Java Syntax and Style

Programmer-Defined Names

- In addition to reserved words, Java uses standard names for library packages and classes:

  String, Graphics, javax.swing, JApplet, JButton, ActionListener, java.awt

- The programmer gives names to his or her classes, methods, fields, and variables.

Names

- Syntax: A name can include:
  - upper- and lowercase letters
  - digits
  - underscore characters
- Syntax: A name cannot begin with a digit.
- Style: Names should be descriptive to improve readability.

Names

- Programmers follow strict style conventions.
- Style: Names of classes begin with an uppercase letter, subsequent words are capitalized:

```
public class FallingCube
```

- Style: Names of methods, fields, and variables begin with a lowercase letter, subsequent words are capitalized.

```
private final int delay = 30;
public void dropCube()
```

Names

- Method names often sound like verbs:

  setBackground, getText, dropCube, start

- Field names often sound like nouns:

  cube, delay, button, whiteboard

- Constants sometimes use all caps:

  PI, CUBESIZE

- It is OK to use standard short names for temporary “throwaway” variables:

  i, k, x, y, str

Syntax vs. Style

- Syntax is part of the language. The compiler checks it.
- Style is a convention widely adopted by software professionals.
- The main purpose of style is to improve the **readability of programs**.

Style

- Arrange code on separate lines; insert blank lines between fragments of code.
- Use comments.
- Indent blocks within braces.

Style (cont'd)

Before:

```
public boolean
moveDown(){if
(cubeY<6*cubeX)
{cubeY+=yStep;
return true;}else
return false;}
```

Compiles fine!

After:

```
public boolean moveDown()
{
    if (cubeY < 6 * cubeX)
    {
        cubeY += yStep;
        return true;
    }
    else
    {
        return false;
    }
}
```

Style (cont'd)

```
public void fill (char ch)
{
    int rows = grid.length, cols = grid[0].length;
    int r, c;

    for (r = 0; r < rows; r++)
    {
        for (c = 0; c < cols; c++)
        {
            grid[r][c] = ch;
        }
    }
}
```

Add blank lines for readability

Add spaces around operators and after semicolons

Java Arrays

Introduction

- Array is a useful and powerful aggregate data structure presence in modern programming languages
- Arrays allow easy access and manipulation to the values/objects that they store
- Arrays are indexed by a sequence of integers

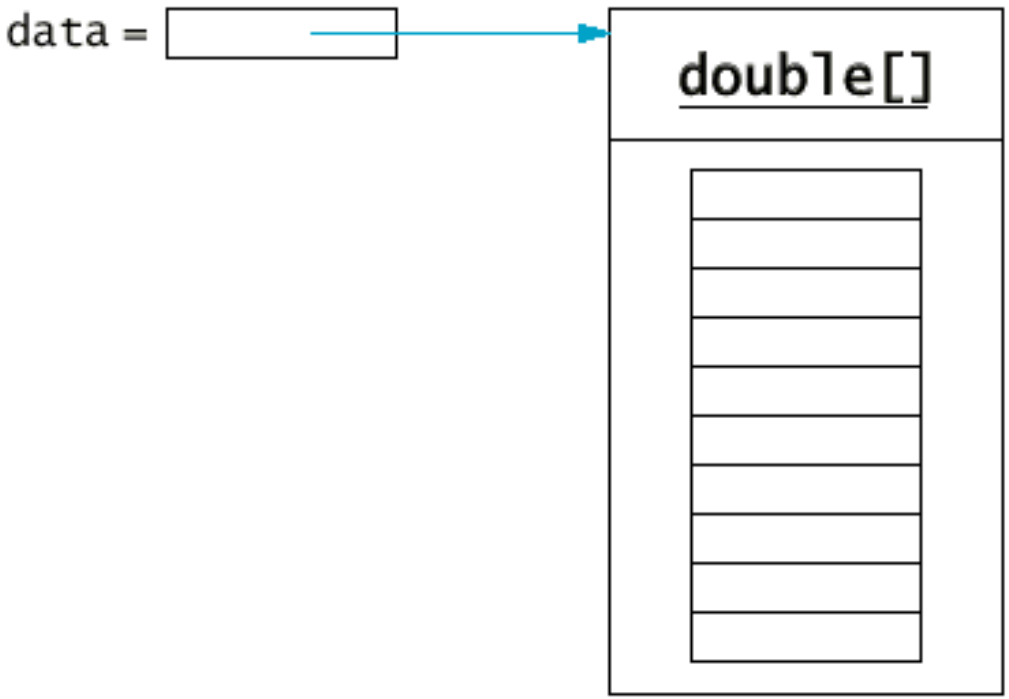
Arrays

- new is used to construct a new array:

  new double[10]

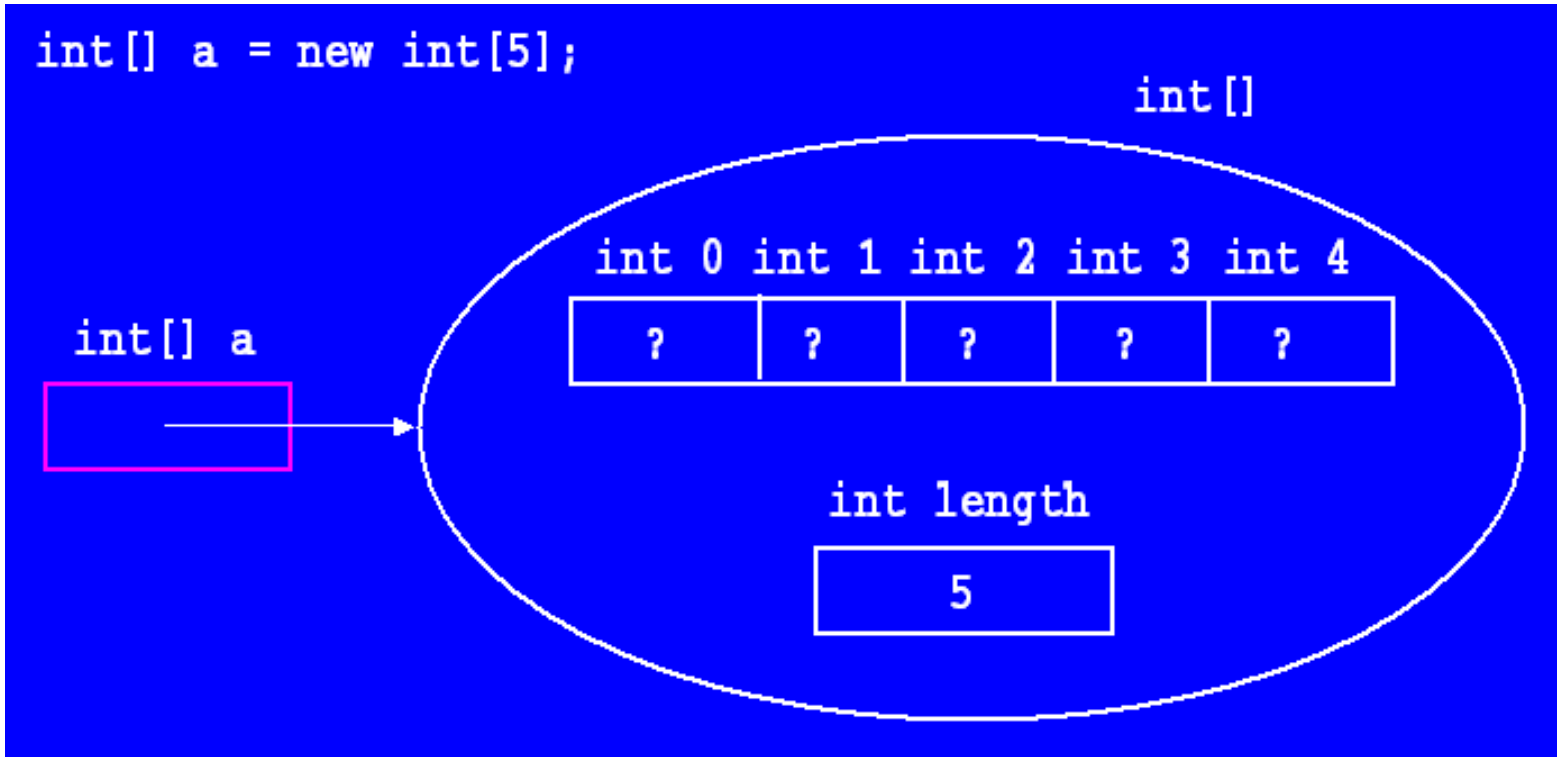
- Store 10 double type variables in an array of doubles

  double[] data = new double[10];



integer Arrays

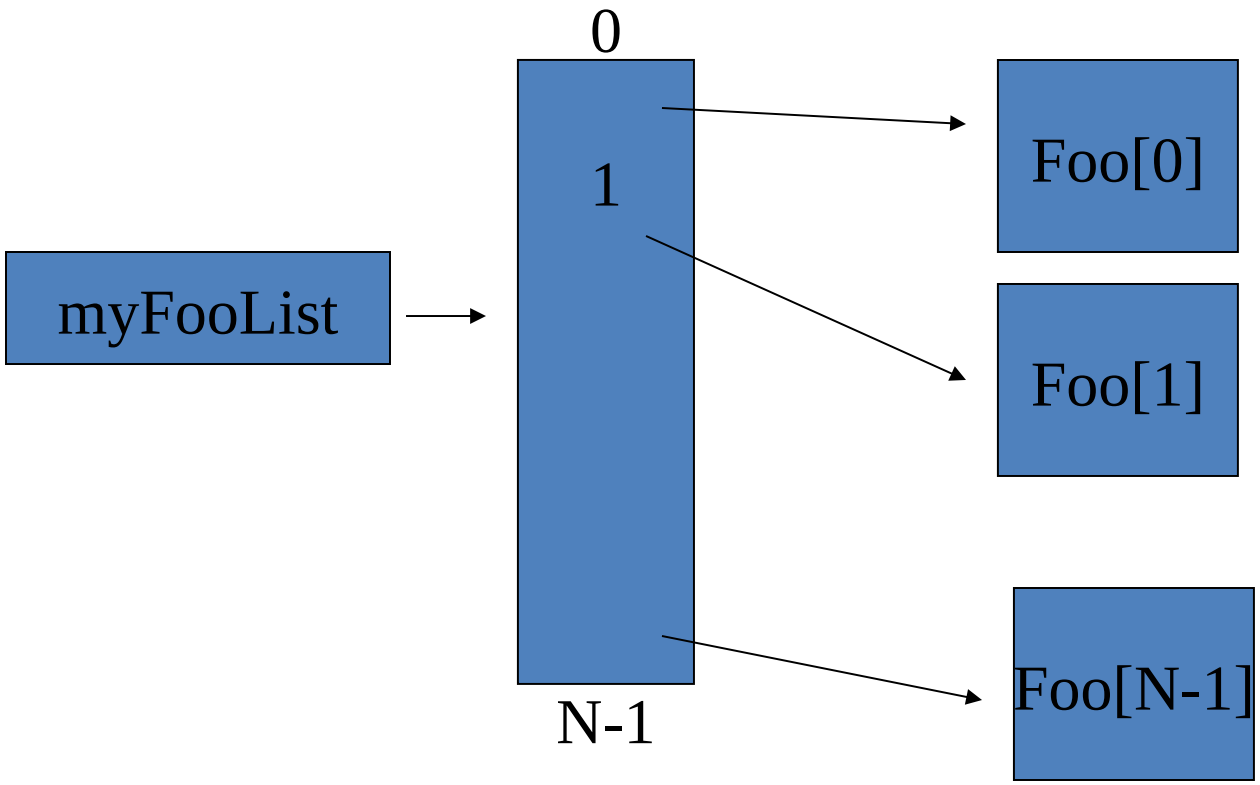
int[] a = new int[5];



Array of Object References

class Foo() { ….}

Foo[ ] myFooList = new Foo[N];



Arrays

- *fixed length*
- Element of specific type or references to `Objects`
- `[ ]` is used to access array elements
  `data[4] = 29.95;`
- Use length attribute to get array length.
  - `data.length`
  - (Not a method!)

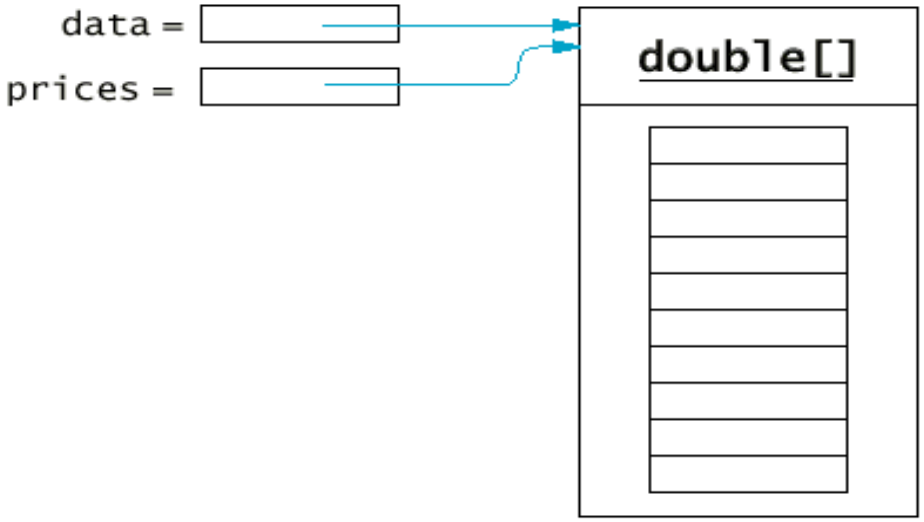
Array

- homogeneous data structure: each of its members stores the same type (either primitive or reference)
- indices go from 0 to one less than the length of the array
- each array object stores a **public final int length** instance variable that stores the length of the array
- we can access the value stored in this field, in the example above, by writing **a.length**

Copying Arrays

Copying an array reference yields a second reference to the same array

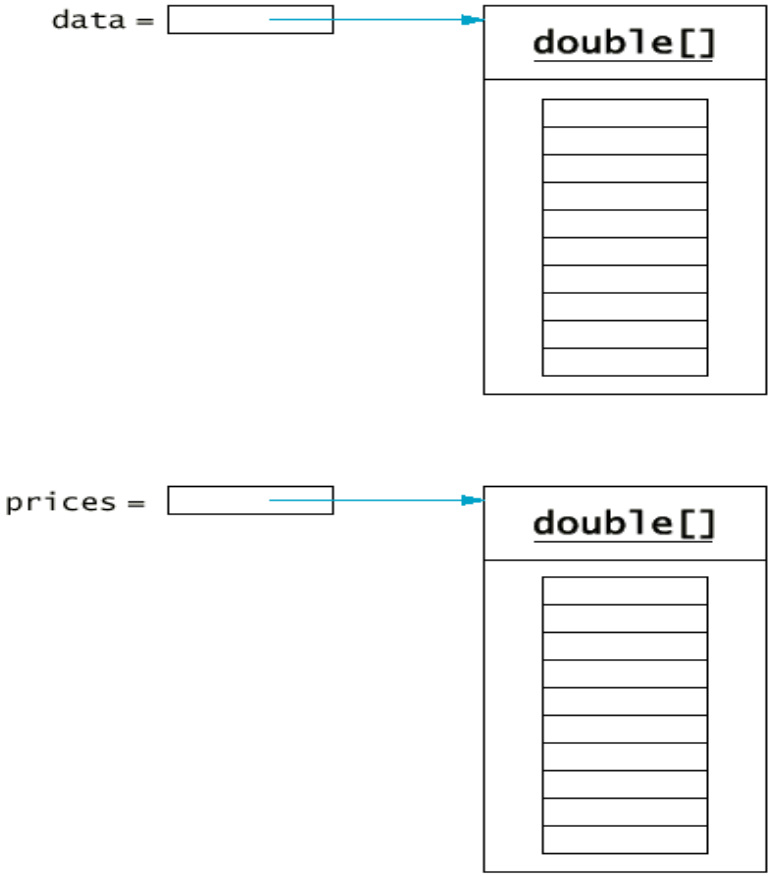
```
double[] data = new double[10];
  // fill array . . .
  double[] prices = data;
```



Cloning Arrays

- Use `clone` to make true copy

```
double[] prices = (double[])data.clone();
```



Swapping Array Elements

- Suppose you want to swap two elements in the array, say entries with indices i and j. Assuming we are dealing with an array of ints
  - int temp = A[i]; // save a copy of A[i] in temp
  - A [i] = A[j]; // copy the content of A[j] to A[i]
  - A[j] = temp; // copy the content of temp to A[j]
- Note that : A[i]= A[j] and A[j] = A[i] do not swap content

- Exercise: Reverse an array using swaps

Accessing Arrays

- int[] a = new int[]{4, 2, 0, 1, 3};
- system.out.println(a[0]);

- if (a[**5**] == 0) ...some statement
- if the value computed for the index is less than 0, or greater than OR EQUAL TO the length of the array
  - trying to access the member at an illegal index causes Java to throw the
  - ArrayIndexOutOfBoundsException which contains a message showing what index was attempted to be accessed