Character

- char var1=‘G’;
- char var2=‘O’;
- char var3=‘O’;
- char var4=‘D’;
- Print them using single print statement
- Concatenation?
- Storing inside a string variable?

One possible Solution

```
str = String.valueOf(a)+String.valueOf(b)+String.valueOf(c);
```

Try it with your name

Boolean

boolean b1=true/false;

Take an integer number as input from user

If the number is multiple of 2, set the flag variable to True

Now If the flag is True print Multiple of 2 otherwise print not a multiple of 2

Memory

- All the primitive types get memory from stack

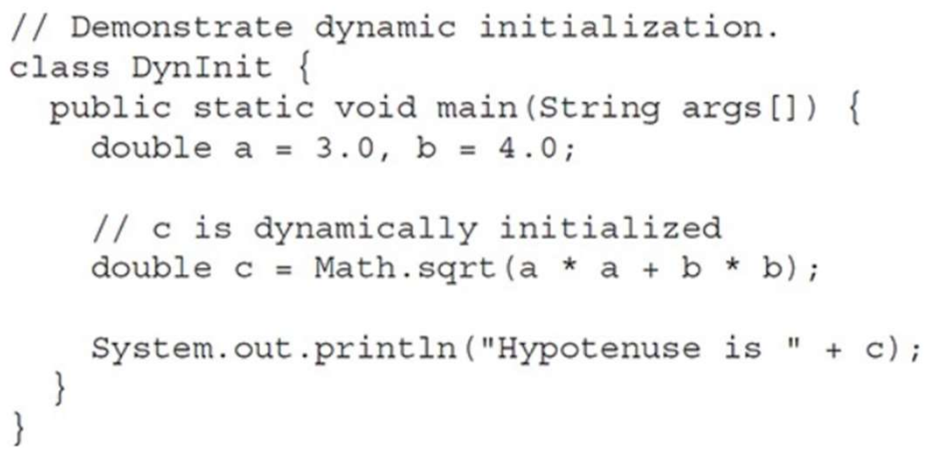
Declaring Variables

- Declaring a variable is for:
  - Setting the identifier
  - Type of value
  - Initial value that it takes
  - Exp: int a;   , char c;   , boolean b;

Declare a variable and print it without assigning any values

Dynamic Initialization

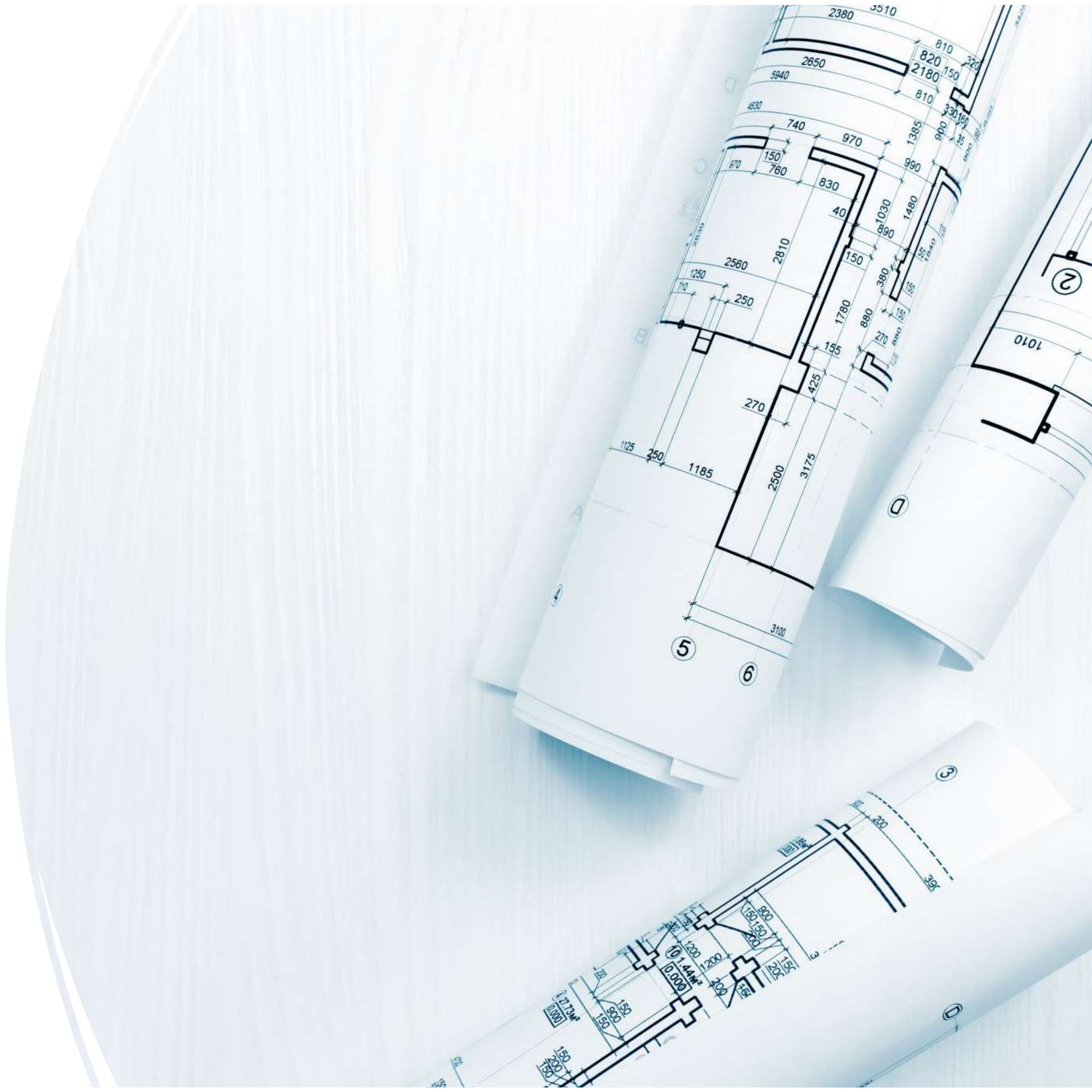
- Use of Math class
- Values are not always assigned as a constant, there could be a method call etc



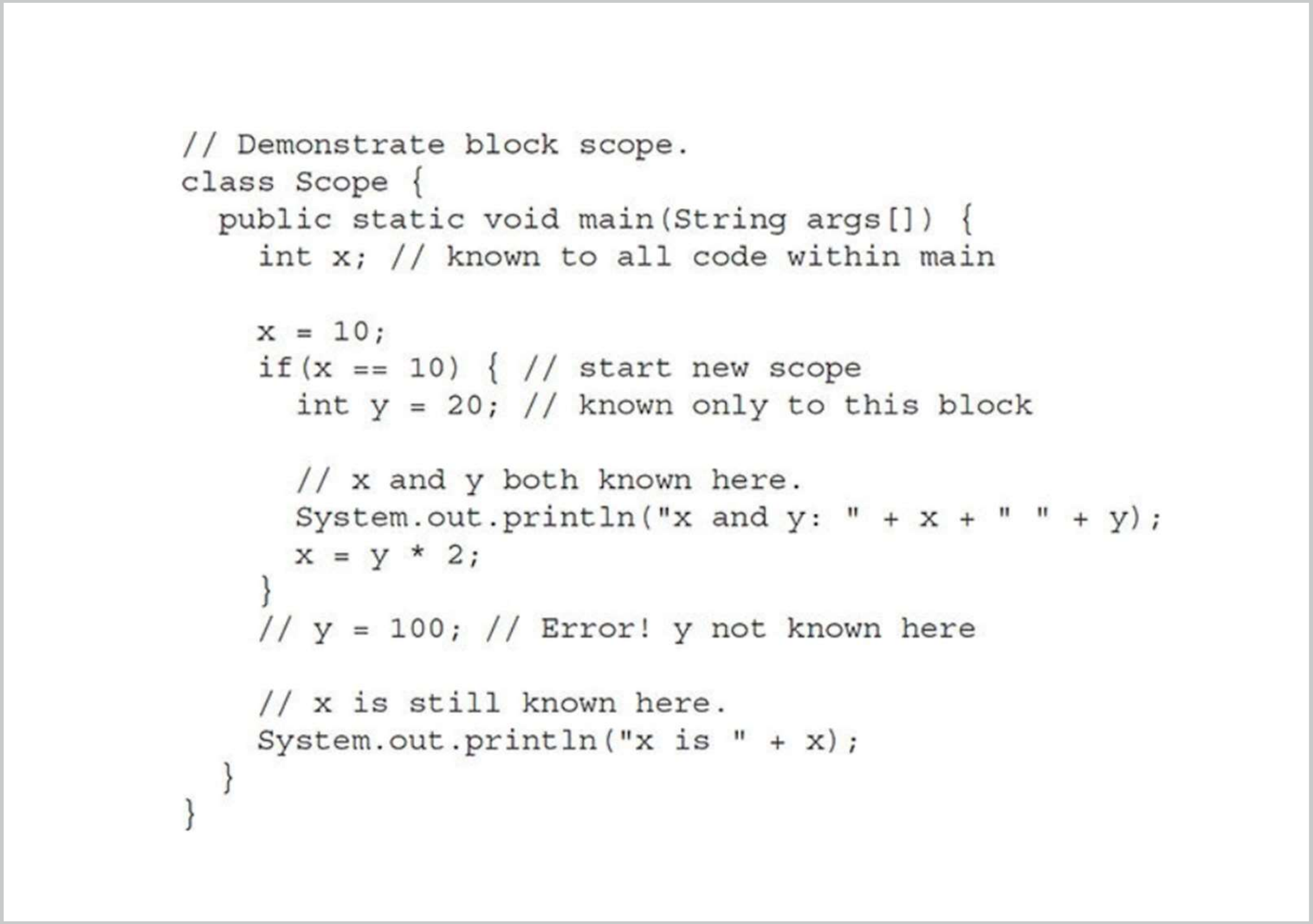
```java
// Demonstrate dynamic initialization.
class DynInit {
  public static void main(String args[]) {
    double a = 3.0, b = 4.0;

    // c is dynamically initialized
    double c = Math.sqrt(a * a + b * b);

    System.out.println("Hypotenuse is " + c);
  }
}
```

Scope and lifetime of a variable

- Scope defines the visibility of your variable along with its lifetime
- A block defines a new scope
- Method’s scope is within curly braces:
    - Defining a variable inside method limits its scope to outside world
    - Concept of Local Variable

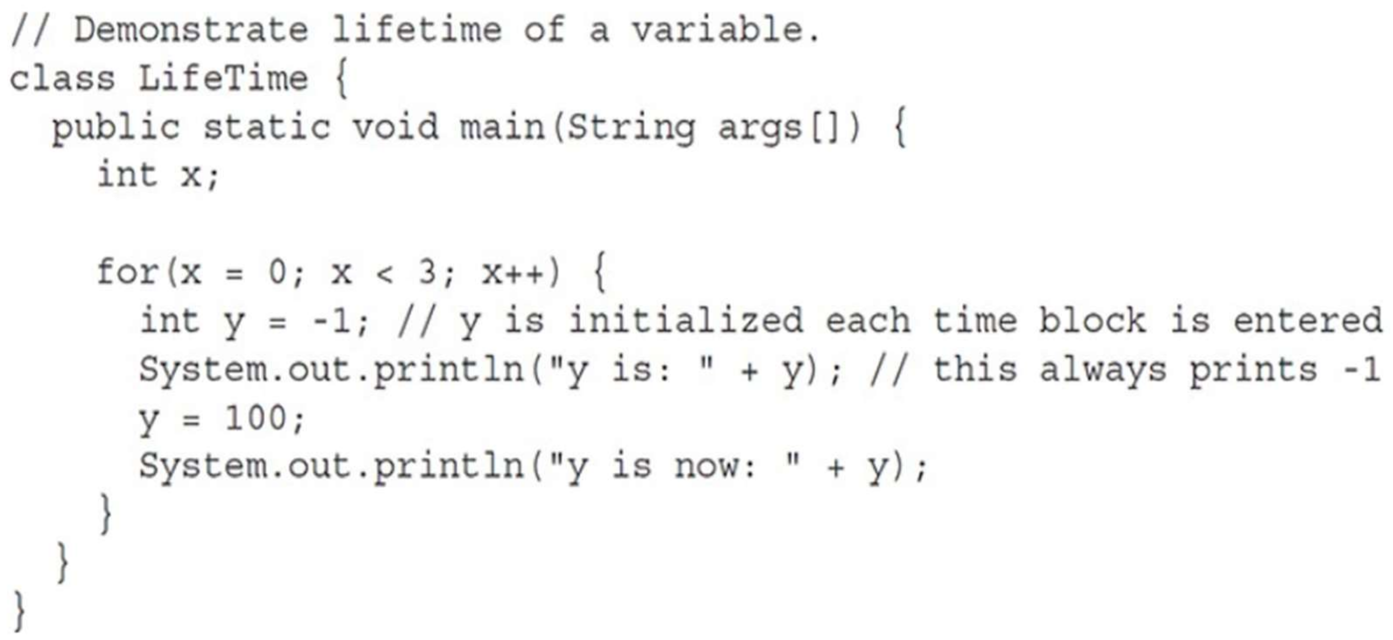


Scope



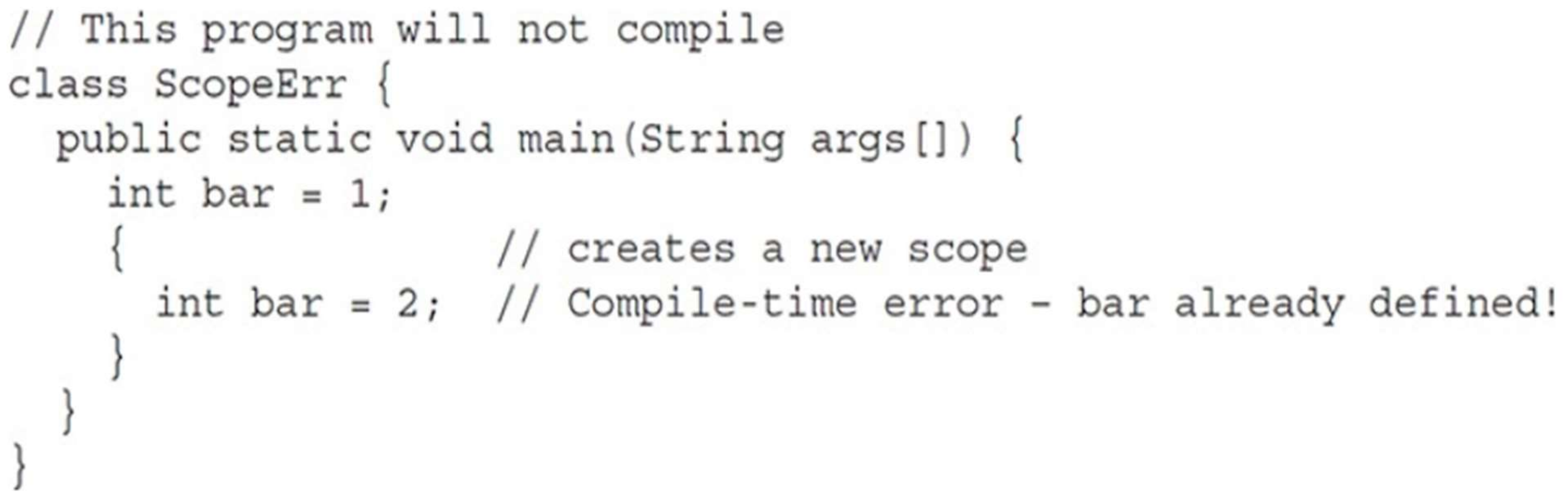
```
// Demonstrate block scope.
class Scope {
  public static void main(String args[]) {
    int x; // known to all code within main

    x = 10;
    if(x == 10) { // start new scope
      int y = 20; // known only to this block

      // x and y both known here.
      System.out.println("x and y: " + x + " " + y);
      x = y * 2;
    }
    // y = 100; // Error! y not known here

    // x is still known here.
    System.out.println("x is " + x);
  }
}
```

Lifetime



```
// Demonstrate lifetime of a variable.
class LifeTime {
  public static void main(String args[]) {
    int x;

    for(x = 0; x < 3; x++) {
      int y = -1; // y is initialized each time block is entered
      System.out.println("y is: " + y); // this always prints -1
      y = 100;
      System.out.println("y is now: " + y);
    }
  }
}
```

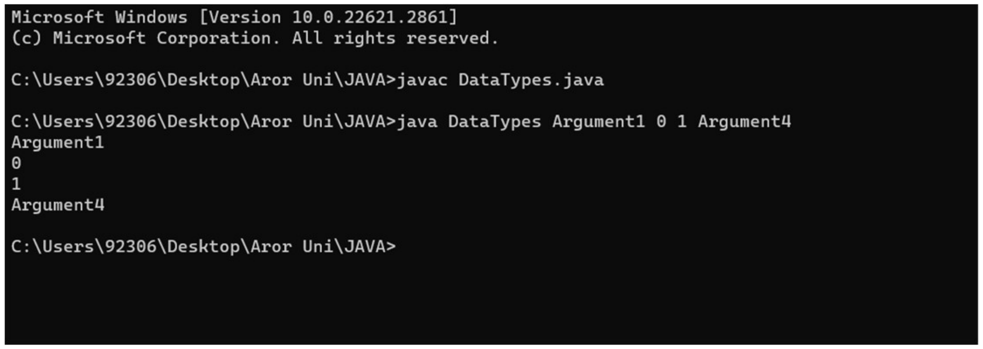
Same name issue



```
// This program will not compile
class ScopeErr {
  public static void main(String args[]) {
    int bar = 1;
    {                 // creates a new scope
      int bar = 2;  // Compile-time error - bar already defined!
    }
  }
}
```

Command Line Arguments

```
import java.util.Scanner;
class DataTypes{

public static void main(String var[]){

for(int i=0; i<var.length; i++){
System.out.println(var[i]);
}


}


}
```

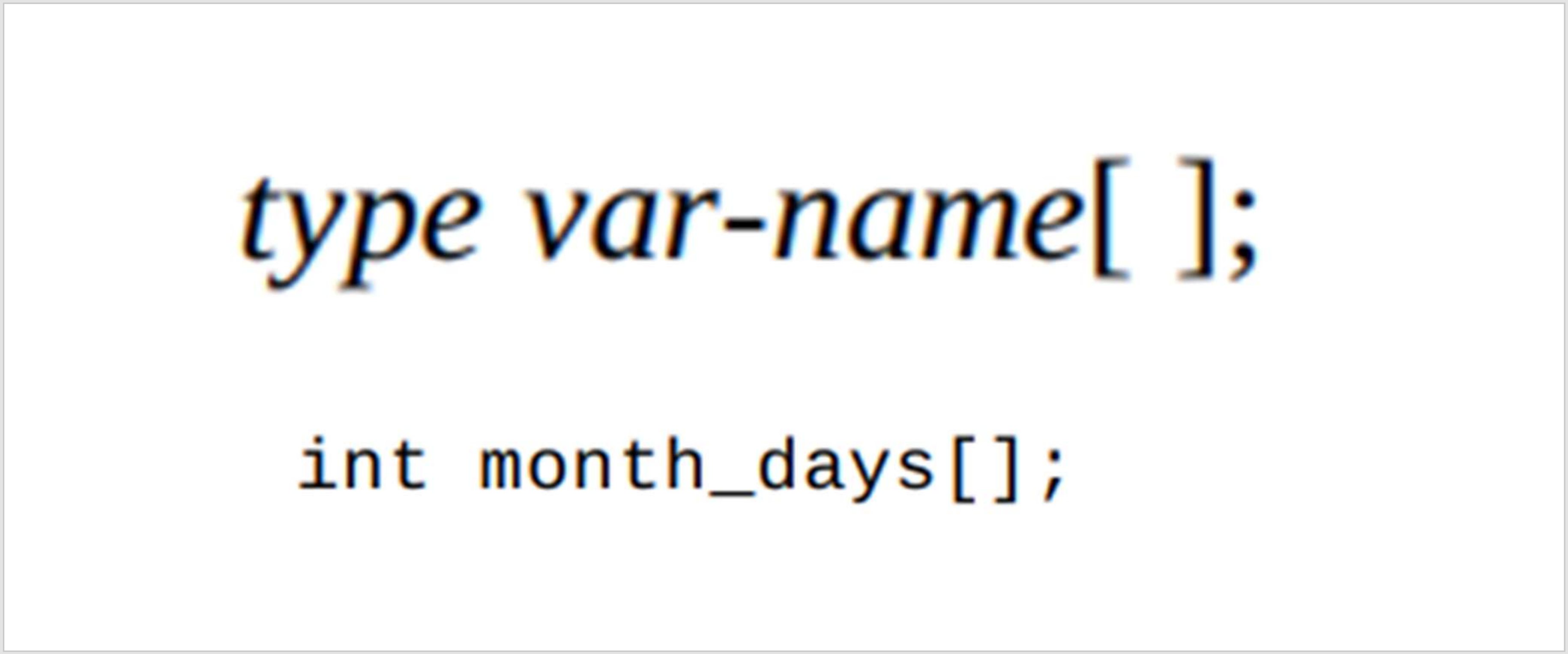


```
Microsoft Windows [Version 10.0.22621.2861]
(c) Microsoft Corporation. All rights reserved.

C:\Users\92306\Desktop\Aror Uni\JAVA>javac DataTypes.java

C:\Users\92306\Desktop\Aror Uni\JAVA>java DataTypes Argument1 0 1 Argument4
Argument1
0
1
Argument4

C:\Users\92306\Desktop\Aror Uni\JAVA>
```

Arrays

- Grouping of related(homogenous) data
- Each element is accessed:
  - Via Index (starting from zero)

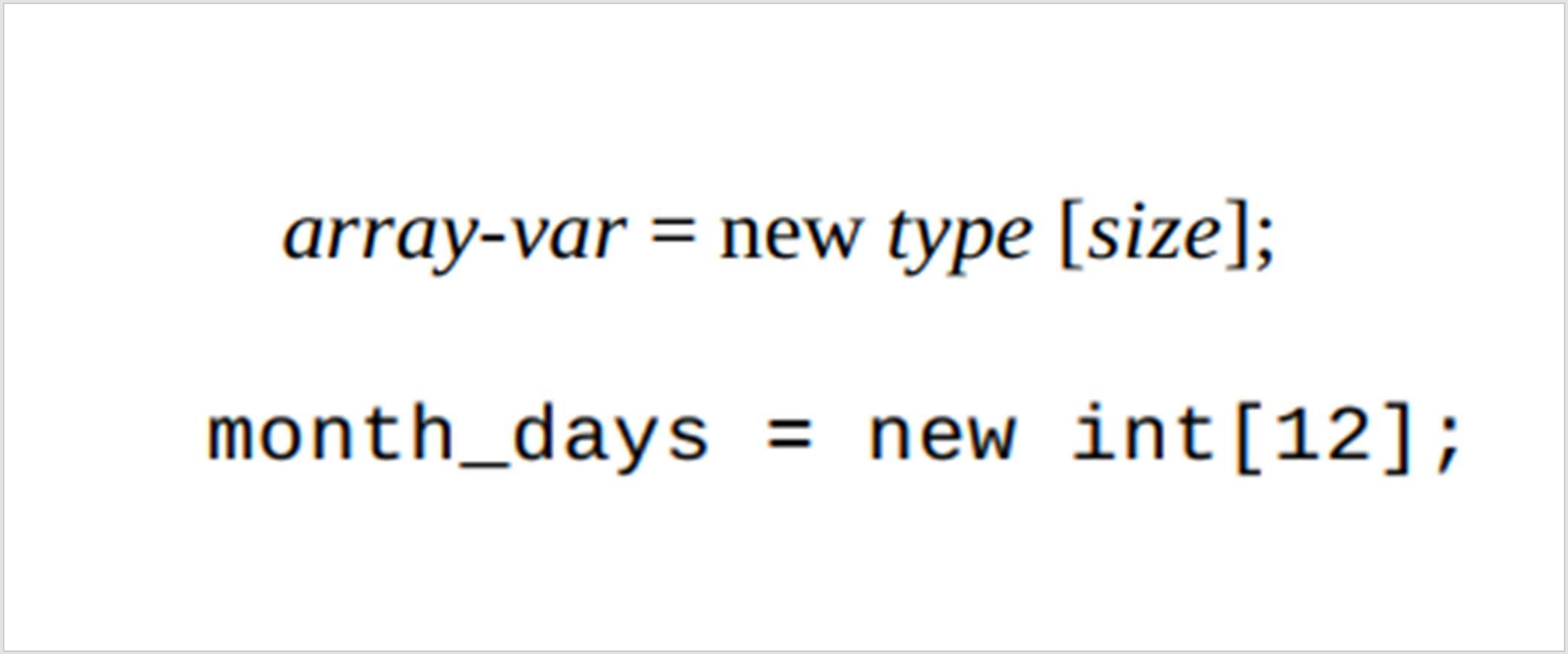
Array Declaration



type var-name[ ];

```
int month_days[];
```

Allocation of memory with new



array-var = new *type* [*size*];

```
month_days = new int[12];
```

Access without assigning values to array elements

- Numeric data types with a zero value
- Boolean with false
- Reference types with null values

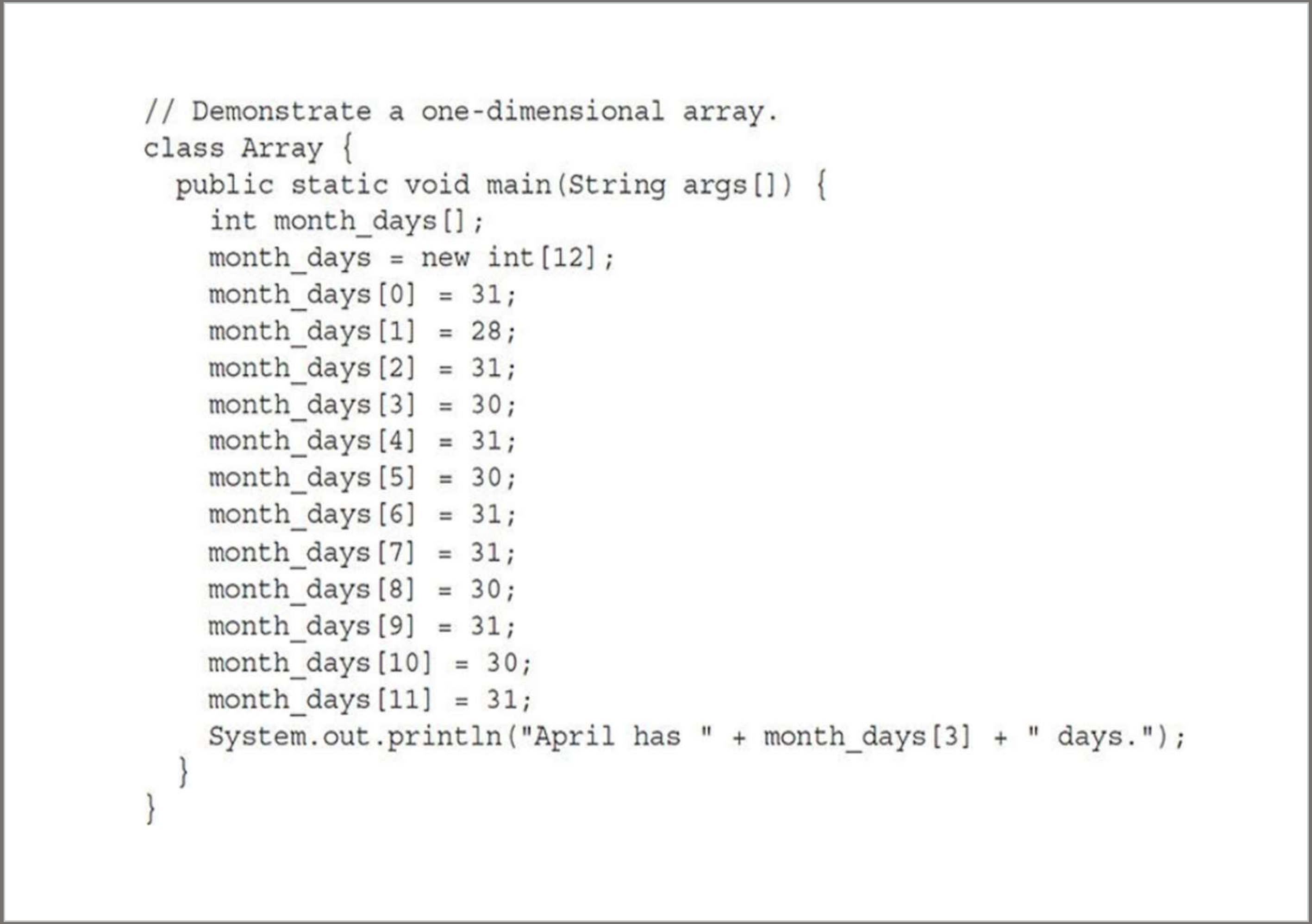
Assigning and printing values

```
month_days[1] = 28;
```

The next line displays the value stored at index 3:

```
System.out.println(month_days[3]);
```

Putting it all to gather



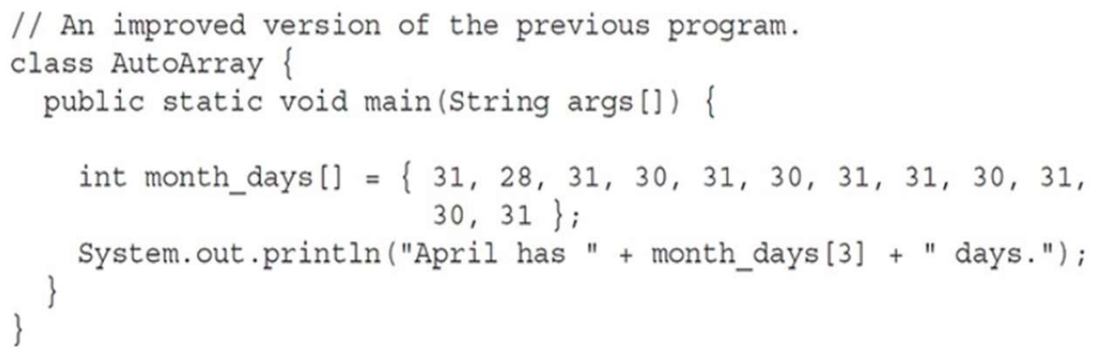
```
// Demonstrate a one-dimensional array.
class Array {
  public static void main(String args[]) {
    int month_days[];
    month_days = new int[12];
    month_days[0] = 31;
    month_days[1] = 28;
    month_days[2] = 31;
    month_days[3] = 30;
    month_days[4] = 31;
    month_days[5] = 30;
    month_days[6] = 31;
    month_days[7] = 31;
    month_days[8] = 30;
    month_days[9] = 31;
    month_days[10] = 30;
    month_days[11] = 31;
    System.out.println("April has " + month_days[3] + " days.");
  }
}
```

Combine declaration and allocation

```
int month_days[] = new int[12];
```

Array Initializer

- List of comma separated values, surrounded by curly braces
- Array size auto decided, according to number of elements



```
// An improved version of the previous program.
class AutoArray {
  public static void main(String args[]) {

    int month_days[] = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31,
                         30, 31 };
    System.out.println("April has " + month_days[3] + " days.");
  }
}
```