Programming 1

Lecture 8 – Enum, 2D Array, ArrayList, For-Each, StringBuilder...

Faculty of Information Technology
Hanoi University

Contents

(*) This lecture uses several images from the book Big Java by C. S. Horstmann

A motivating example...

- Supposed that you need to represent weekdays in a Java program.
- You could use Strings:
- What if the value of `weekday` is not the name of a week day?

  (e.g. `weekday = "June"`)

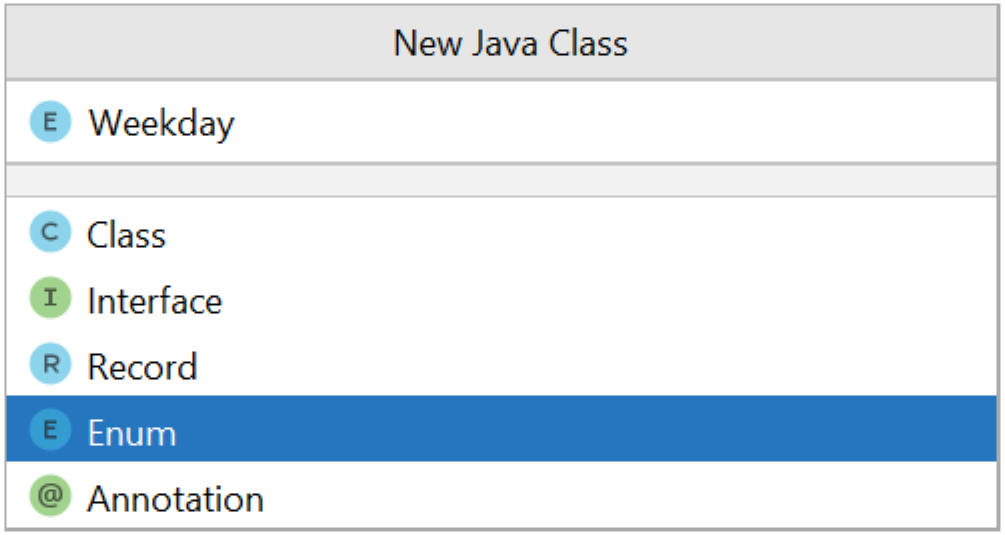
```
switch (weekday) {
    case "Monday":
        break;
    case "Tuesday":
        break;
    // ...
}
```

Creating Enum

- Enum in java is a data type that contains fixed set of constants.
- An enum type is a special kind of Java class.

```
public enum Weekday {
    MON, TUE, WED, THU, FRI, SAT, SUN
}
```

- Enum constants are separated by commas.
- Constants should be in UPPERCASE.



Using Enum

- Declaring variables with enum type:

```java
Weekday d = Weekday.MON;
Weekday d1;
d1 = Weekday.TUE;
```

- Using enum values:

```java
if (d == d1) {
    System.out.println("Equal");
} else {
    System.out.println("Not equal");
}
System.out.println(d + " & " + d1);
```

Enum in Switch

- **Level** enum definition

```
public enum Level {
    LOW, MEDIUM, HIGH
}
```

- Using the above enum:

```
Level myVar = Level.MEDIUM;
switch (myVar) {
    case LOW:
        System.out.println("Low level");
        break;
    case MEDIUM:
        System.out.println("Medium level");
        break;
    case HIGH:
        System.out.println("High level");
        break;
}
```

Looping through an Enum

- Get an array of all enum values:

```
Weekday[] wds = Weekday.values();
```

- Loop through the array:

```
Weekday[] wds = Weekday.values();
for (int i = 0; i < wds.length; i++) {
    System.out.print(wds[i] + ", ");
}
```

- Output:

```
MON, TUE, WED, THU, FRI, SAT, SUN,
```

Static Import

- The `import` statement
  - import classes
- The `import static` statement
  - import static members (attributes & methods)
- Usage
  - Use static methods and static variables without referencing their class name.
  - To use enum constants without class name.

Static Import & Enum

- Without static import:

```
Weekday wd = Weekday.MON;
```

- With `import static Weekday.*;`

```
Weekday wd = MON;
```

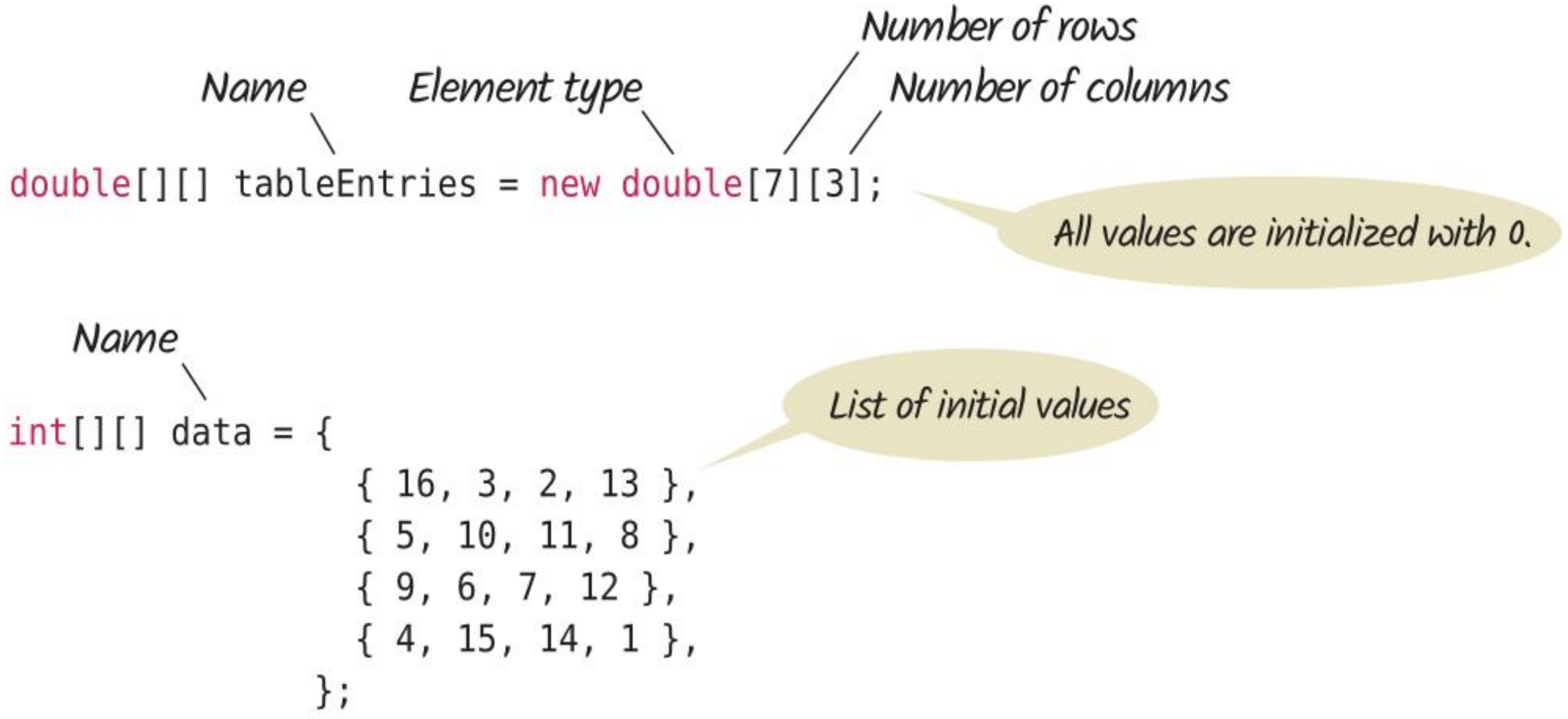
Multi-Dimensional Arrays

- Example of a two-dimensional array:

```
int[][] arr = new int[5][10];
```

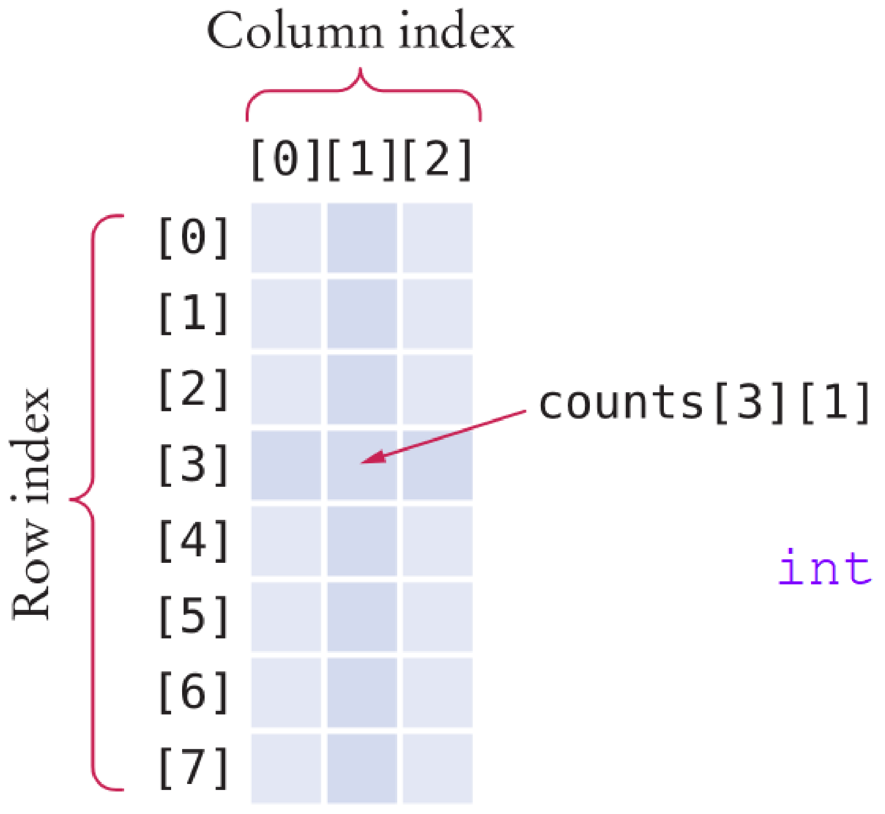
- Above is an array of 5 rows and `10` columns.
  - 5 is the length of the first dimension
  - 10 is the length of the second dimension
- An array can have many dimensions.
  - High-dimensional arrays (e.g. 4D, 5D… are difficult to visualize)

2D array declaration



```
double[][] tableEntries = new double[7][3];

int[][] data = {
                 { 16, 3, 2, 13 },
                 { 5, 10, 11, 8 },
                 { 9, 6, 7, 12 },
                 { 4, 15, 14, 1 },
               };
```

Accessing 2D array elements



```
int[][] counts = new int[8][3];
```

Loop through a 2D array

```
for (int i = 0; i < counts.length; i++) {
    for (int j = 0; j < counts[i].length; j++) {
        // do something with counts[i][j]
    }
}
```

The ArrayList class

- The `ArrayList` class defines a *dynamically sized* array.
  - The path to this class is `java.util.ArrayList`.
- Array lists can grow and shrink as needed.
- The `ArrayList` class supplies methods for common tasks, such as inserting and removing elements.

Declaring and using ArrayList

- Example: declaring an ArrayList of strings

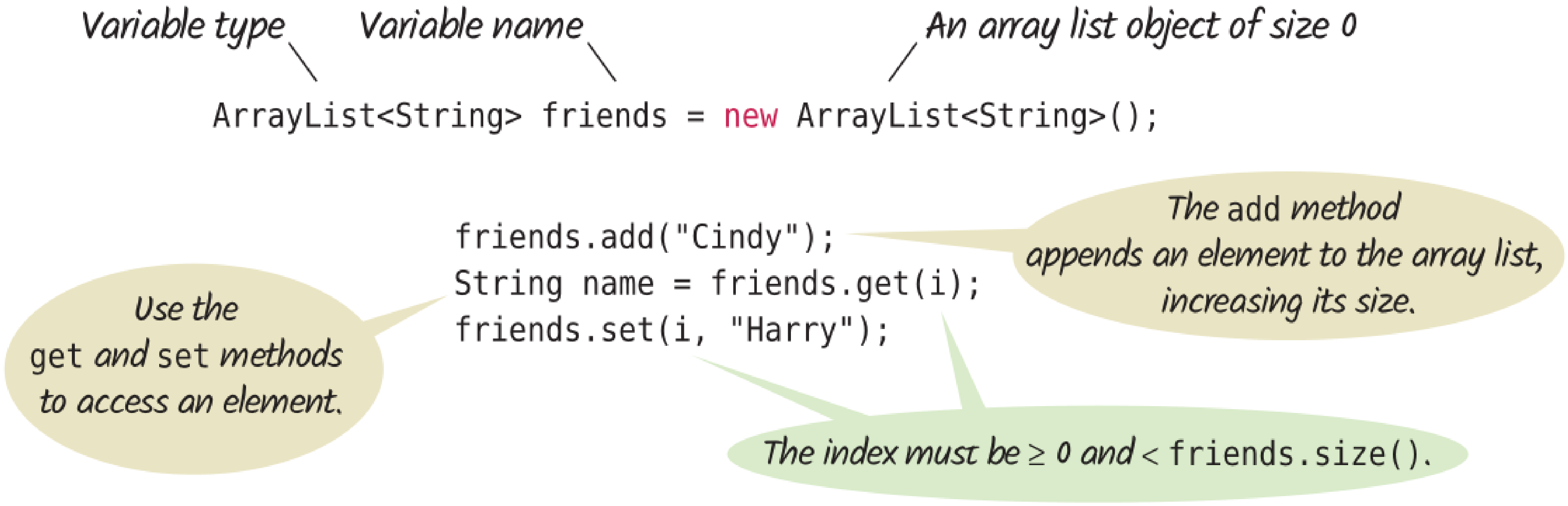
```
ArrayList<String> names = new ArrayList<String>();
```

- The type of the list's element is specified as `String`.
  - If unspecified, the elements take the `Object` type.

- Syntax:
  - To construct an array list: `new ArrayList<typeName>()`
  - To access an element:

```
arrayListVar.get(index)
arrayListVar.set(index, value)
```

ArrayList usage

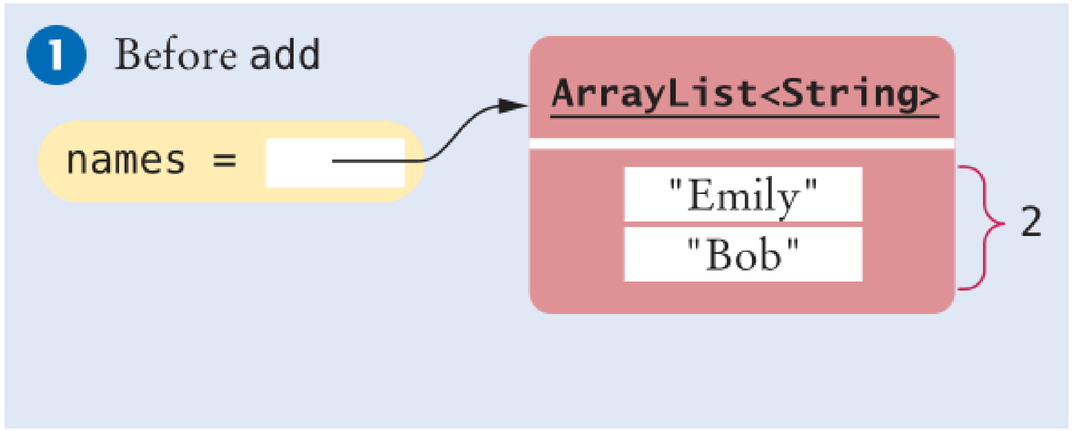

Variable type
Variable name
An array list object of size 0
The add method appends an element to the array list, increasing its size.
Use the get and set methods to access an element.
The index must be ≥ 0 and < friends.size().

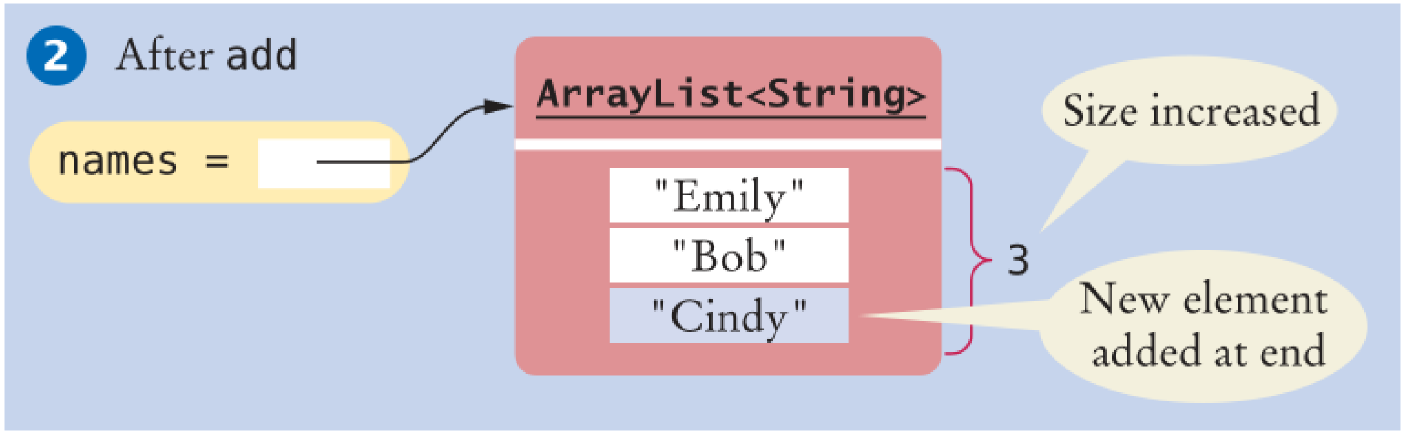
```
ArrayList<String> friends = new ArrayList<String>();

friends.add("Cindy");
String name = friends.get(i);
friends.set(i, "Harry");
```

- Need to import `java.util.ArrayList`
- Cannot use primitives (`int`, `double`…) as element type

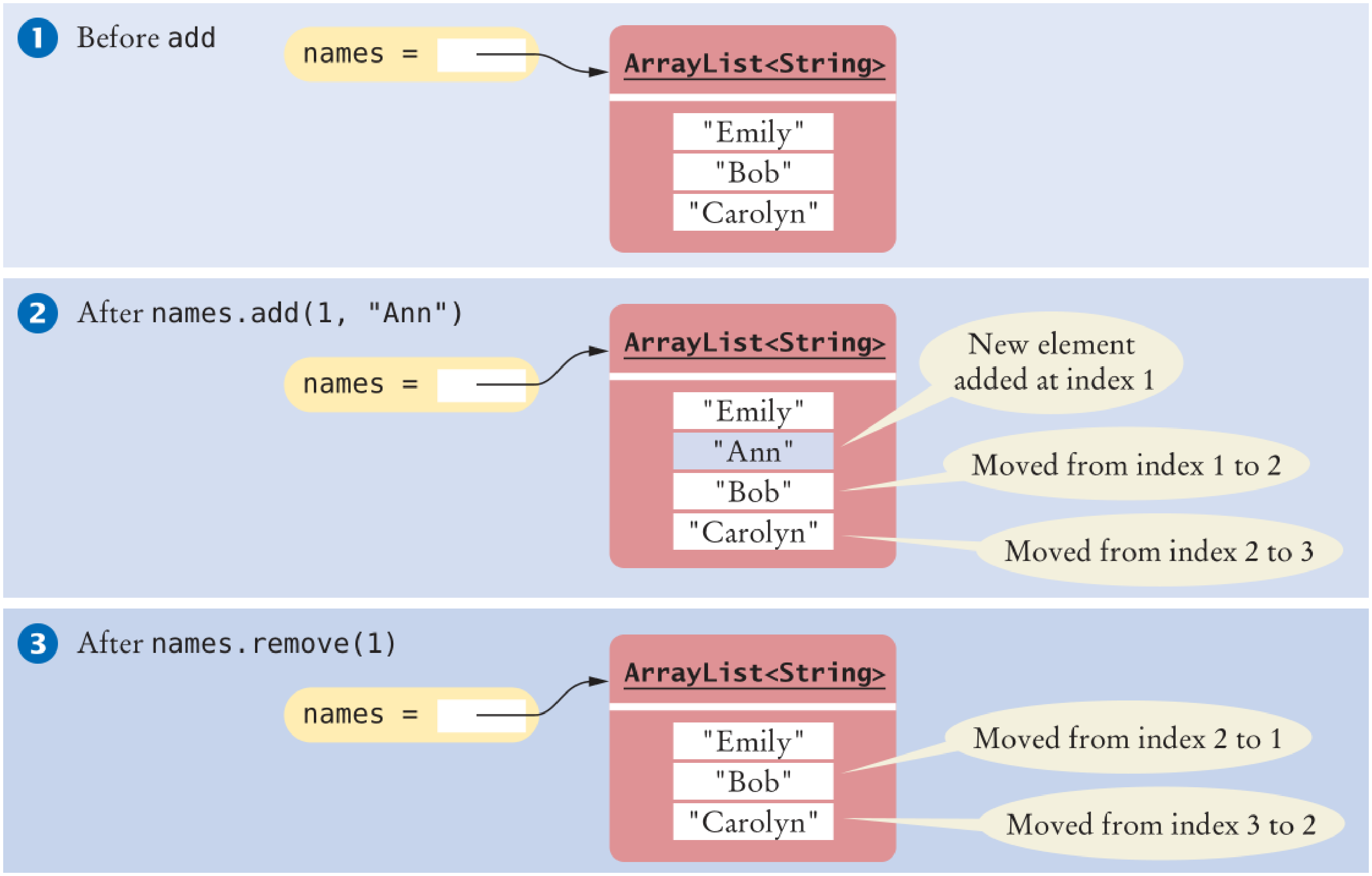
How elements are added

```
names.add("Cindy");
```





Inserting & Removing elements



The enhanced for loop

- **Example:** calculate array sum

```
double[] values = ...;
double total = 0;
for (double element : values) {
    total = total + element;
}
```

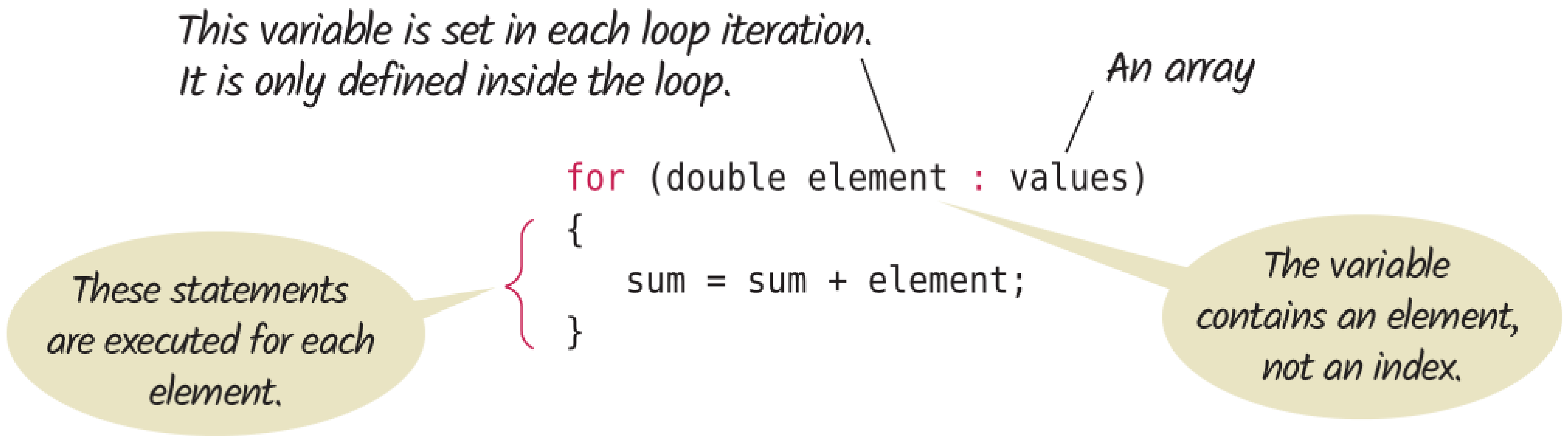
- Read this loop as: “for each `element` in `values`”
- It is equivalent to:

```
for (int i = 0; i < values.length; i++) {
    total = total + values[i];
}
```

The enhanced for loop

- **Syntax:**

```
for (TypeName variable : collection) {
    statements
}
```



```
for (double element : values)
{
   sum = sum + element;
}
```

This variable is set in each loop iteration. It is only defined inside the loop.
An array
These statements are executed for each element.
The variable contains an element, not an index.

Enhanced for loop: ArrayList

```
ArrayList<String> names = ...;
for (String name : names) {
    System.out.println(name);
}
```

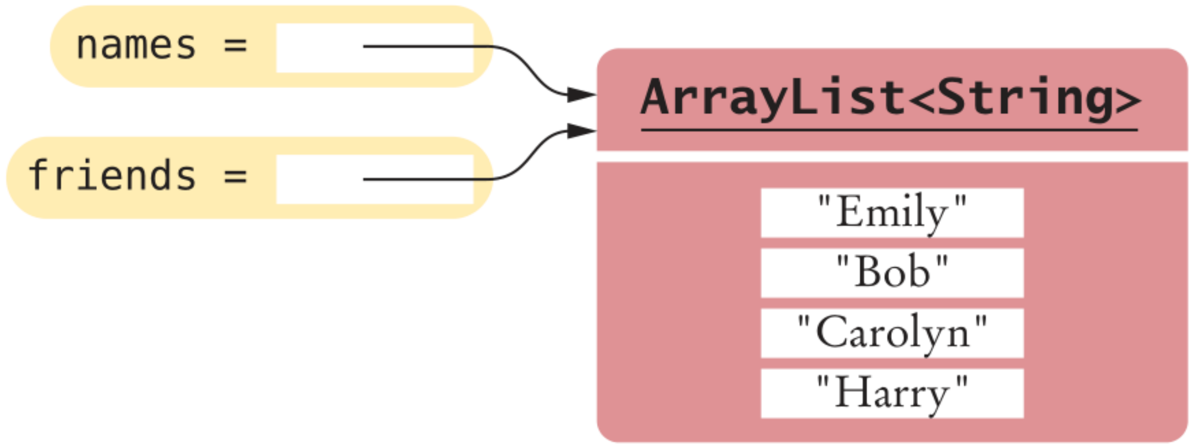
is equivalent to

```
for (int i = 0; i < names.size(); i++) {
    String name = names.get(i);
    System.out.println(name);
}
```

Copying an ArrayList

```
ArrayList<String> friends = names;
```

- The above statement only copy the ArrayList's reference.



- To really make a copy of an ArrayList (a new object):

```
ArrayList<String> list2 = new ArrayList<String>(list1);
```

Wrapper classes

- **Problem:** unable to use primitives in ArrayList
  - and many other situations
- **Solution:** have reference-type equivalents of primitive types
- The conversion between wrapper object and primitive value happens automatically

```
// auto-boxing
Double x = 1.5;
// auto-unboxing
double y = x;
```

Primitive Type	Wrapper Class
byte	Byte
boolean	Boolean
char	Character
double	Double
float	Float
int	Integer
long	Long
short	Short

StringBuilder

- **Problem:** modifying Strings is slow (computationally expensive).

```java
String s = "abc";
String r = "";
for (int i = 0; i < s.length(); i++) {
    r = s.charAt(i) + r;
}
```

- **Solution:** `StringBuilder` improves performance.

```java
StringBuilder sb = new StringBuilder();
for (int i = s.length() - 1; i >= 0; i++) {
    sb.append(s.charAt(i));
}
String r = sb.toString();
```

StringBuilder's methods

- `append(CharSequence s)`: appends the specified character sequence to this StringBuilder.
- `insert(int offset, CharSequence s)`: inserts the specified CharSequence into this StringBuilder.
- `reverse()`: causes the internal character sequence to be replaced by the reverse of itself.
- `delete(int start, int end)`: removes a substring from this StringBuilder.
- `deleteCharAt(int index)`: removes a char at the specified position.
- StringBuilder also supports String operations: `charAt()`, `indexOf()`, `substring()`, `length()`, `replace()`

Review: while loop

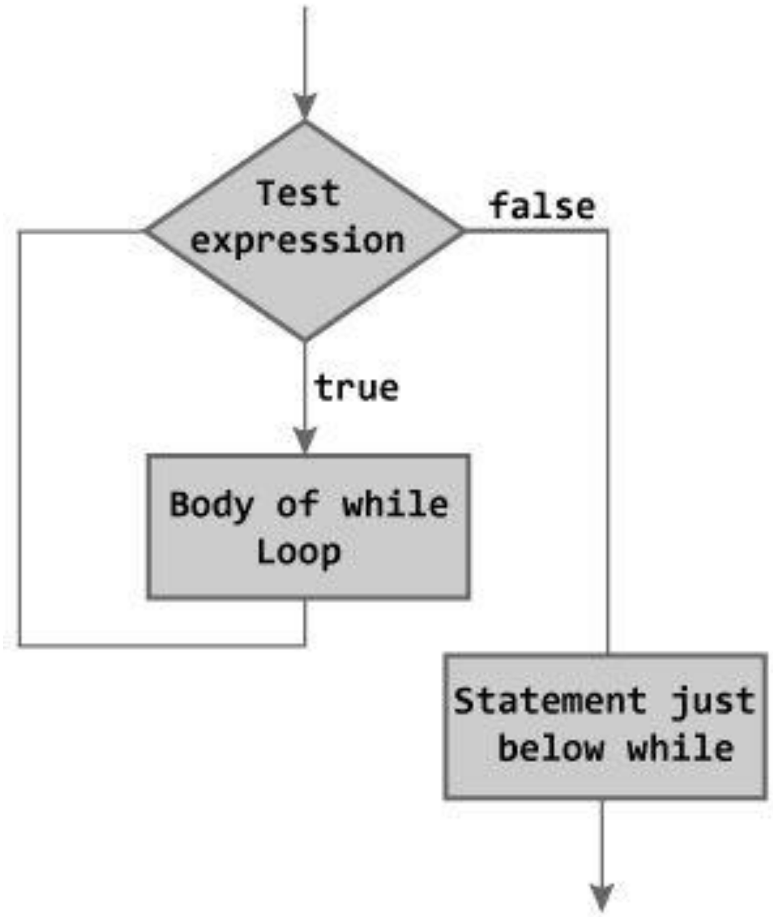


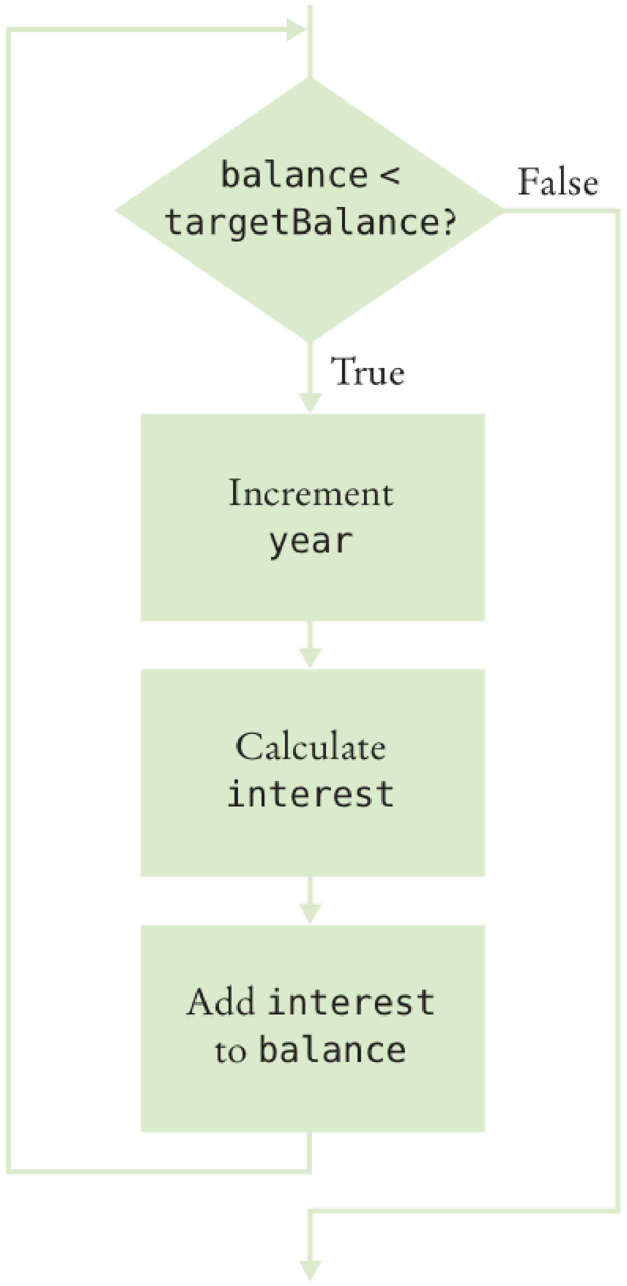
Figure: Flowchart of while Loop

Review: while loop

- Consider a while loop to calculate money investment.

```java
double balance = 100;
double rate = 7.3;
double targetBalance = 200;
int year = 0;
while (balance < targetBalance) {
    year++;
    double interest = balance * rate / 100;
    balance = balance + interest;
}
System.out.println(year + " years.");
```

FlowChart of the code example



While loop trace table

year	interest	balance	balance < targetBalance (200)
0	N/A	100.00	TRUE
1	7.30	107.30	TRUE
2	7.83	115.13	TRUE
3	8.40	123.54	TRUE
4	9.02	132.56	TRUE
5	9.68	142.23	TRUE
6	10.38	152.62	TRUE
7	11.14	163.76	TRUE
8	11.95	175.71	TRUE
9	12.83	188.54	TRUE
10	13.76	202.30	FALSE

While loop debugging text

- Print out the values to trace them.
  - Useful when dealing with loop problems.

```
while (balance < targetBalance) {
    year++;
    double interest = balance * rate / 100;
    balance = balance + interest;
    System.out.println("Year: " + year);
    System.out.println("Interest: " + interest);
    System.out.println("Balance: " + balance);
}
```