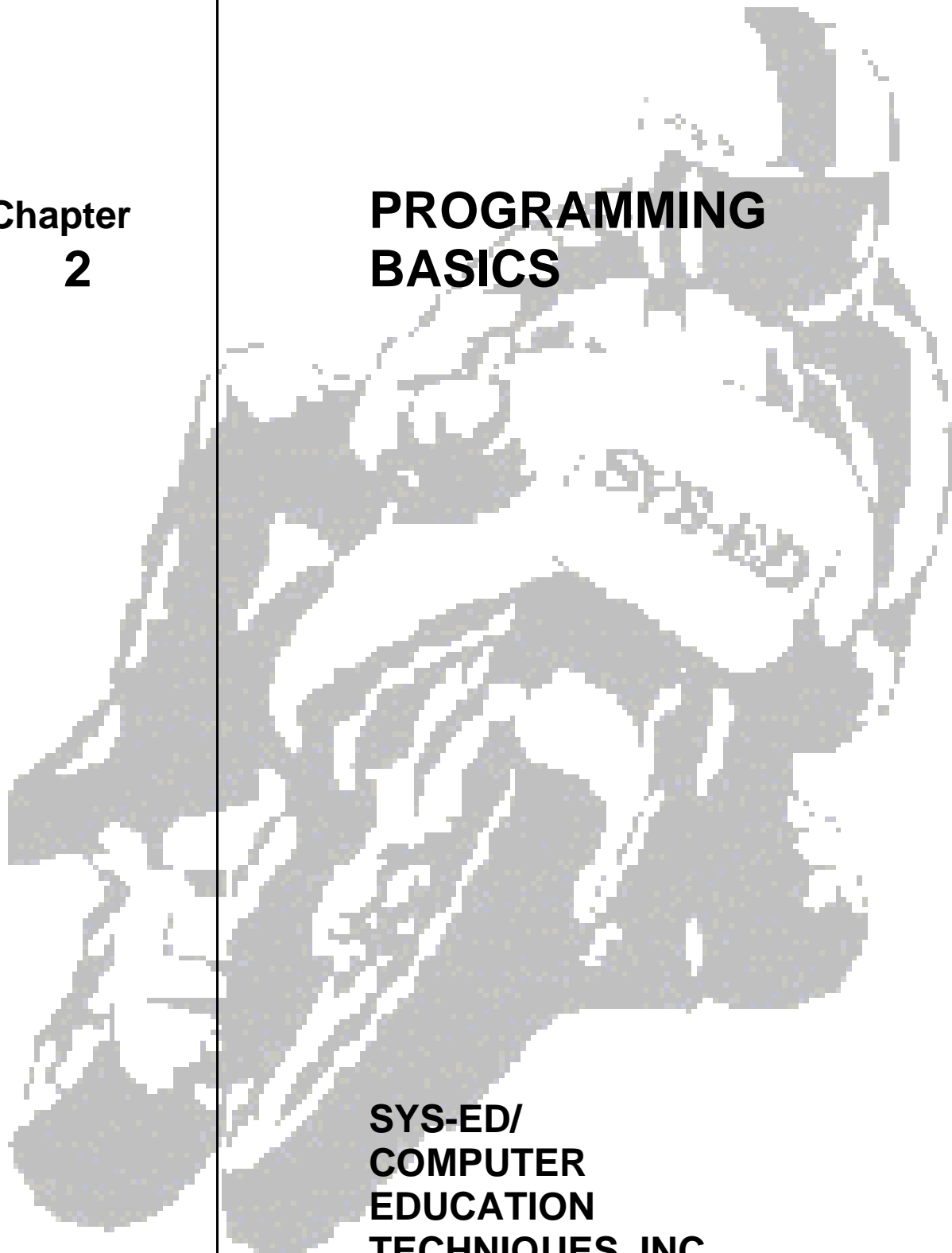


**Chapter  
2**

**PROGRAMMING  
BASICS**

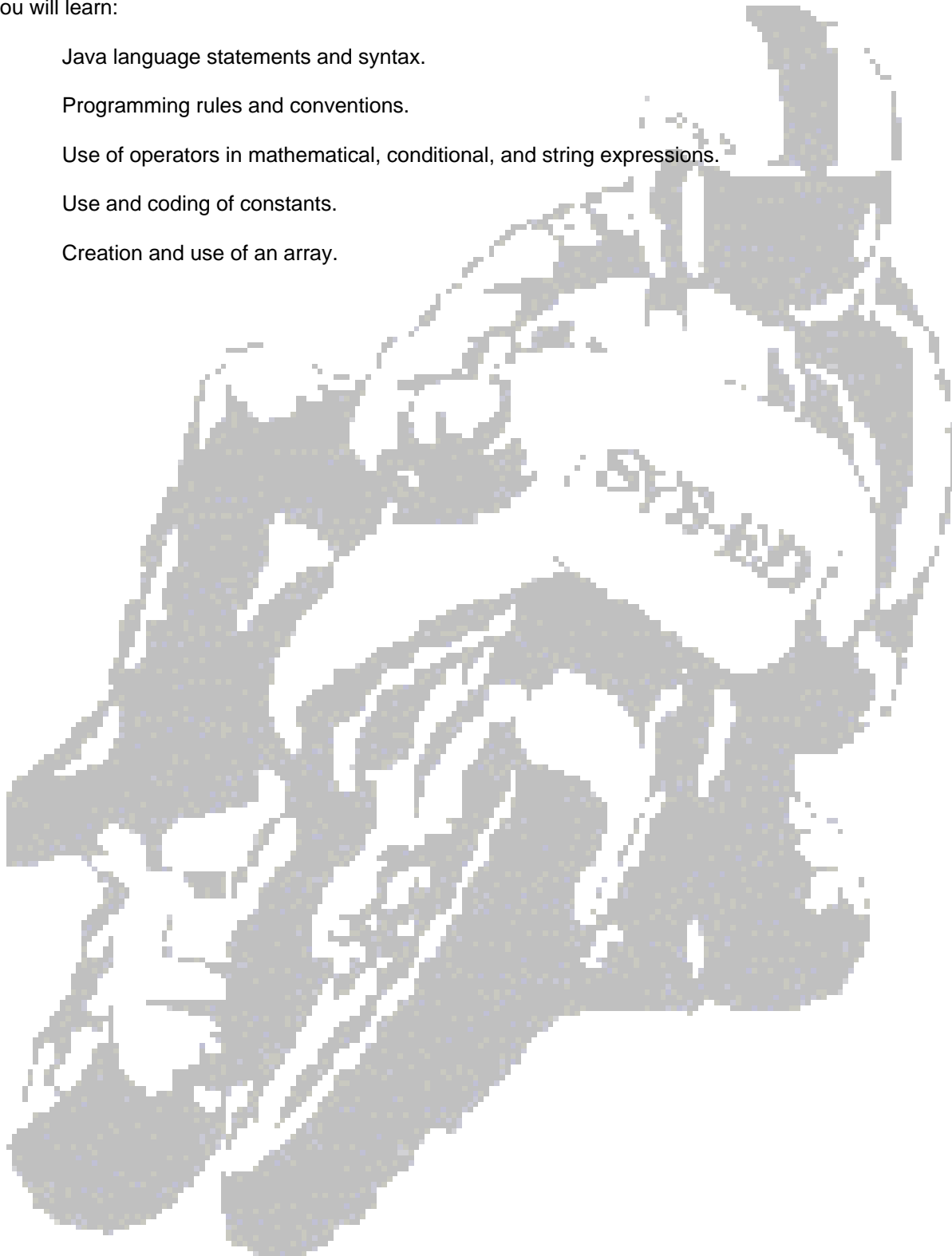


**SYS-ED/  
COMPUTER  
EDUCATION  
TECHNIQUES, INC.**

**Objectives**

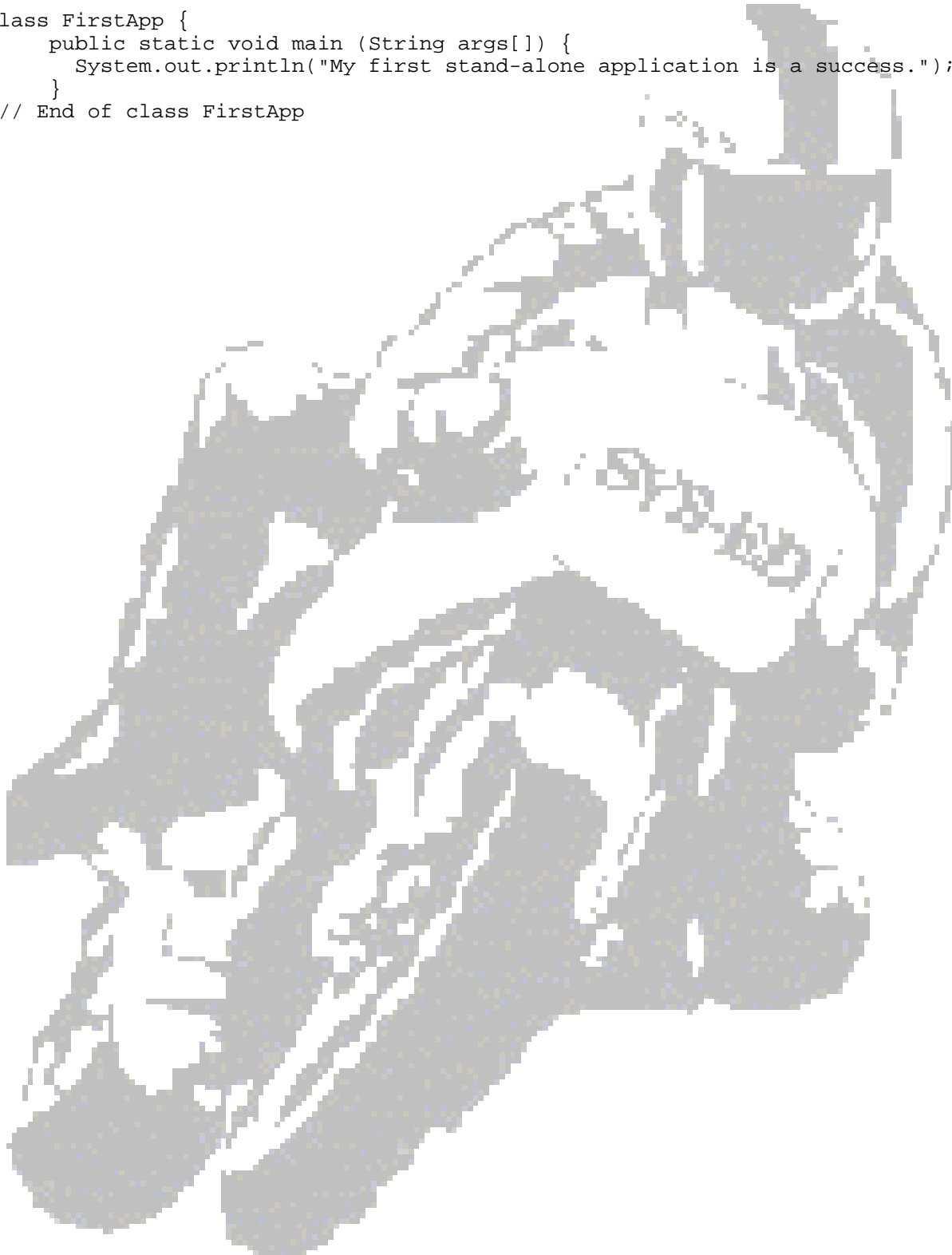
You will learn:

- C Java language statements and syntax.
- C Programming rules and conventions.
- C Use of operators in mathematical, conditional, and string expressions.
- C Use and coding of constants.
- C Creation and use of an array.



**1 Simple First Program**

```
1 class FirstApp {  
2     public static void main (String args[]) {  
3         System.out.println("My first stand-alone application is a success.");  
4     }  
5 }// End of class FirstApp
```



---

## 2 Variables

Variables act as holding areas for data of various types. The holding areas can be changed, initialized, and tested.

These variables are referred to by a variable name.

---

### 2.1. Character Variables

Character variables hold a single symbol or character or digit.

This statement below defines and initializes a character variable called ans.

```
char ans='Y';
```

Character values are enclosed within single quotes.

---

### 2.2. String Variables

String variables hold a sequence of symbols, characters, or digits.

This code defines several string variables.

```
String ans="YES";  
String tobe="To be or not to be, that is the question.";  
String addr="19 West 34th Street, New York, NY 10001";
```

String values are enclosed within double quotes.

---

### 2.3. Integer Variables

Integer variables hold a integer value, whole number or counting numbers such as:

```
...-2,-1,0,1,2,3,4,5,6,7...
```

This code defines and initializes several integer variables:

```
int one = 1;  
int grade = 90;
```

Integer values are not enclosed within quotes.

---

### 3 Declaring Variables

Several variables of the same type can be defined and initialized by one Java statement.

**Example:**

```
int grade1=75, grade2=85, grade3=100;
```

Variables do not need to be initialized when they are defined.

These Java statements define variables; but do not initialize them.

**Example:**

```
int grade1;  
string msg;  
char ans;
```

These variables can be initialized within a Java applet using assignment statements.

**Example:**

```
weight = 85;  
msg = "Hello, world.";  
Response = 'Y';
```

---

## 4 Variable Names and Data Types

A variable name typically describes the context in which the variable will be used.

A variable name:

- C can contain letters, numbers or underscore.
- C cannot start with a number.
- C must be defined to hold a certain type of data.
- C must not be a Java reserved word.

### Examples:

```
byte anum=23;           //byte holds integer value -128 to 127
short bnum=1000;        //short holds integer value -32768 to 32767
int cnum=64000;         //int holds integer -2147483648 to 2147483647
long dnum=10000000000;  //long integer can hold large 64-bit values
float enum=45.56;      //holds 32-bit floating decimal number
double fnum=123.234;   //holds 64-bit floating decimal number
char ans='y';          //holds single character, digit, or symbol
boolean doorshut=false; //holds either a true or false value
```

---

### 4.1 Variables and Class

In Java variables can also be declared to hold an instance of a particular class.

#### Example:

```
String sCity="New York";
Font f=new Font("Arial",Font.BOLD,24);
Color c=new Color(red1,green1,blue1);
```

---

## 4.2. Variable Scope Demo

```
1 class scopeDemo1
2 {
3     static int a;
4
5     public static void main(String vars[])
6     {
7         a = 1;
8         System.out.println("a=" + a );    // This line will print "a=1"
9         printA();
10    }
11    static void printA()
12    {
13        int a = 2;
14        System.out.println("a=" + a );    // This line will print "a=2"
15    }
16 }
```

---

## 4.3. Another Variable Scope Demo

```
1 class scopeDemo2
2 {
3     static int a = 1;
4
5     public static void main(String vars[])
6     {
7
8         System.out.println( "a=" + a );    // prints "a=1"
9         {
10            int a=2;
11            System.out.println( "a=" + a );    // prints "a=2"
12        }
13
14        {
15            System.out.println( "a=" + a );    // prints "a=1"
16        }
17    }
```

---

#### 4.4 Global Definition of Variables

```
1 class TestScope
2 {
3     static int a, b;          /* global definition of variables */
4
5     public static void main(String args[])
6     {
7         a=1; b=2;           /* assign a value to the global variables */
8
9         aIsAGlobal();       /* call the method aIsAGlobal()    */
10        /* to print the values of a and b */
11
12        aIsALocal();        /* call the method aIsALocal()    */
13        /* to print the values of a and b */
14
15    }
16
17    static void aIsAGlobal()
18    {
19        System.out.println("b=" + b); /* This would print 2 */
20        System.out.println("a=" + a); /* This would print 1 */
21    }
22
23    static void aIsALocal()
24    {
25        int a=4;           /* local definition of a overrides global definition */
26
27        System.out.println("b=" + b); /* This would print 2 */
28        System.out.println("a=" + a); /* This would print 4 */
29        /* as the local a overrides the global a */
30    }
31 }
```

---

## 5 Operators

Java utilizes the following operator types:

C	Arithmetic	C	Assignment
C	Logical	C	Increment and Decrement
C	Relational	C	Bitwise Operators

---

### 5.1 Arithmetic Operators

Arithmetic operators provide the capability for performing arithmetic.

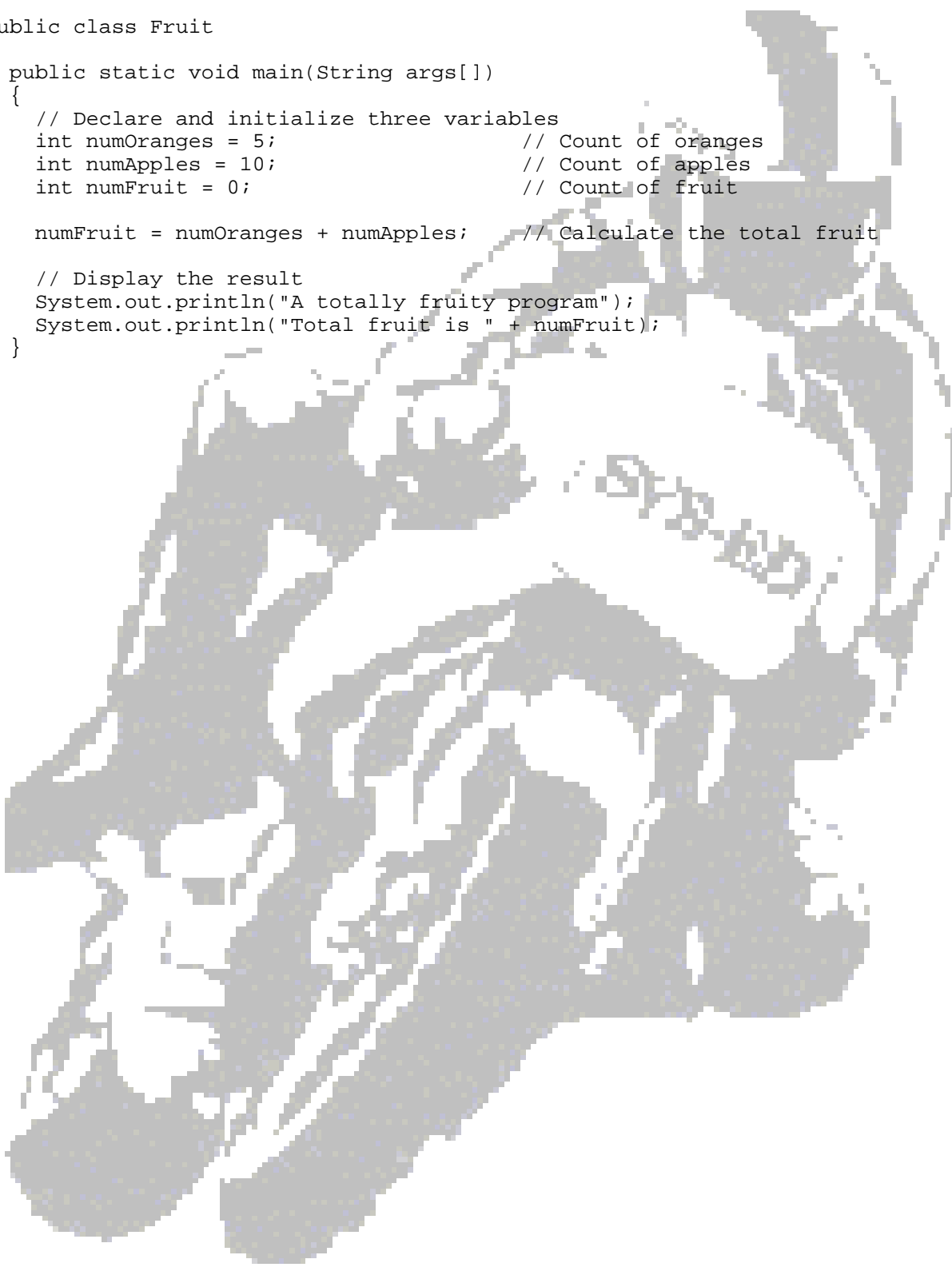
Java's standard arithmetic operators are:

Operator	Meaning
+	addition
-	subtraction
*	multiplication
/	division
%	modulus(remainder) arithmetic

---

## 5.2 Performing Calculations

```
1 public class Fruit
2 {
3     public static void main(String args[])
4     {
5         // Declare and initialize three variables
6         int numOranges = 5;           // Count of oranges
7         int numApples = 10;          // Count of apples
8         int numFruit = 0;            // Count of fruit
9
10        numFruit = numOranges + numApples; // Calculate the total fruit
11
12        // Display the result
13        System.out.println("A totally fruity program");
14        System.out.println("Total fruit is " + numFruit);
15    }
16 }
```



---

## 6 Logical Operators

Logical operators are used for determining whether:

- C an expression is either true.  
or
- C whether both expressions are true when testing multiple expressions or conditions.

Java's logical operators are:

Operator	Meaning
&&	expression one AND expression two are true.
	expression one OR expression two is true.
!	the expression is not true.

---

## 7 Relational Operators

Relational operators are used for determining how two characters or numbers relate to each other in terms of magnitude.

Java's relational operators are:

Operator	Meaning
==	equal to.
!=	not equal to.
>	greater than.
>=	greater than or equal to.
<	less than.
<=	less than or equal to.

## 8 Assignment Operators

The assignment operator(=) can be used for setting or initializing a variable to a value.

### Example:

This statement sets the iScore numerical variable to a value of 87.

```
iScore = 87;
```

Assignment operators allow the results of an evaluated expression to be assigned to a variable.

Assignment operators allow a variable on the left of the = to be set to the outcome of an evaluated expression on the right of the =.

### Examples:

This statement sets/assigns the current value of the iScore numerical variable plus five and puts the results back into the same iScore variable.

```
iScore = iScore + 5;
```

This statement sets/assigns all three numerical variables A, B, and C to zero.

```
A = B = C = 0;
```

The shortcut assignment operators +=, -=, \*=, and /= are used as follows:

Expression	Meaning
$x += y;$	$x = x + y;$
$x -= y;$	$x = x - y;$
$x *= y;$	$x = x * y;$
$x /= y;$	$x = x / y;$

---

## 8.1 Increment and Decrement Operators

The increment operator ++ provides for the incrementing/increasing of a variable by one while the decrement operator -- provides for the decrementing/decreasing of the value stored in a variable by one.

This expression decrements the numerical variable A by one: `A--;`

This expression increments the numerical variable B by one: `B++;`

---

Postfix decrement is when the decrement operator follows the variable.

This assignment statement assigns the value of the variable x to the variable y, then the contents of the variable x is decreased by one.

```
y = x--;
```

---

Prefix decrement is when the decrement operator precedes the variable.

This assignment statement decreases the value of the variable x by one, then assigns the augmented value in variable x to the variable y.

```
y = --x;
```

---

Postfix increment is when the increment operator follows the variable.

This assignment statement assigns the value of the variable x to the variable y, then the contents of the variable x is increased by one.

```
y = x++;
```

---

Prefix increment is when the increment operator precedes the variable.

This assignment statement increases the value of the variable x by one, then assigns the augmented value in variable x to the variable y.

```
y = ++x;
```

---

## 9 Using Operators

```
1 public class NumberCheck
2 {
3     public static void main(String[] args)
4     {
5         int number = 0;
6         // Get a random integer between 1 & 100
7         number = 1+(int)(100*Math.random());
8
9         if(number%2 == 0)           // Test if it is even
10            System.out.println("You have got an even number, " + number);
11        else
12            System.out.println("You have got an odd number, " + number);
13    }
14 }
```

---

**10 Using a Conditional Operator**

```
1 public class ConditionalOp
2 {
3     public static void main(String args[])
4     {
5         int nHats = 1;        // Number of hats
6         System.out.println("I have " + nHats + " hat" +
7                             (nHats == 1 ? "." : "s."));
8
9         nHats++;             // Increment number of hats
10        System.out.println("I have " + nHats + " hat" +
11                            (nHats == 1 ? "." : "s."));
12    }
13 }
```

---

## 11 Bitwise Operators

Bitwise operators perform operations on the individual bits of an integer.

The bitwise operators and their function are:

Operator	Function
&	Bitwise AND.
	Bitwise OR.
^	Bitwise XOR.
>	Shift Bits Right.
>>>	Zero fill right shift.
~	Bitwise complement.
&=	Bitwise AND assignment.
=	Bitwise OR assignment.
^=	Bitwise XOR assignment.
>=	Shift Bits Right assignment.
>>>=	Zero fill right shift assignment.

---

## 12 Literals

Literals are values or constants of different types that are assigned to variables of their respective variable types.

The assignment statements assign:

- C a numeric literal to a numeric variable.
- C a character literal to a character variable.
- C a string literal to a string variable.
- C a boolean literal to a boolean variable.

### Examples:

```
iEmployeeID = 1500;  
cSex = 'M';  
sName = "John Smith";  
bMarried = true;
```

Boolean variables take on either a true or false value.

---

### 12.1 Escape Codes

These special character escape codes can be used as literals:

Escape Code	Meaning
\n	Newline
\t	Tab
\b	Backspace
\r	Carriage return
\f	Formfeed
\\	Backslash
\'	Single quote
\"	Double quote
\xdd	Hexadecimal

---

## 12.2 String Literals

A string literal is a sequence of symbols or text embraced by double quotes.

### Examples:

```
"This is a test"
```

```
"The cat said, \"Meow!\""
```

```
"January\tFebruary\tMarch"
```

---

## 12.3 Using Escape Sequence Codes

```
1 public class EscapeSequence
2 {
3     public static void main(String[] args)
4     {
5         System.out.println("\"It\'s freezing in here\", he said coldly.");
6     }
7 }
```

---

## 13 Statements and Expressions

### Statement

A statement performs a single java task.

Java statements are always terminated by a ;.

### Examples:

```
int x =0;
setBackground(Color.red);
g.drawString("Hi Ma",10,50);
```

### Expression

An expression is a statement that returns a value.

This arithmetic expression returns the value of 5; assigning this value to the variable y.

### Example:

```
y = 2 + 3;
```

---

## 14 Arrays

Arrays hold a list of related items of the same data type.

### Example:

This Java statement defines and initializes a grades array containing five grades.

```
int[] grades = {90,80,85,75,83};
```

- C Each item of an array can be referenced and used by an integer index.
- C The items in an array are referenced by an integer beginning at 0.

The first item is indexed or referred to by the integer 0 and the second item is indexed or referred to by the integer 1 and so on.

### Examples:

To assign 89 as the first grade item in the grades array, this Java assignment statement could be used:

```
grades[0] = 89;
```

To assign 92 as the third grade item in the grades array, the following Java assignment statement can be used:

```
grades[2] = 92;
```

It is also possible to define an array of multiple character values without assigning the initial array values.

### Example:

The items in this character array would be referenced/indexed by the integers 0 through 9.

```
char[] letters = new char[10];
```

---

## 14.1 Prints Out the Command Line Arguments

```
1 class EchoCmdArgs
2 {
3     public static void main (String args[])
4     {
5         int i=0;           // i will iterate through the arguments
6
7         while (i<args.length)
8         {
9             System.out.println(args[i]); // Print out the argument
10            i = i + 1;           // Go to the next
11        }
12    }
13
14 }
```