character string
- The System.out object represents a destination (the monitor screen) to which we can send output
- The print method is similar to the println method, except that it does not advance to the next line

$$\underbrace{\texttt{System.out}}_{\text{object}}\texttt{.}\underset{\substack{\uparrow \\ \text{method} \\ \text{name}}}{\texttt{println}}\ \texttt{(}\underbrace{\texttt{"Whatever you are, be a good one."}}_{\substack{\text{information provided to the method} \\ \text{(parameters)}}}\texttt{);}$$

The print and println Method



A_Countdown.java

```
// Countdown.java Author: Lewis/Loftus
package chapter_02;

public class A_Countdown
{
    // -----------------------------------------------------------------
    // Prints two lines of output representing a rocket countdown.
    // -----------------------------------------------------------------
    public static void main(String[] args)
    {
        System.out.print("Three... ");
        System.out.print("Two... ");
        System.out.print("One... ");
        System.out.print("Zero... ");
        System.out.println("Liftoff!"); // appears on first output line
        System.out.println("Houston, we have a problem.");
    }
}
```

```
<terminated> A_Countdown [Java Application] /Syster
Three... Two... One... Zero... Liftoff!
Houston, we have a problem.
```

void java.io.PrintStream.println(String x)

println

`public void println(String x)`

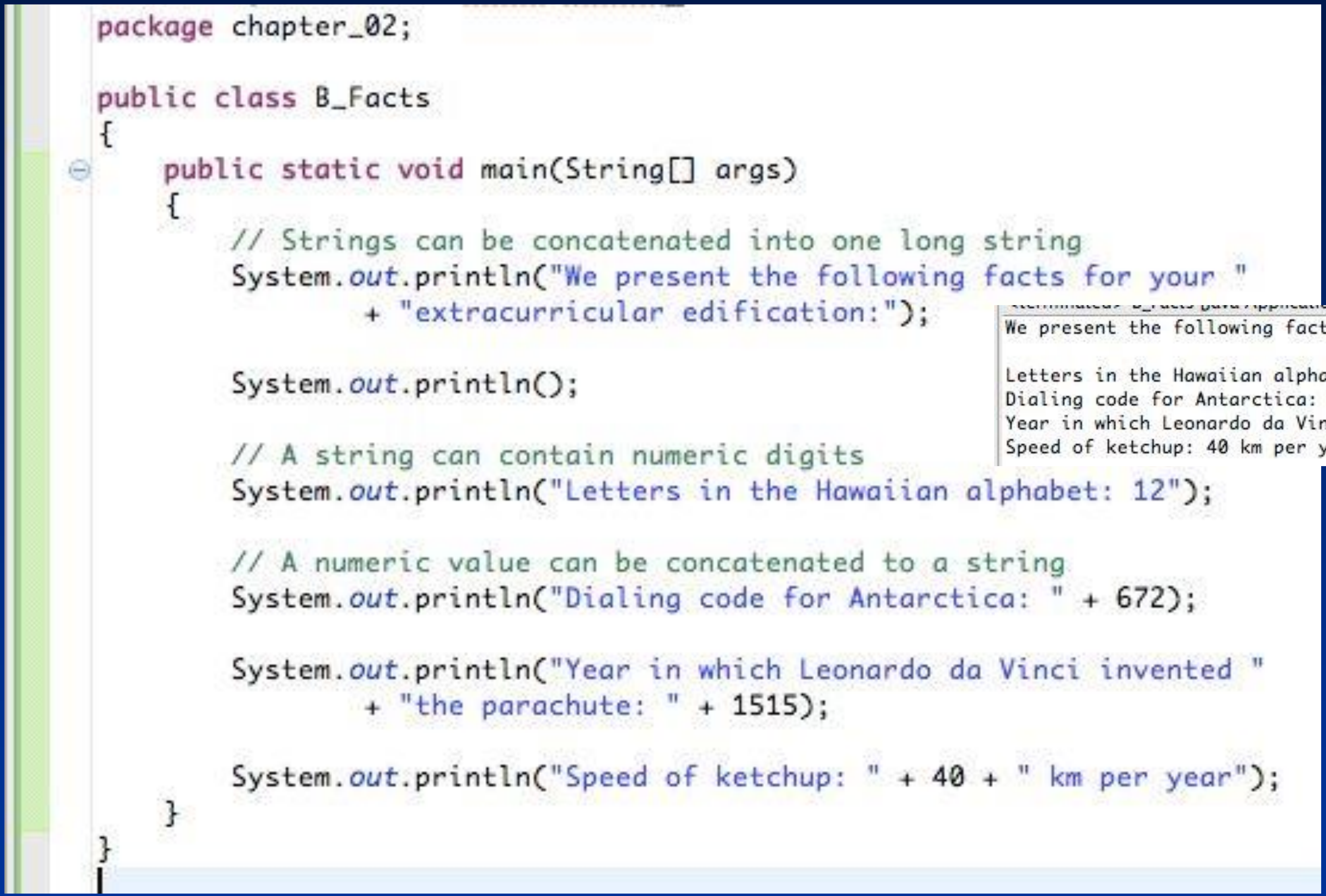
Prints a String and then terminate the line. This method behaves as though it invokes `print(String)` and then `println()`.

Parameters:
`x` - The `String` to be printed.

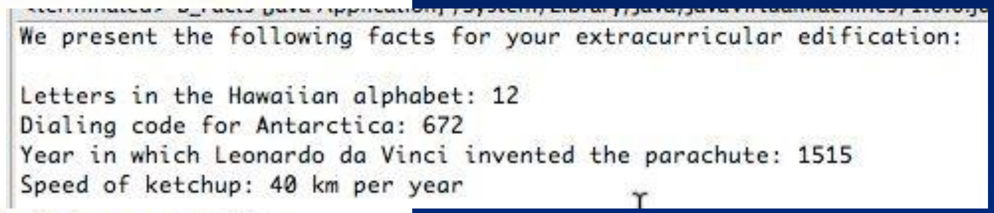
String Concatenation

- The string concatenation operator (+) is used to append one string to the end of another
  - “Peanut butter” + “ and jelly”=“Peanut butter and jelly”
- A string literal cannot be broken across two lines in a program
- The operator (+) depends on the type of information on which it operates
  - If both operands are strings, or if one is a string and one is a number, it performs string concatenation
  - If both operands are numeric, it adds them
- The + operator is evaluated left to right, but parentheses can be used to force the order

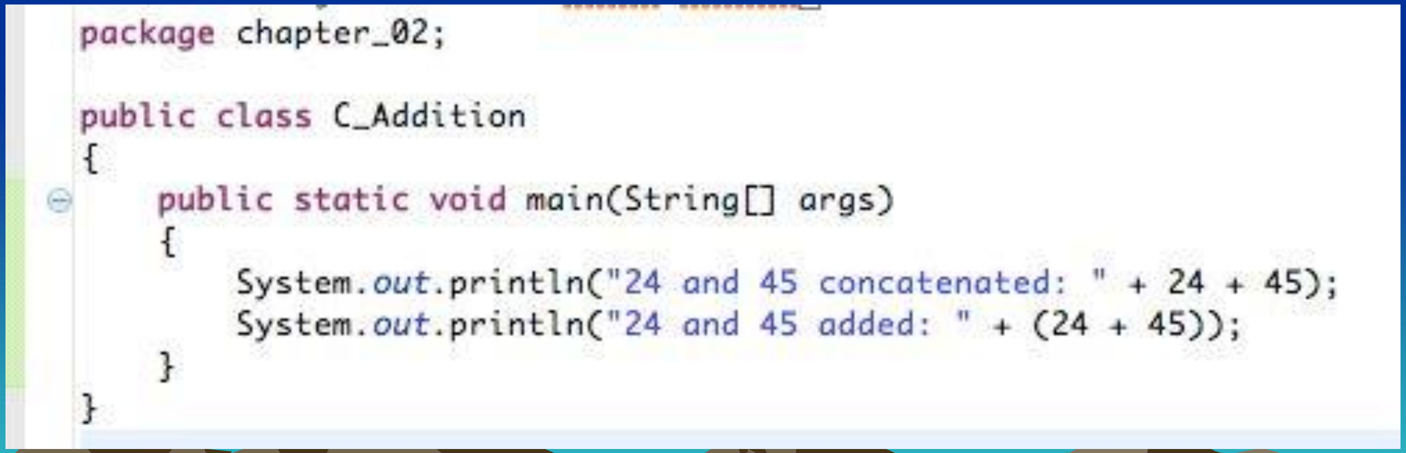
String Concatenation vs Addition



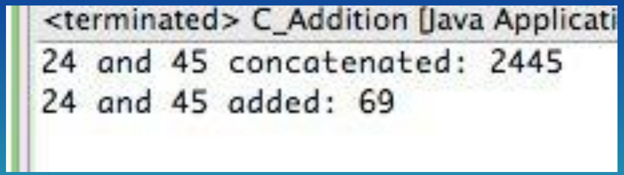
```java
package chapter_02;

public class B_Facts
{
    public static void main(String[] args)
    {
        // Strings can be concatenated into one long string
        System.out.println("We present the following facts for your "
                + "extracurricular edification:");

        System.out.println();

        // A string can contain numeric digits
        System.out.println("Letters in the Hawaiian alphabet: 12");

        // A numeric value can be concatenated to a string
        System.out.println("Dialing code for Antarctica: " + 672);

        System.out.println("Year in which Leonardo da Vinci invented "
                + "the parachute: " + 1515);

        System.out.println("Speed of ketchup: " + 40 + " km per year");
    }
}
```



```
We present the following facts for your extracurricular edification:

Letters in the Hawaiian alphabet: 12
Dialing code for Antarctica: 672
Year in which Leonardo da Vinci invented the parachute: 1515
Speed of ketchup: 40 km per year
```



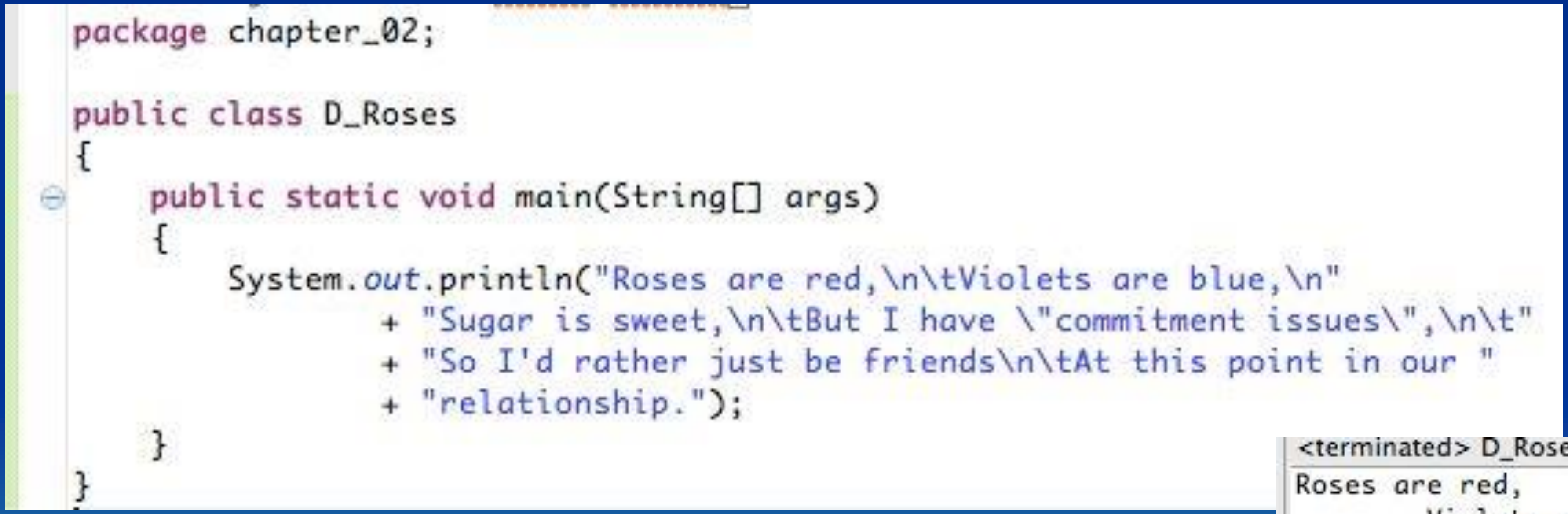
```java
package chapter_02;

public class C_Addition
{
    public static void main(String[] args)
    {
        System.out.println("24 and 45 concatenated: " + 24 + 45);
        System.out.println("24 and 45 added: " + (24 + 45));
    }
}
```



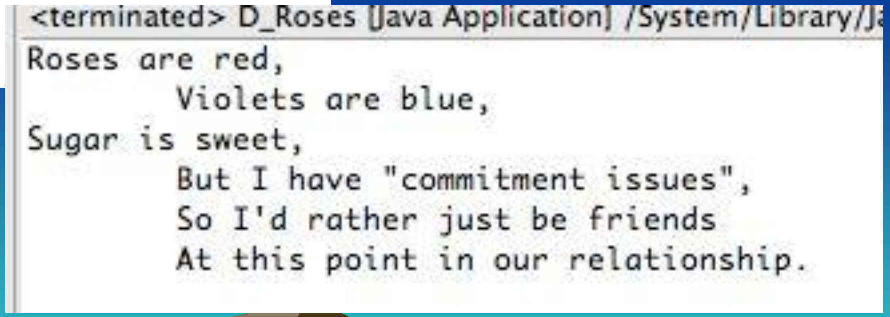
```
<terminated> C_Addition [Java Applicati
24 and 45 concatenated: 2445
24 and 45 added: 69
```

Escape Sequences

Escape Sequence	Meaning
\b	backspace
\t	tab
\n	newline
\r	carriage return
\"	double quote
\'	single quote
\\	backslash



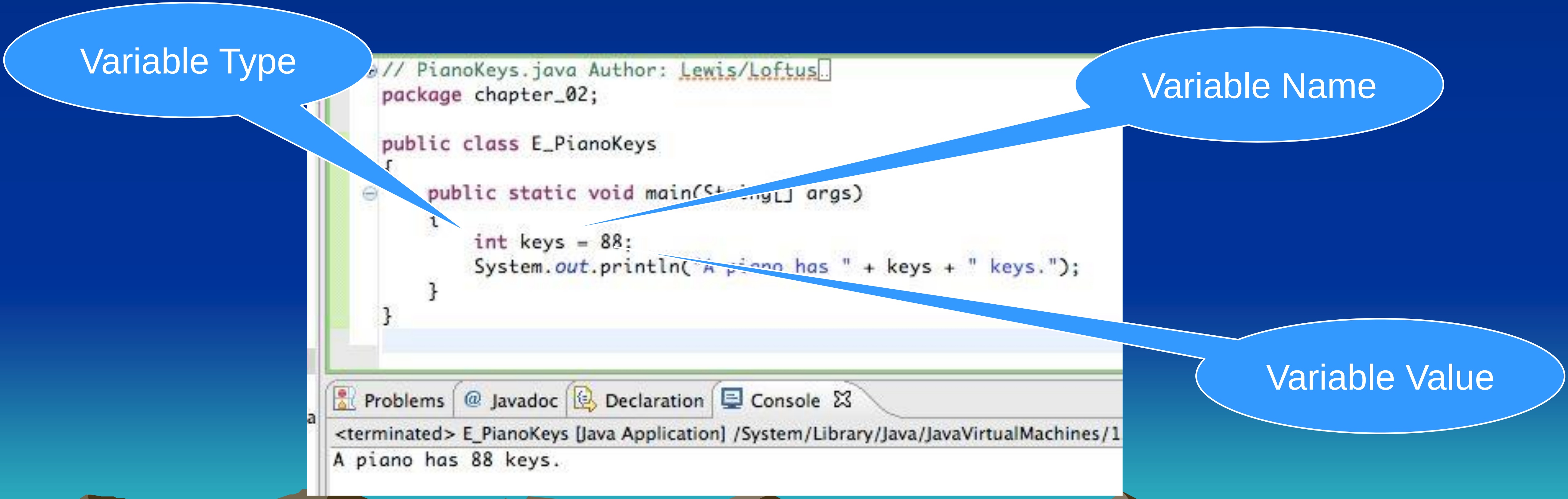
```java
package chapter_02;

public class D_Roses
{
    public static void main(String[] args)
    {
        System.out.println("Roses are red,\n\tViolets are blue,\n"
                + "Sugar is sweet,\n\tBut I have \"commitment issues\",\n\t"
                + "So I'd rather just be friends\n\tAt this point in our "
                + "relationship.");
    }
}
```



```
<terminated> D_Roses [Java Application] /System/Library/J
Roses are red,
        Violets are blue,
Sugar is sweet,
        But I have "commitment issues",
        So I'd rather just be friends
        At this point in our relationship.
```

Variables

- A variable is a name for a location in memory that holds a value
- A variable declaration specifies the variable's name and the type of information that it will hold
- When a variable is referenced in a program, its current value is used

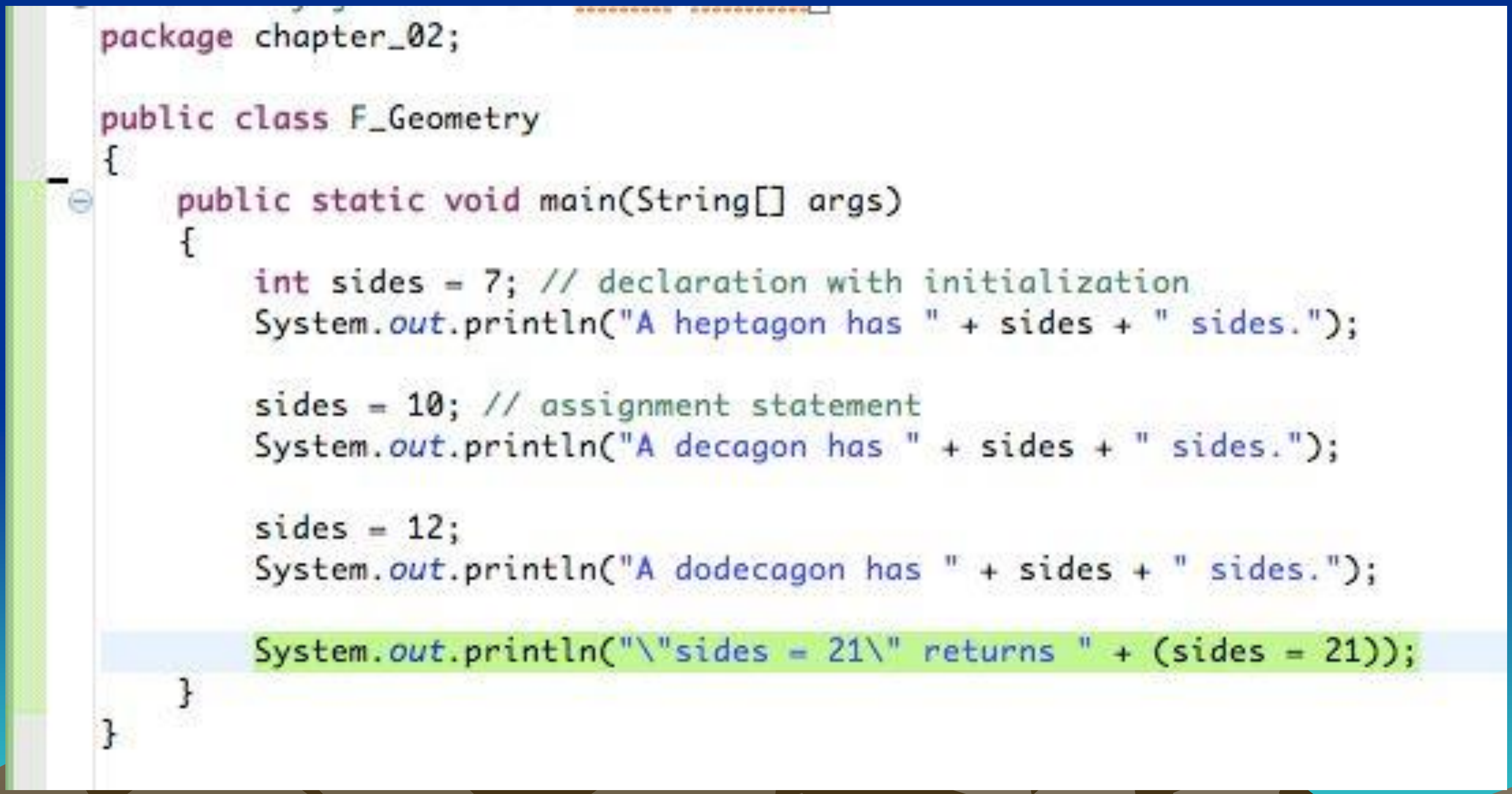


```
// PianoKeys.java Author: Lewis/Loftus
package chapter_02;

public class E_PianoKeys
{
    public static void main(String[] args)
    {
        int keys = 88;
        System.out.println("A piano has " + keys + " keys.");
    }
}
```

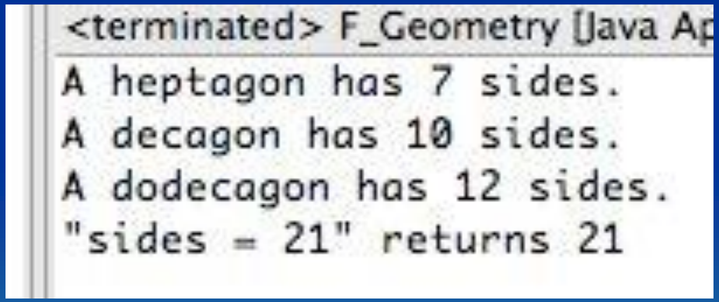
```
<terminated> E_PianoKeys [Java Application] /System/Library/Java/JavaVirtualMachines/1
A piano has 88 keys.
```

Assignment

- The assignment operator is the = sign
  - Example: total = 50;
- You can only assign a value to a variable that is consistent with the variable's declared type
- The assign statement returns the value stored in the variable



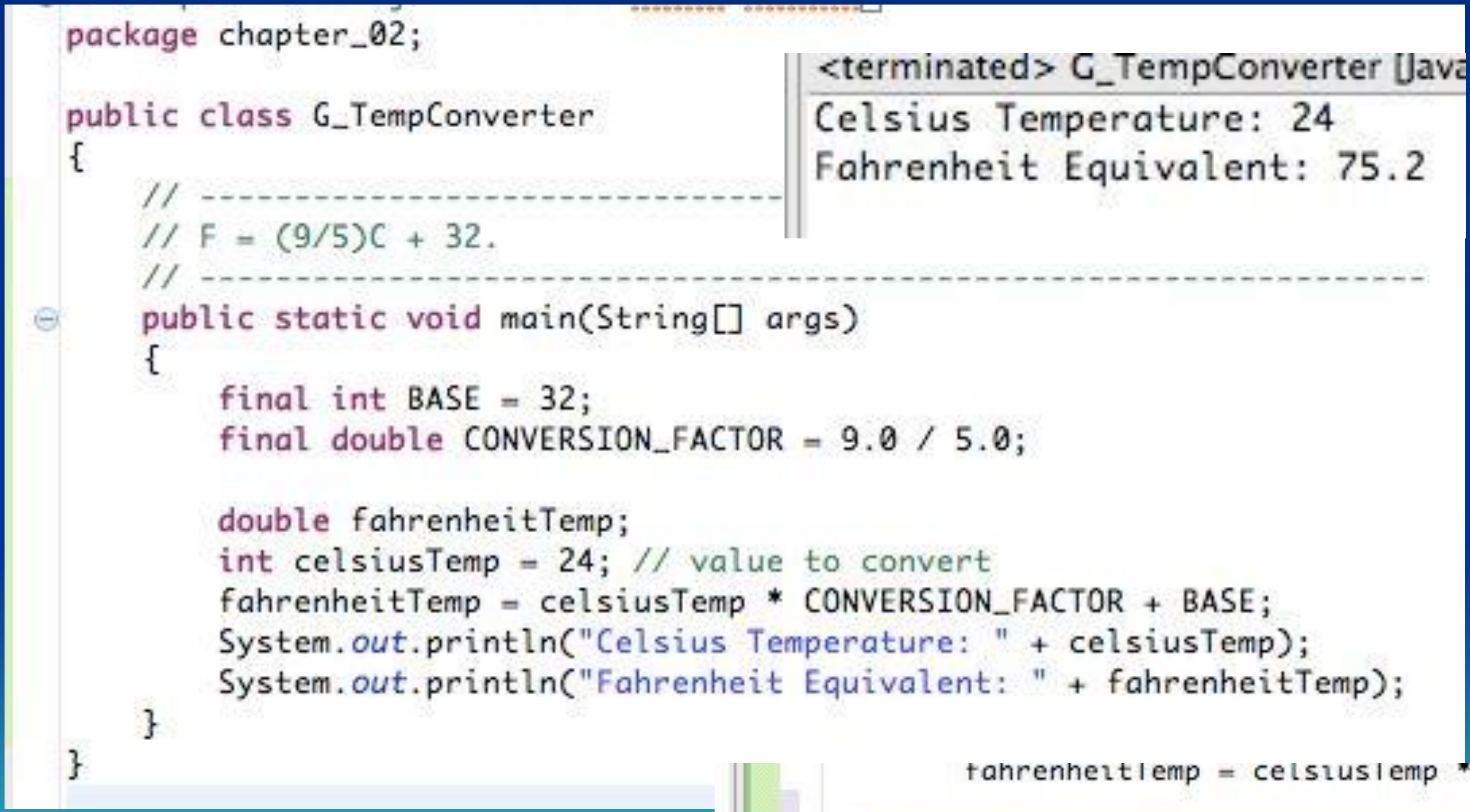
```
package chapter_02;

public class F_Geometry
{
    public static void main(String[] args)
    {
        int sides = 7; // declaration with initialization
        System.out.println("A heptagon has " + sides + " sides.");

        sides = 10; // assignment statement
        System.out.println("A decagon has " + sides + " sides.");

        sides = 12;
        System.out.println("A dodecagon has " + sides + " sides.");

        System.out.println("\"sides = 21\" returns " + (sides = 21));
    }
}
```



```
<terminated> F_Geometry [Java Ap
A heptagon has 7 sides.
A decagon has 10 sides.
A dodecagon has 12 sides.
"sides = 21" returns 21
```

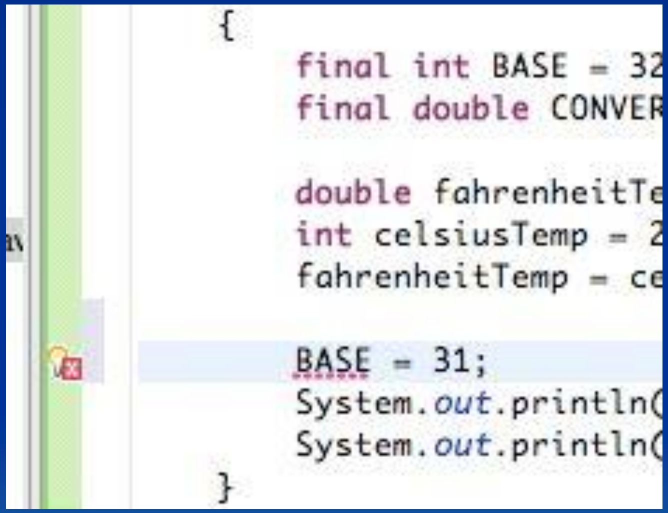
Constants

- A constant is an identifier that is similar to a variable except that it holds the same value during its entire existence
- The compiler will issue an error if you try to change the value of a constant
- In Java, we use the final modifier to declare a constant
    - final int MIN_HEIGHT = 69;

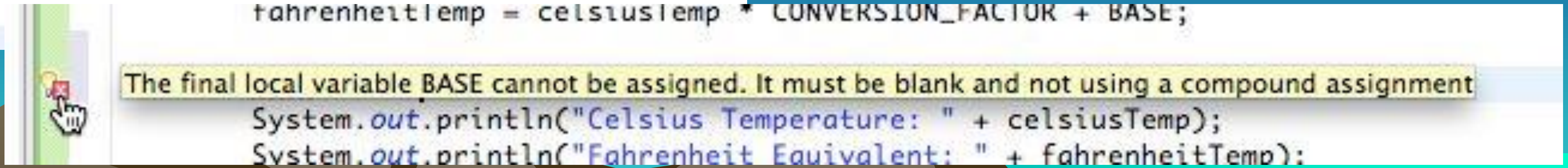


```
package chapter_02;

public class G_TempConverter
{
    // ------------------------------------
    // F = (9/5)C + 32.
    // ------------------------------------------------------------
    public static void main(String[] args)
    {
        final int BASE = 32;
        final double CONVERSION_FACTOR = 9.0 / 5.0;

        double fahrenheitTemp;
        int celsiusTemp = 24; // value to convert
        fahrenheitTemp = celsiusTemp * CONVERSION_FACTOR + BASE;
        System.out.println("Celsius Temperature: " + celsiusTemp);
        System.out.println("Fahrenheit Equivalent: " + fahrenheitTemp);
    }
}
```

```
<terminated> G_TempConverter [Java
Celsius Temperature: 24
Fahrenheit Equivalent: 75.2
```



```
{
    final int BASE = 32
    final double CONVER

    double fahrenheitTe
    int celsiusTemp = 2
    fahrenheitTemp = ce

    BASE = 31;
    System.out.println(
    System.out.println(
}
```



```
fahrenheitTemp = celsiusTemp * CONVERSION_FACTOR + BASE;
The final local variable BASE cannot be assigned. It must be blank and not using a compound assignment
System.out.println("Celsius Temperature: " + celsiusTemp);
System.out.println("Fahrenheit Equivalent: " + fahrenheitTemp);
```

Primitive Data

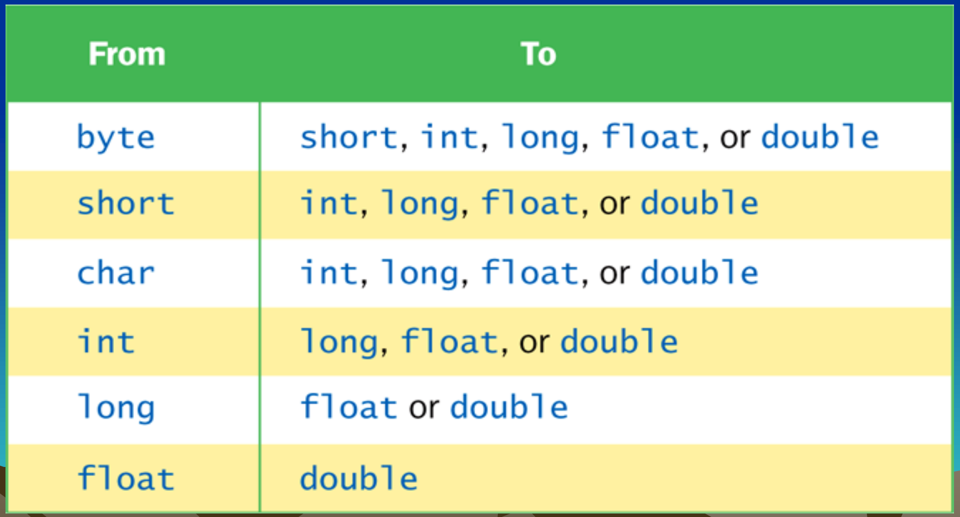
- There are eight primitive data types in Java
  - Integers: byte, short, int, long
  - Floating numbers: float, double
  - Characters: char
  - Boolean: boolean

Type	Storage	Min Value	Max Value
byte	8 bits	-128	127
short	16 bits	-32,768	32,767
int	32 bits	-2,147,483,648	2,147,483,647
long	64 bits	$< -9 \times 10^{18}$	$> 9 \times 10^{18}$
float	32 bits	+/- 3.4 x 10^{38} with 7 significant digits	
double	64 bits	+/- 1.7 x 10^{308} with 15 significant digits	

Expressions, operator precedence, and data conversions

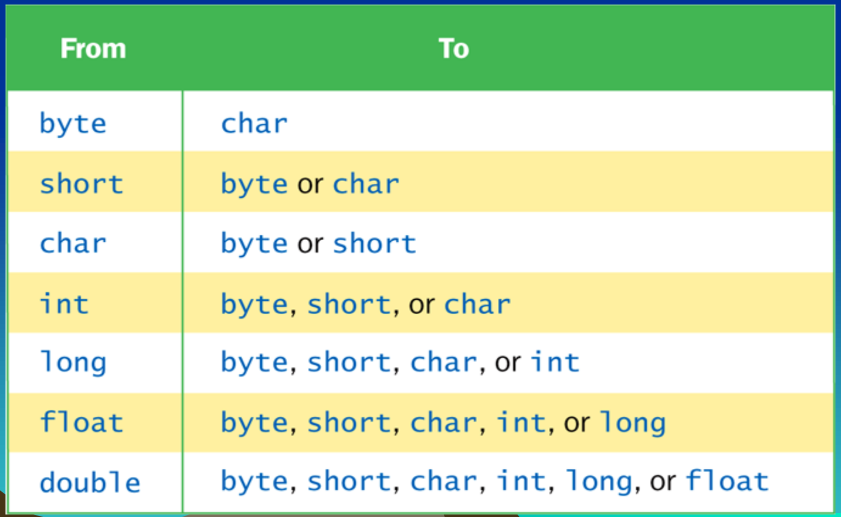
- Expressions, operator precedence, and data conversions work exactly the way they do in C, C++.
- Take Home Reading: chapter 2 pages 75-86.
- In Java, conversion occurs in three ways:
  - Assignment conversion,
  - promotion (x = double/integer, integer will be promoted to double)
  - casting (most used, (new type) variable name)

Widening Conversions



From	To
byte	short, int, long, float, or double
short	int, long, float, or double
char	int, long, float, or double
int	long, float, or double
long	float or double
float	double

Narrowing Conversions

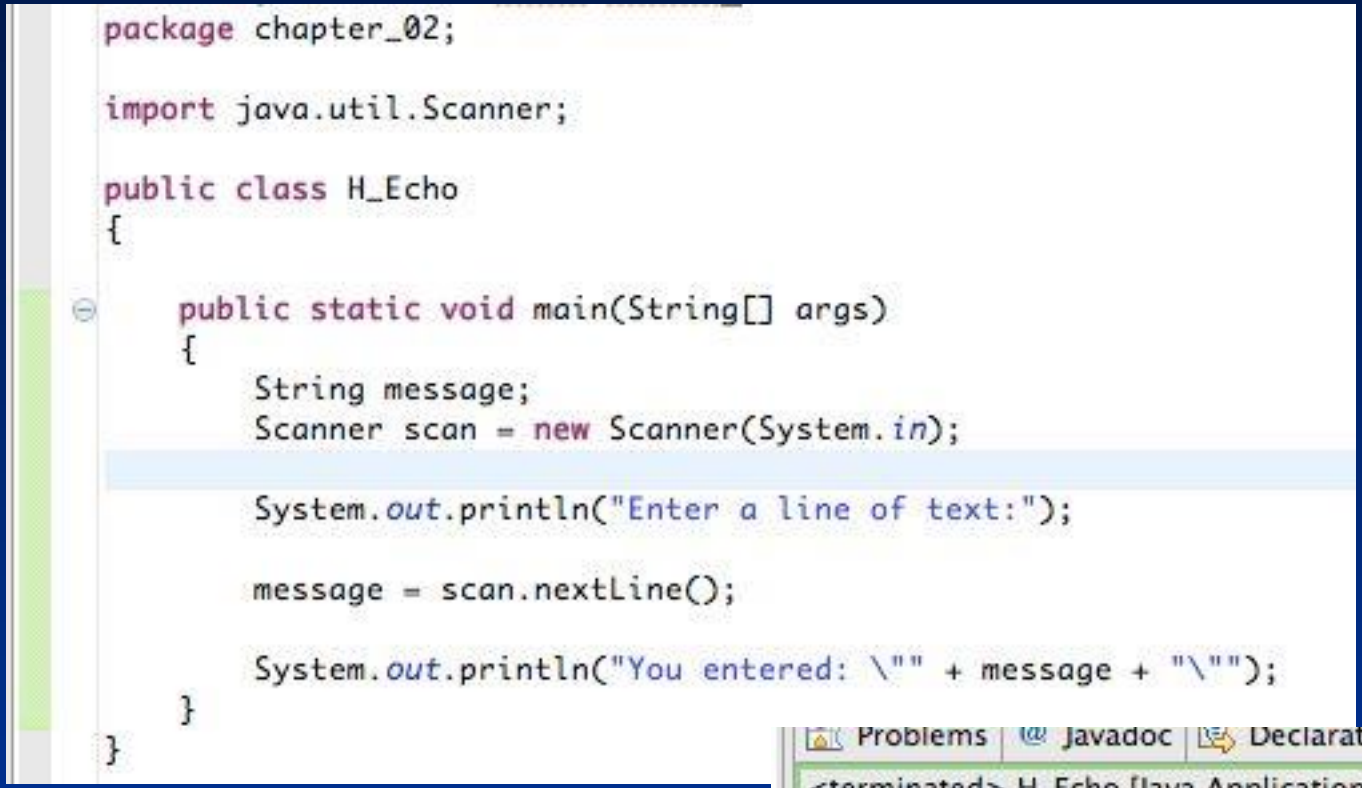


From	To
byte	char
short	byte or char
char	byte or short
int	byte, short, or char
long	byte, short, char, or int
float	byte, short, char, int, or long
double	byte, short, char, int, long, or float

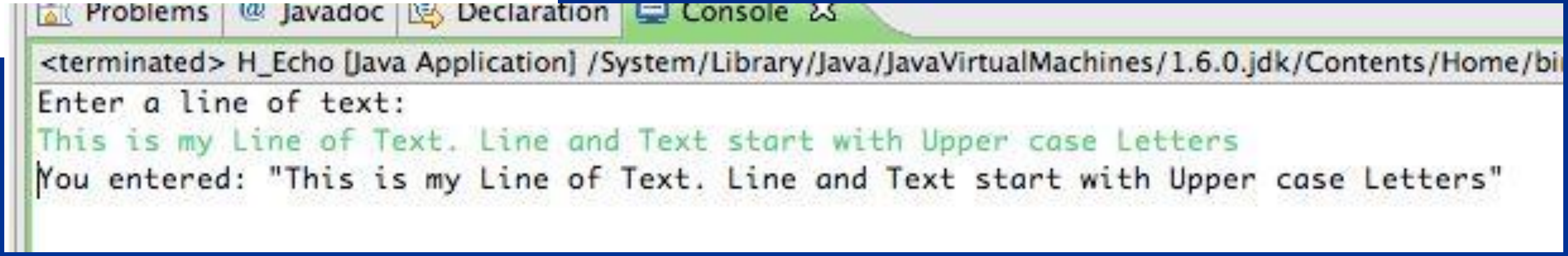
Interactive Programs and Scanner

- Programs generally need input on which to operate
- The Scanner class provides convenient methods for reading input values of various types
- A Scanner object can be set up to read input from various sources
- Keyboard input is represented by the System.in object
- The following line creates a Scanner object that reads from the keyboard:
    - Scanner scan = new Scanner (System.in);
- Once created, the Scanner object can be used to invoke various input methods, such as:
    - answer = scan.nextLine();

The Scanner Class

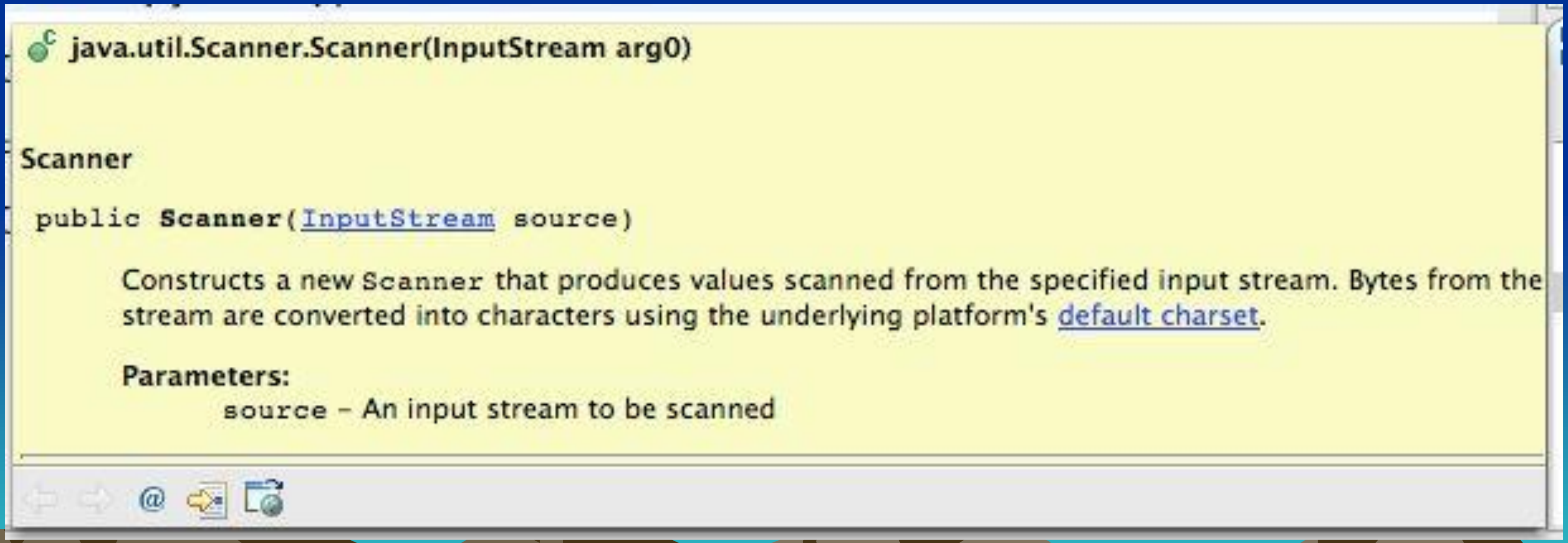


```
package chapter_02;

import java.util.Scanner;

public class H_Echo
{

    public static void main(String[] args)
    {
        String message;
        Scanner scan = new Scanner(System.in);

        System.out.println("Enter a line of text:");

        message = scan.nextLine();

        System.out.println("You entered: \"" + message + "\"");
    }
}
```



Problems | @ Javadoc | Declaration | Console

```
<terminated> H_Echo [Java Application] /System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home/bi
Enter a line of text:
This is my Line of Text. Line and Text start with Upper case Letters
You entered: "This is my Line of Text. Line and Text start with Upper case Letters"
```



java.util.Scanner.Scanner(InputStream arg0)

Scanner

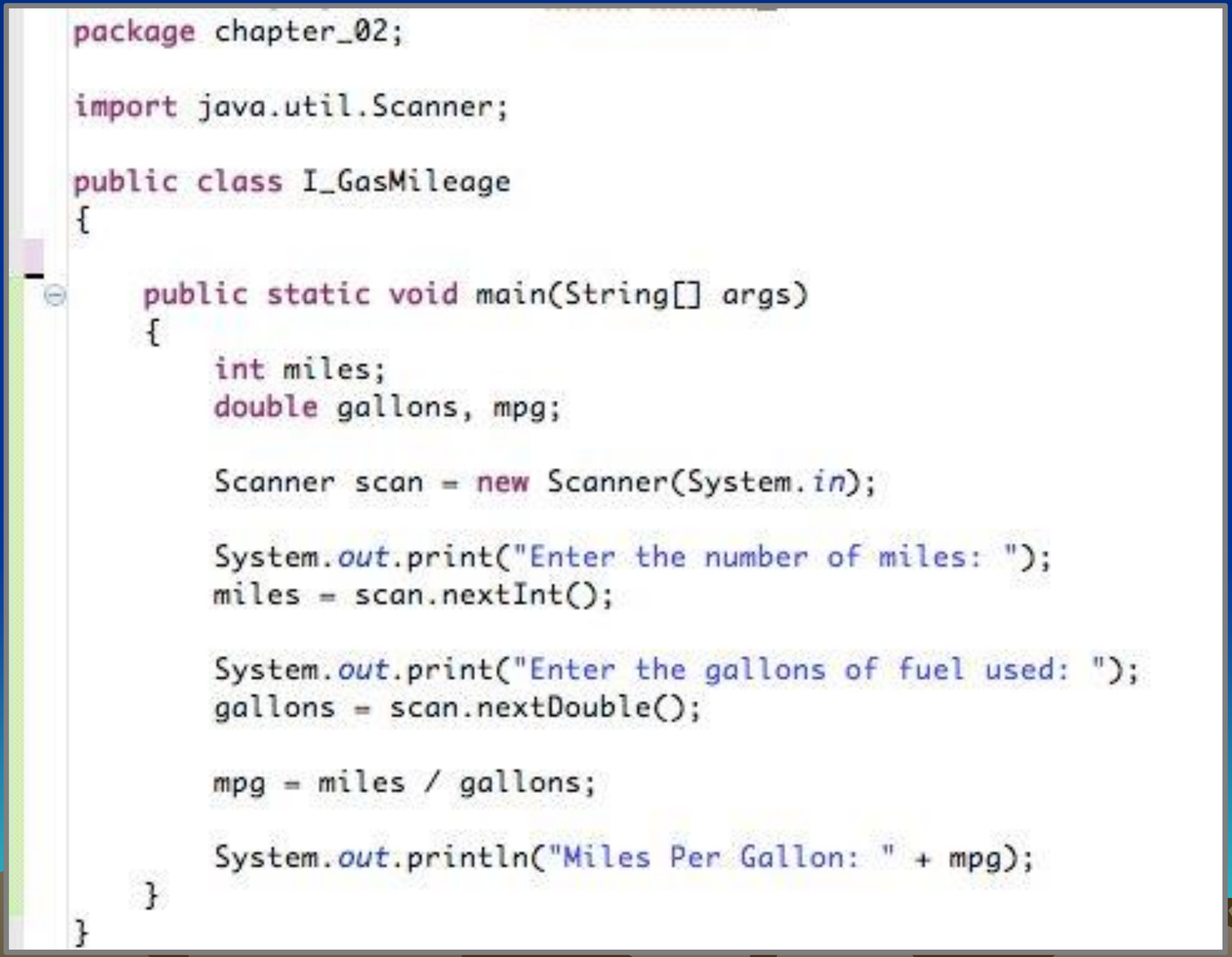
`public Scanner(InputStream source)`

Constructs a new `Scanner` that produces values scanned from the specified input stream. Bytes from the stream are converted into characters using the underlying platform's default charset.

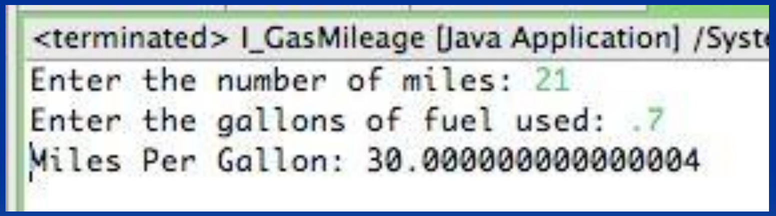
Parameters:
`source` - An input stream to be scanned

Input Tokens

- The next method of the Scanner class reads the next input token and returns it as a string
- Methods such as nextInt and nextDouble read data of particular types

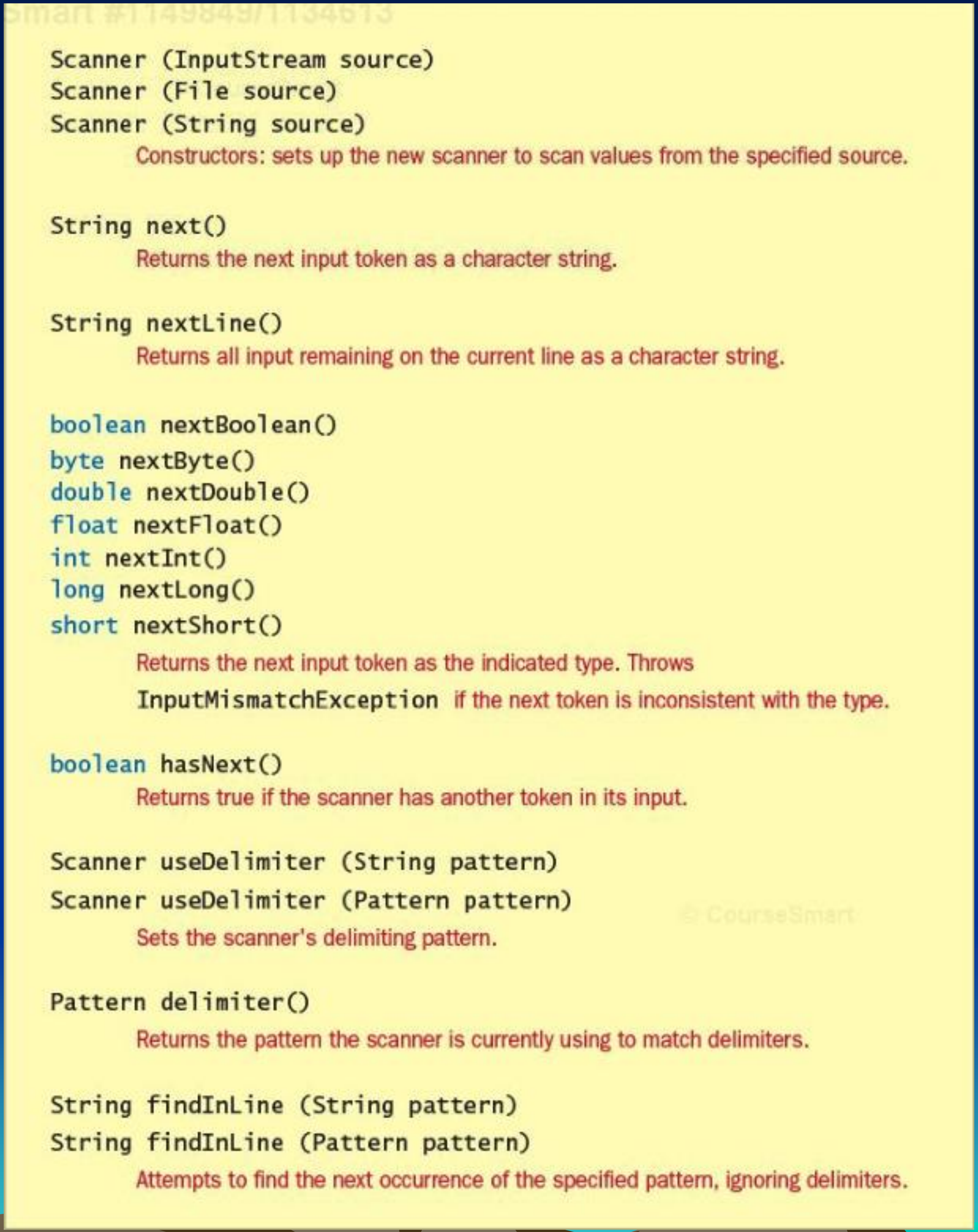


```
package chapter_02;

import java.util.Scanner;

public class I_GasMileage
{

    public static void main(String[] args)
    {
        int miles;
        double gallons, mpg;

        Scanner scan = new Scanner(System.in);

        System.out.print("Enter the number of miles: ");
        miles = scan.nextInt();

        System.out.print("Enter the gallons of fuel used: ");
        gallons = scan.nextDouble();

        mpg = miles / gallons;

        System.out.println("Miles Per Gallon: " + mpg);
    }
}
```



```
<terminated> I_GasMileage [Java Application] /Syst
Enter the number of miles: 21
Enter the gallons of fuel used: .7
Miles Per Gallon: 30.000000000000004
```

More on Scanner



```
Scanner (InputStream source)
Scanner (File source)
Scanner (String source)
```
Constructors: sets up the new scanner to scan values from the specified source.

```
String next()
```
Returns the next input token as a character string.

```
String nextLine()
```
Returns all input remaining on the current line as a character string.

```
boolean nextBoolean()
byte nextByte()
double nextDouble()
float nextFloat()
int nextInt()
long nextLong()
short nextShort()
```
Returns the next input token as the indicated type. Throws `InputMismatchException` if the next token is inconsistent with the type.

```
boolean hasNext()
```
Returns true if the scanner has another token in its input.

```
Scanner useDelimiter (String pattern)
Scanner useDelimiter (Pattern pattern)
```
Sets the scanner's delimiting pattern.

```
Pattern delimiter()
```
Returns the pattern the scanner is currently using to match delimiters.

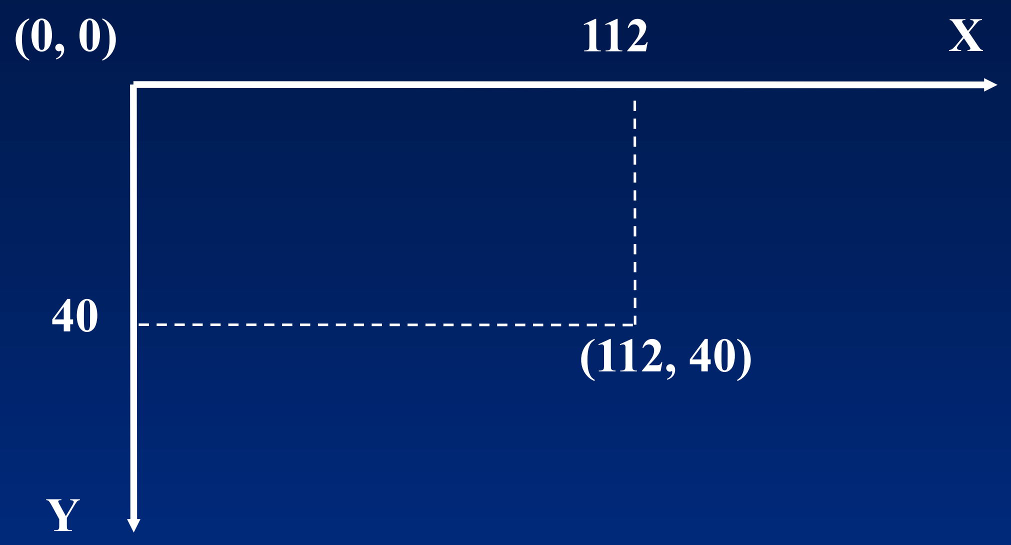
```
String findInLine (String pattern)
String findInLine (Pattern pattern)
```
Attempts to find the next occurrence of the specified pattern, ignoring delimiters.

Introduction to Graphics

- The last few sections of each chapter of the textbook focus on graphics and graphical user interfaces
- A picture or drawing must be digitized for storage on a computer
- A picture is made up of pixels (picture elements), and each pixel is stored separately
- The number of pixels used to represent a picture is called the picture resolution
- The number of pixels that can be displayed by a monitor is called the monitor resolution

Coordinate Systems and Representing Color

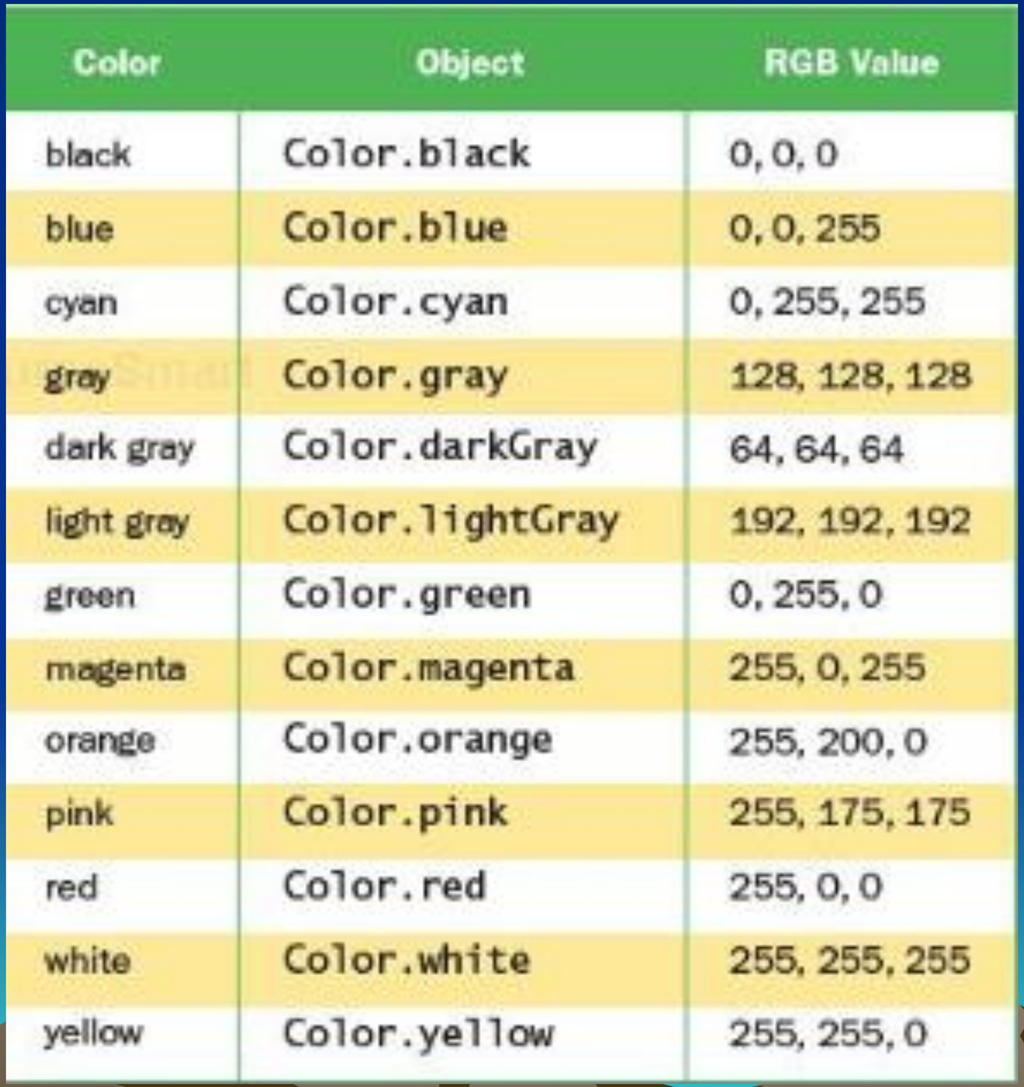
- Each pixel can be identified using a two-dimensional coordinate system
- When referring to a pixel in a Java program, we use a coordinate system with the origin in the top-left corner



- A black and white picture could be stored using one bit per pixel (0 = white and 1 = black)
- A colored picture requires more information
- RGB Color System: Every color can be represented as a mixture of the three additive primary colors Red, Green, and Blue

The Color Class

- A color in a Java program is represented as an object created from the Color class
- The Color class also contains several predefined colors, including the following:

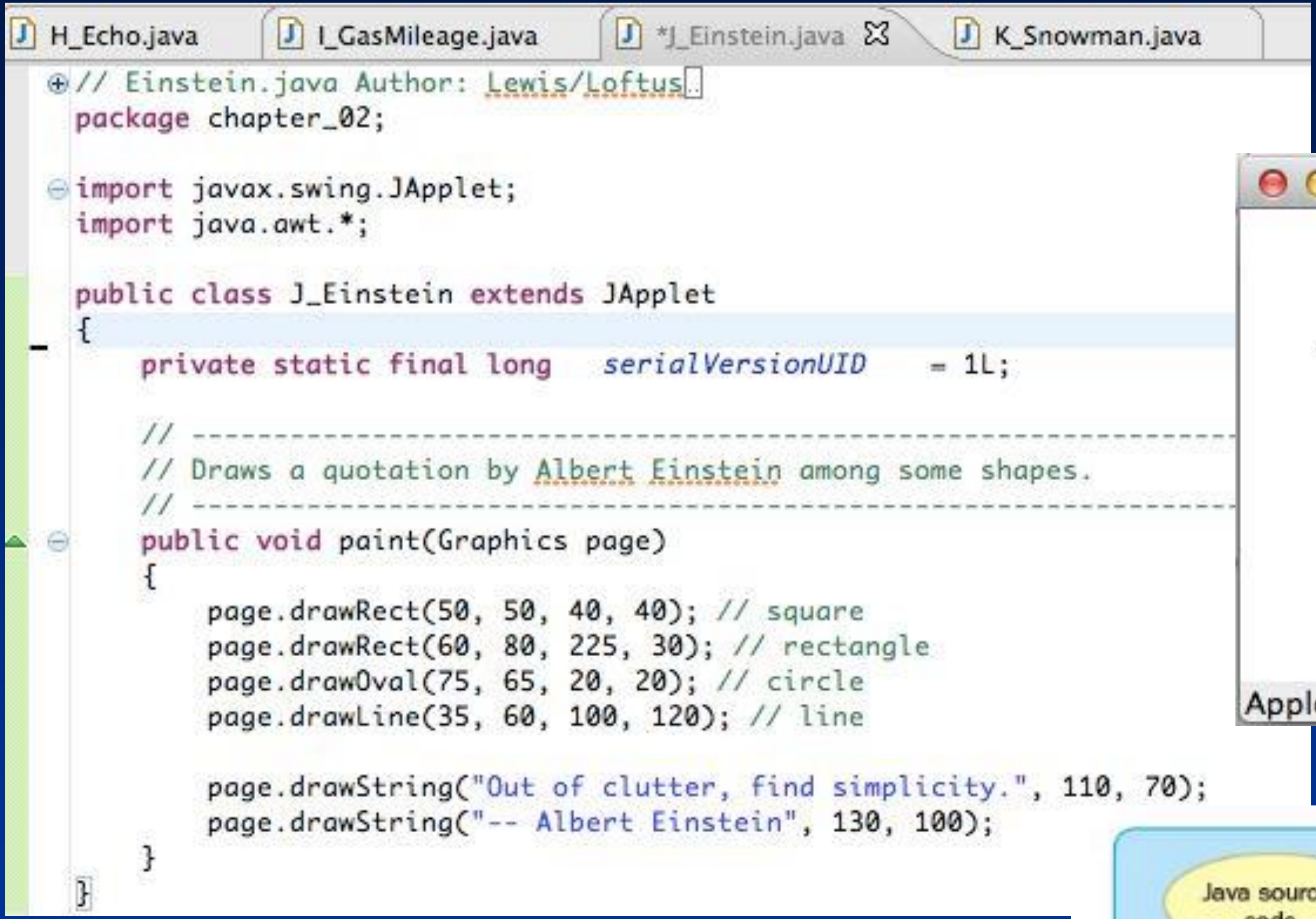


Color	Object	RGB Value
black	`Color.black`	0, 0, 0
blue	`Color.blue`	0, 0, 255
cyan	`Color.cyan`	0, 255, 255
gray	`Color.gray`	128, 128, 128
dark gray	`Color.darkGray`	64, 64, 64
light gray	`Color.lightGray`	192, 192, 192
green	`Color.green`	0, 255, 0
magenta	`Color.magenta`	255, 0, 255
orange	`Color.orange`	255, 200, 0
pink	`Color.pink`	255, 175, 175
red	`Color.red`	255, 0, 0
white	`Color.white`	255, 255, 255
yellow	`Color.yellow`	255, 255, 0

Applets

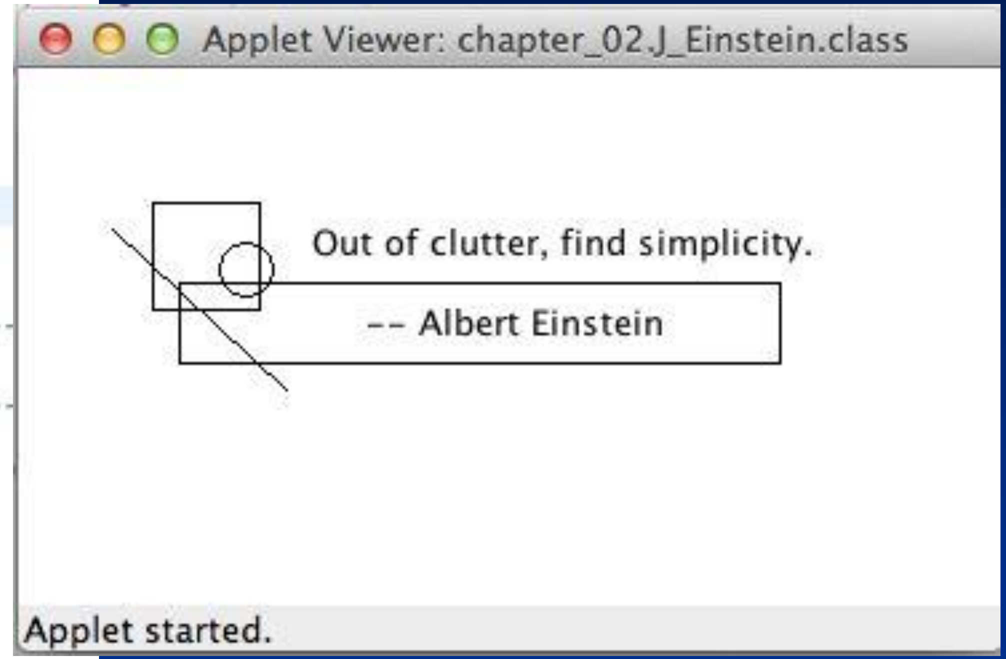
- A Java application is a stand-alone program with a main method
- A Java applet is a program that is intended to be transported over the Web and executed using a web browser
- An applet doesn't have a main method
- An applet also can be executed using the applet viewer tool of the Java SDK

Einstein



H_Echo.java | I_GasMileage.java | *J_Einstein.java | K_Snowman.java

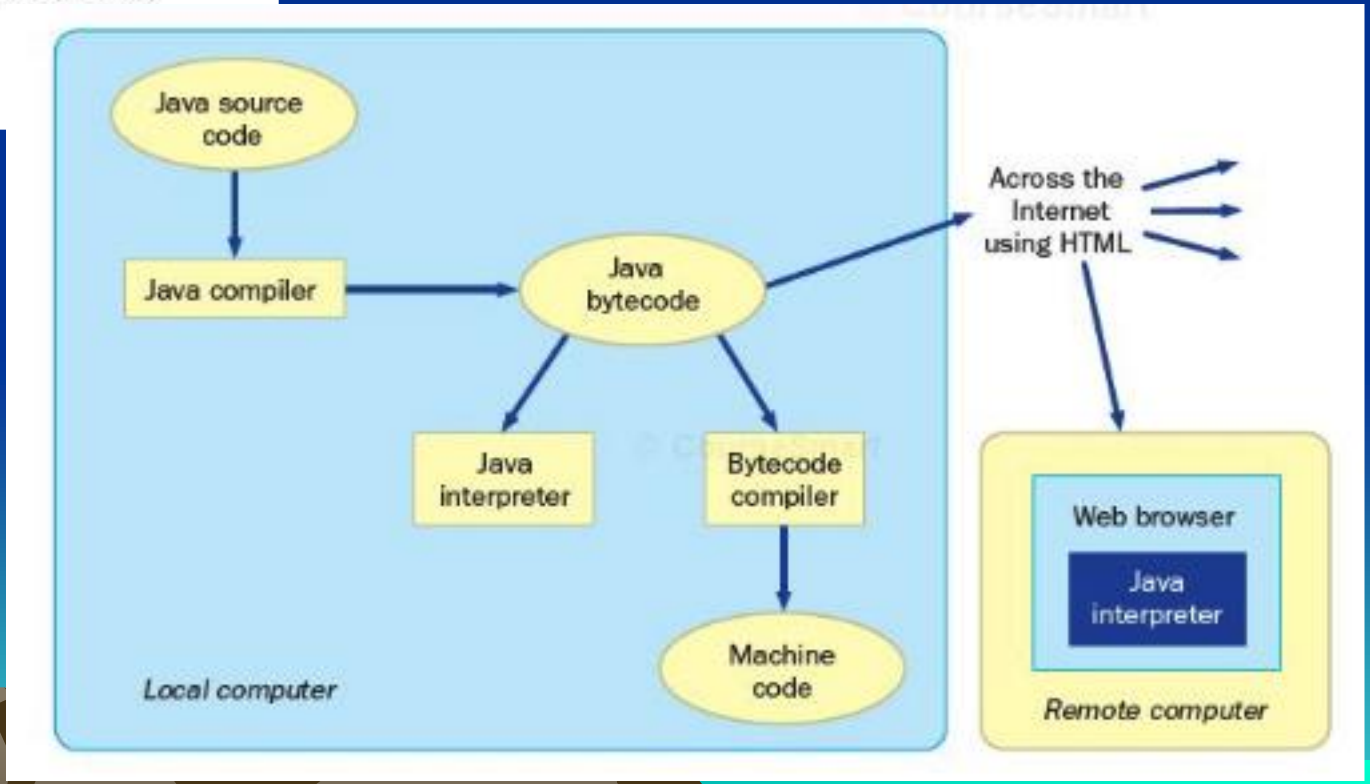
```
// Einstein.java Author: Lewis/Loftus
package chapter_02;

import javax.swing.JApplet;
import java.awt.*;

public class J_Einstein extends JApplet
{
    private static final long   serialVersionUID   = 1L;

    // -----------------------------------------------------------------
    // Draws a quotation by Albert Einstein among some shapes.
    // -----------------------------------------------------------------
    public void paint(Graphics page)
    {
        page.drawRect(50, 50, 40, 40); // square
        page.drawRect(60, 80, 225, 30); // rectangle
        page.drawOval(75, 65, 20, 20); // circle
        page.drawLine(35, 60, 100, 120); // line

        page.drawString("Out of clutter, find simplicity.", 110, 70);
        page.drawString("-- Albert Einstein", 130, 100);
    }
}
```



Applet Viewer: chapter_02.J_Einstein.class

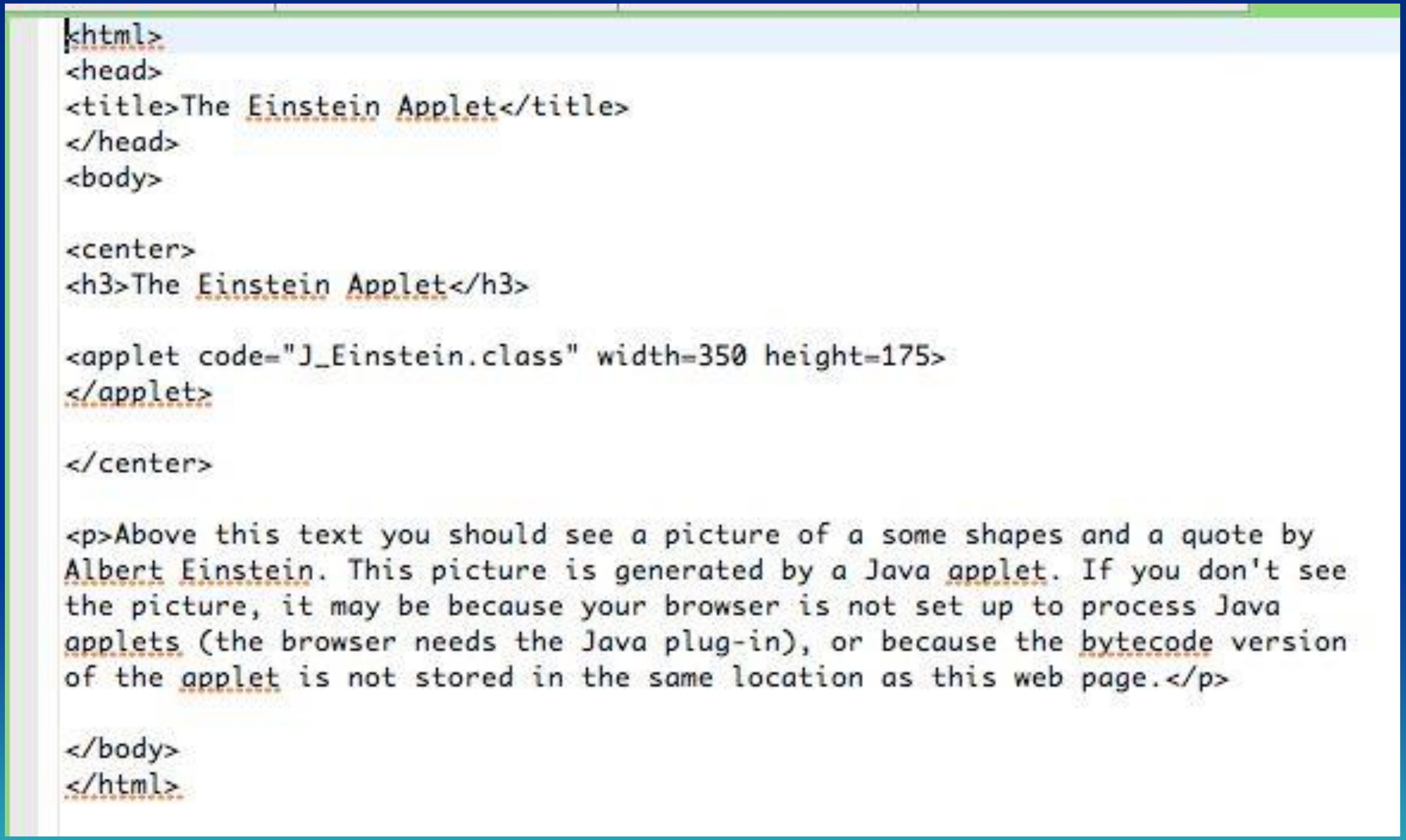
```
Out of clutter, find simplicity.
-- Albert Einstein
```

Applet started.



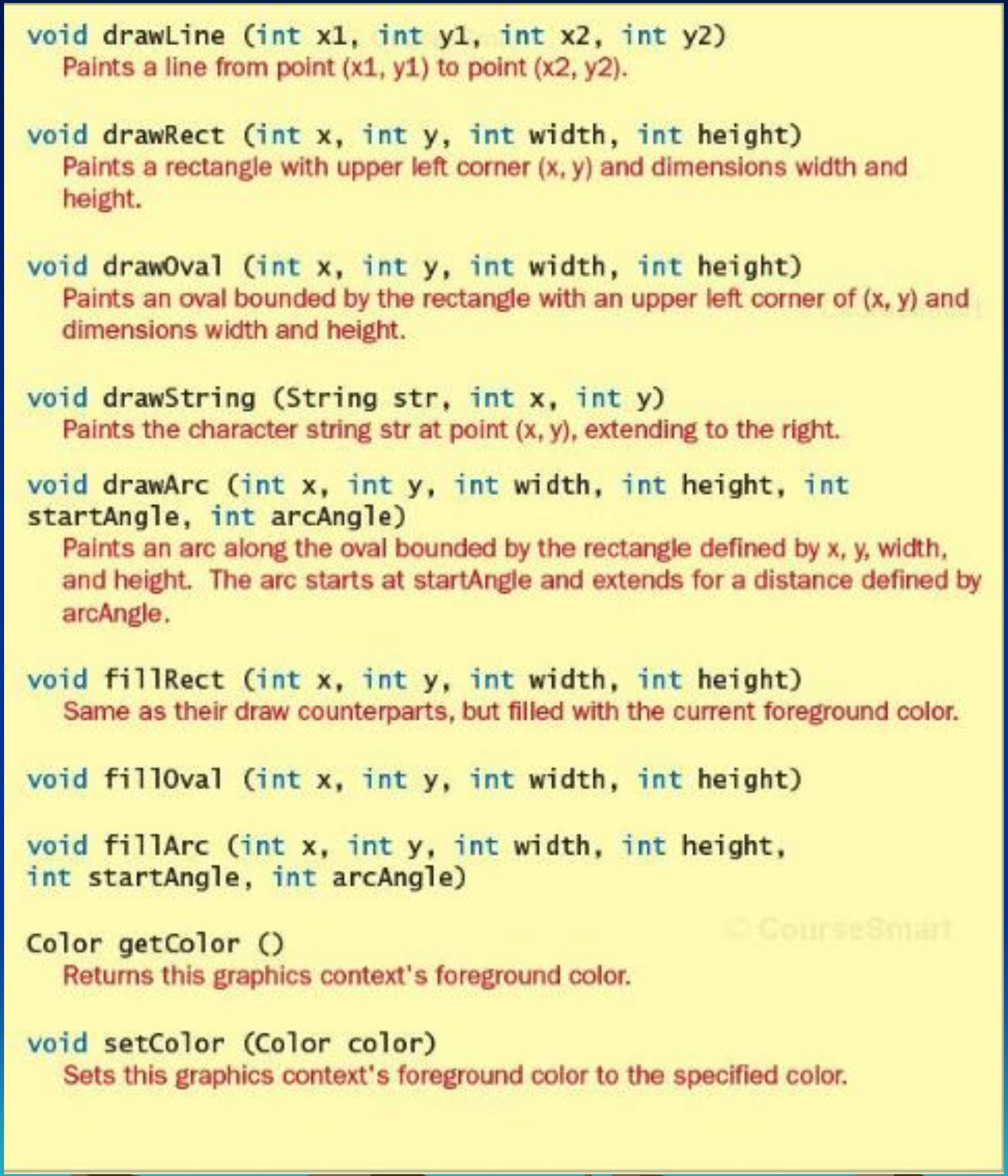
The HTML applet Tag

- An applet is embedded into an HTML file using a tag that references the bytecode file of the applet
- The bytecode version of the program is transported across the web and executed by a Java interpreter that is part of the browser



```
<html>
<head>
<title>The Einstein Applet</title>
</head>
<body>

<center>
<h3>The Einstein Applet</h3>

<applet code="J_Einstein.class" width=350 height=175>
</applet>

</center>

<p>Above this text you should see a picture of a some shapes and a quote by
Albert Einstein. This picture is generated by a Java applet. If you don't see
the picture, it may be because your browser is not set up to process Java
applets (the browser needs the Java plug-in), or because the bytecode version
of the applet is not stored in the same location as this web page.</p>

</body>
</html>
```

Drawing Shapes



```
void drawLine (int x1, int y1, int x2, int y2)
```
Paints a line from point (x1, y1) to point (x2, y2).

```
void drawRect (int x, int y, int width, int height)
```
Paints a rectangle with upper left corner (x, y) and dimensions width and height.

```
void drawOval (int x, int y, int width, int height)
```
Paints an oval bounded by the rectangle with an upper left corner of (x, y) and dimensions width and height.

```
void drawString (String str, int x, int y)
```
Paints the character string str at point (x, y), extending to the right.

```
void drawArc (int x, int y, int width, int height, int
startAngle, int arcAngle)
```
Paints an arc along the oval bounded by the rectangle defined by x, y, width, and height. The arc starts at startAngle and extends for a distance defined by arcAngle.

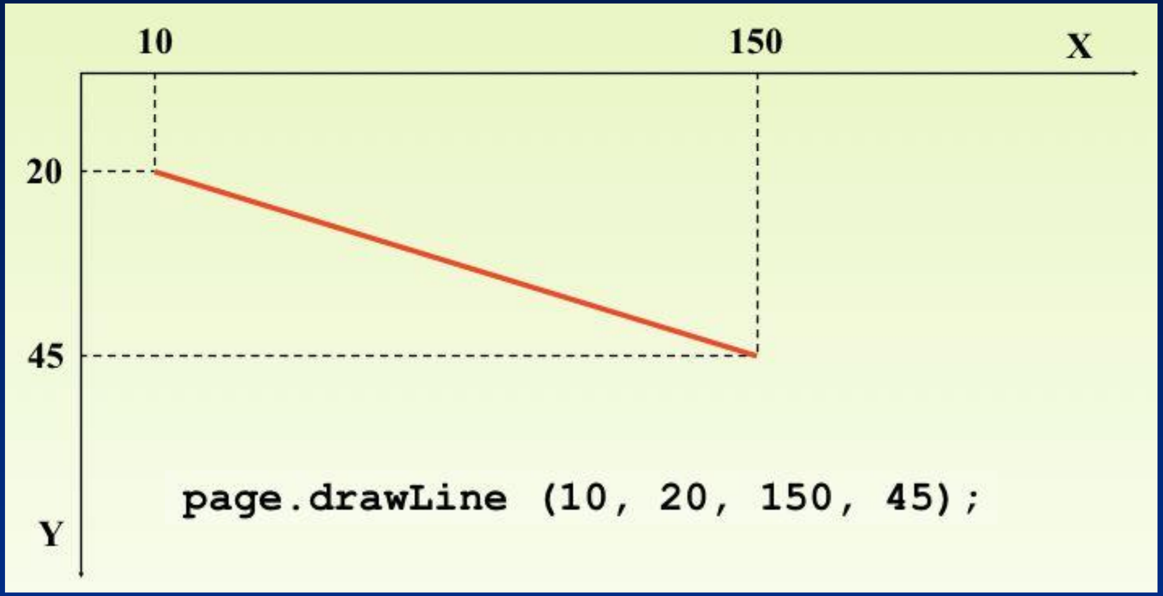
```
void fillRect (int x, int y, int width, int height)
```
Same as their draw counterparts, but filled with the current foreground color.

```
void fillOval (int x, int y, int width, int height)

void fillArc (int x, int y, int width, int height,
int startAngle, int arcAngle)

Color getColor ()
```
Returns this graphics context's foreground color.

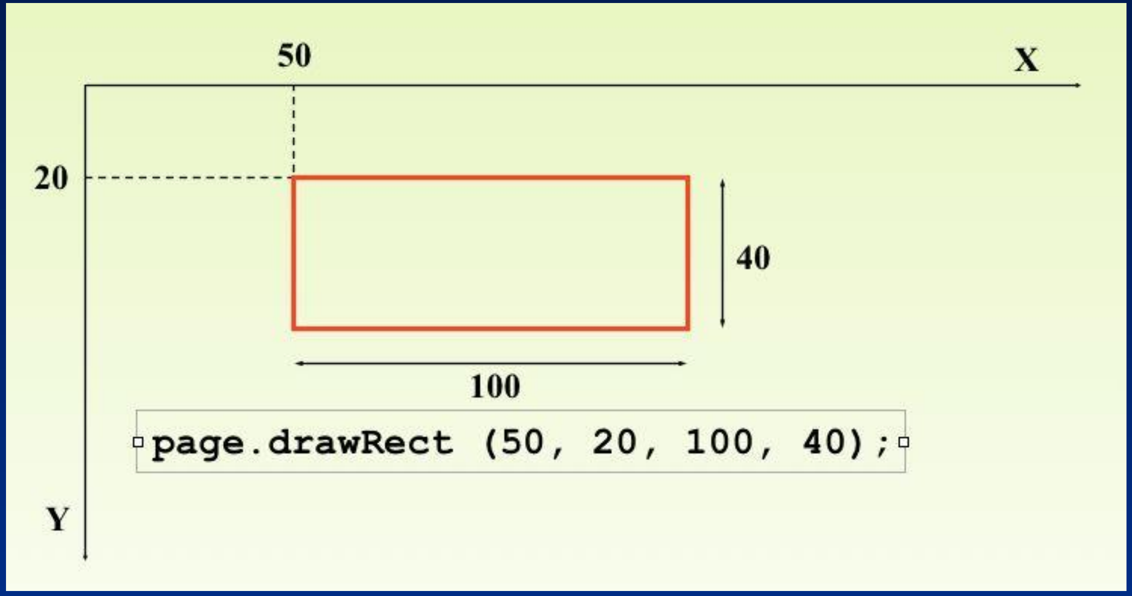
```
void setColor (Color color)
```
Sets this graphics context's foreground color to the specified color.

- A shape can be filled or unfilled, depending on which method is invoked
- The method parameters specify coordinates and sizes
- Shapes with curves, like an oval, are usually drawn by specifying the shape's bounding rectangle
- An arc can be thought of as a section of an oval

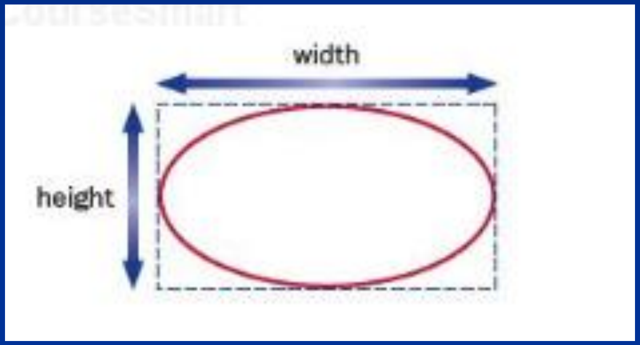
Drawing Shapes

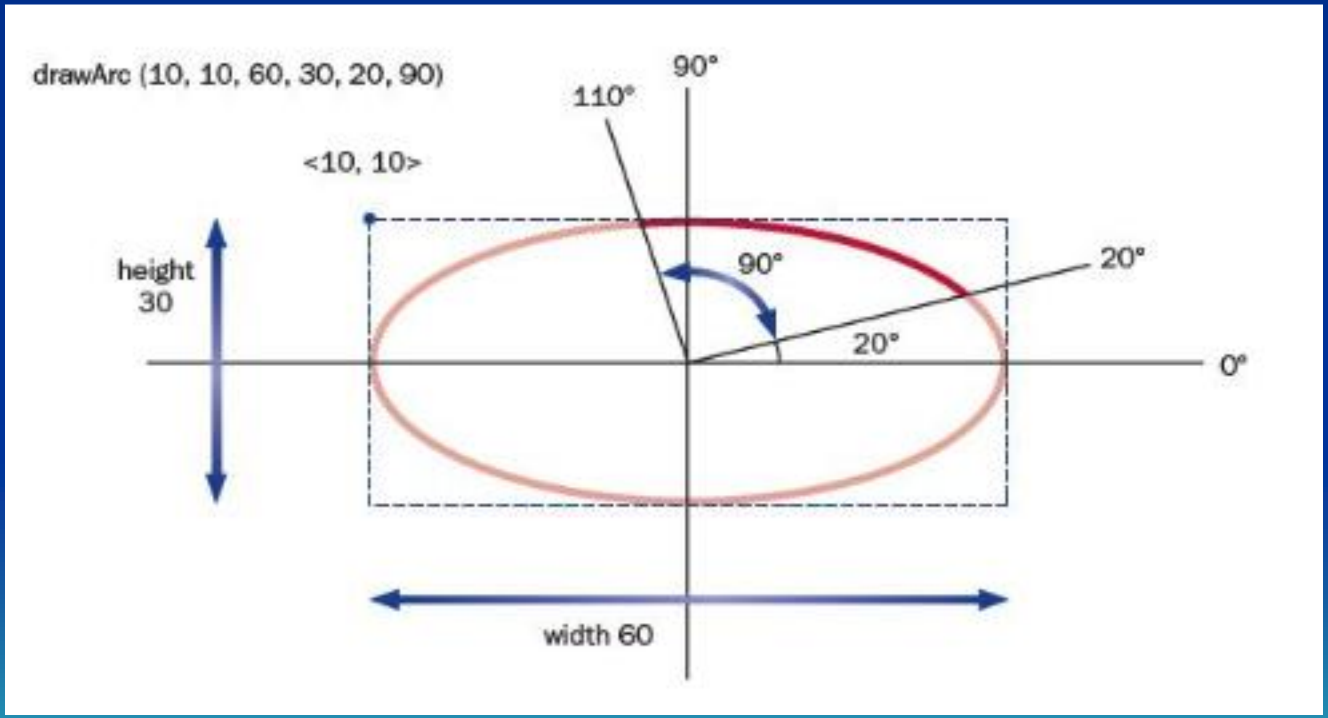


```
page.drawLine (10, 20, 150, 45);
```



```
page.drawRect (50, 20, 100, 40);
```

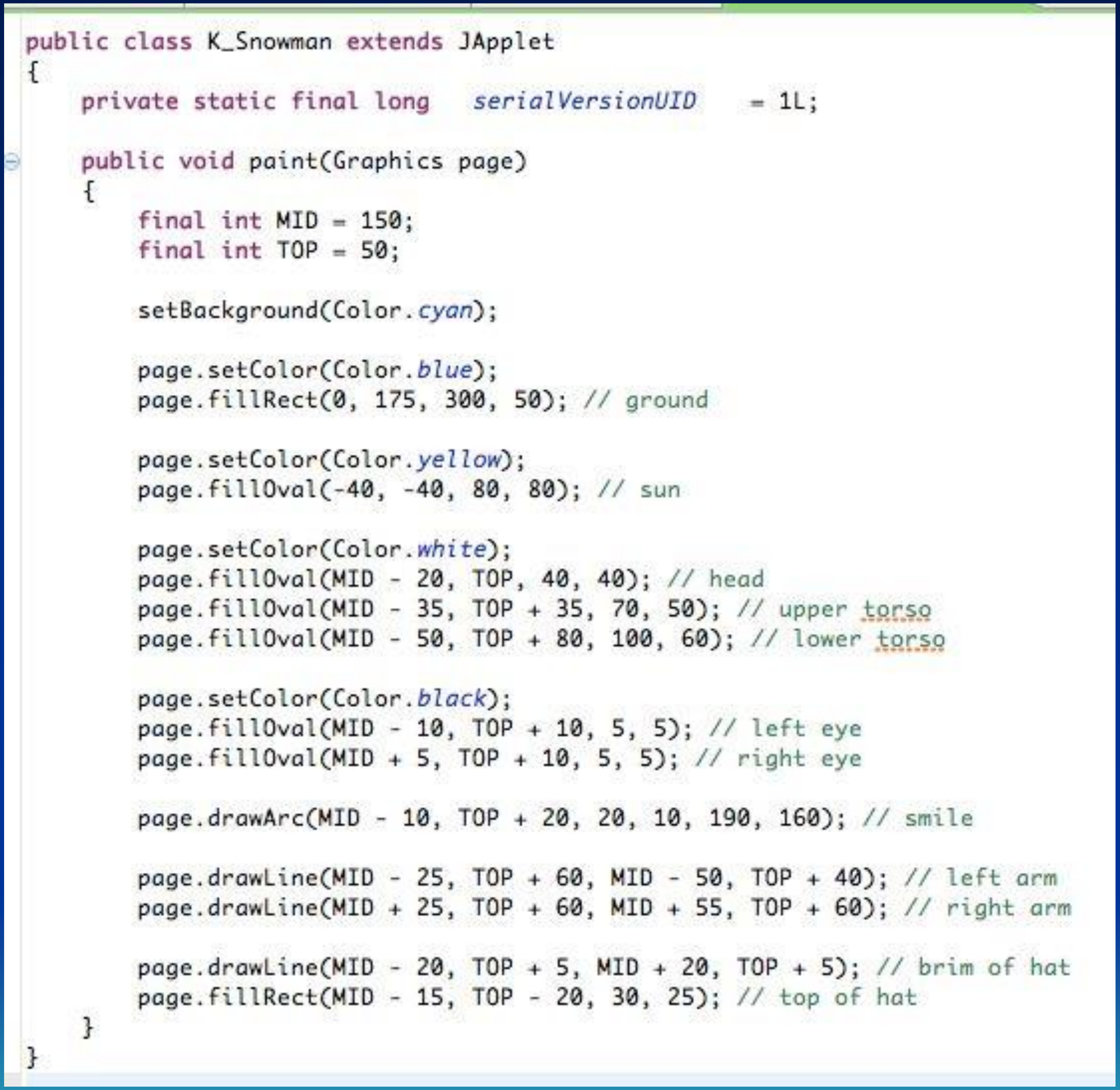




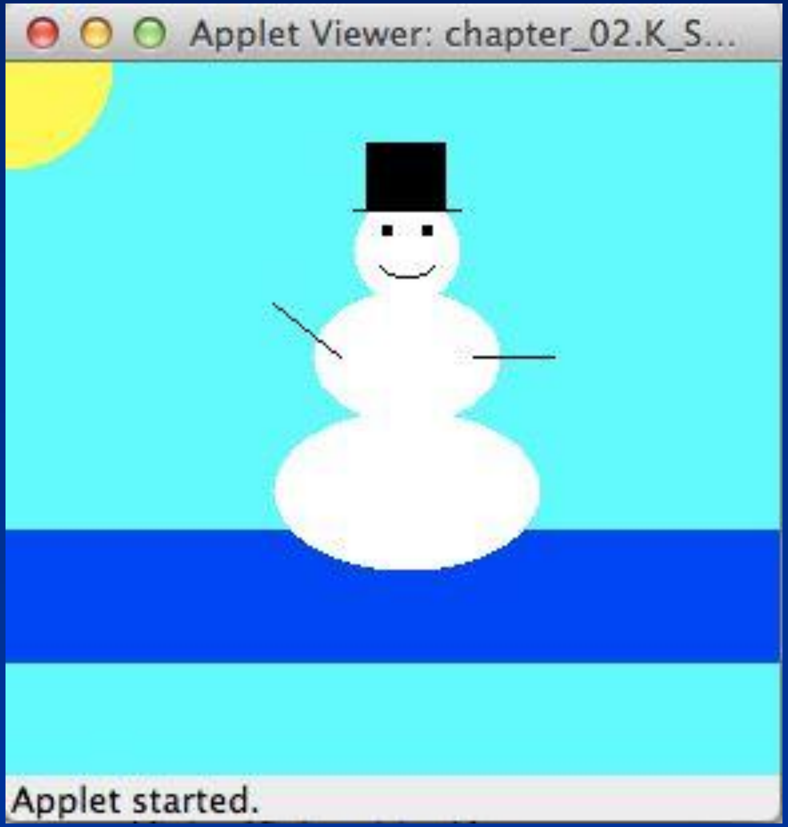
```
drawArc (10, 10, 60, 30, 20, 90)
```

- Every drawing surface has a background color
- Every graphics context has a current foreground color

Snowman



```
public class K_Snowman extends JApplet
{
    private static final long   serialVersionUID   = 1L;

    public void paint(Graphics page)
    {
        final int MID = 150;
        final int TOP = 50;

        setBackground(Color.cyan);

        page.setColor(Color.blue);
        page.fillRect(0, 175, 300, 50); // ground

        page.setColor(Color.yellow);
        page.fillOval(-40, -40, 80, 80); // sun

        page.setColor(Color.white);
        page.fillOval(MID - 20, TOP, 40, 40); // head
        page.fillOval(MID - 35, TOP + 35, 70, 50); // upper torso
        page.fillOval(MID - 50, TOP + 80, 100, 60); // lower torso

        page.setColor(Color.black);
        page.fillOval(MID - 10, TOP + 10, 5, 5); // left eye
        page.fillOval(MID + 5, TOP + 10, 5, 5); // right eye

        page.drawArc(MID - 10, TOP + 20, 20, 10, 190, 160); // smile

        page.drawLine(MID - 25, TOP + 60, MID - 50, TOP + 40); // left arm
        page.drawLine(MID + 25, TOP + 60, MID + 55, TOP + 60); // right arm

        page.drawLine(MID - 20, TOP + 5, MID + 20, TOP + 5); // brim of hat
        page.fillRect(MID - 15, TOP - 20, 30, 25); // top of hat
    }
}
```


Applet Viewer: chapter_02.K_S...
Applet started.