Static methods

Remember the main method header?

public static void main(String[] args)

What does static mean?

- associates a method with a particular class name
- any method can call a static method either:
  - directly from within same class OR
  - using class name from outside class

Scope of Local Variables

A local variable: a variable defined inside a method.

Scope: the part of the program where the variable can be referenced.

The scope of a local variable starts from its declaration and continues to the end of the block that contains the variable.

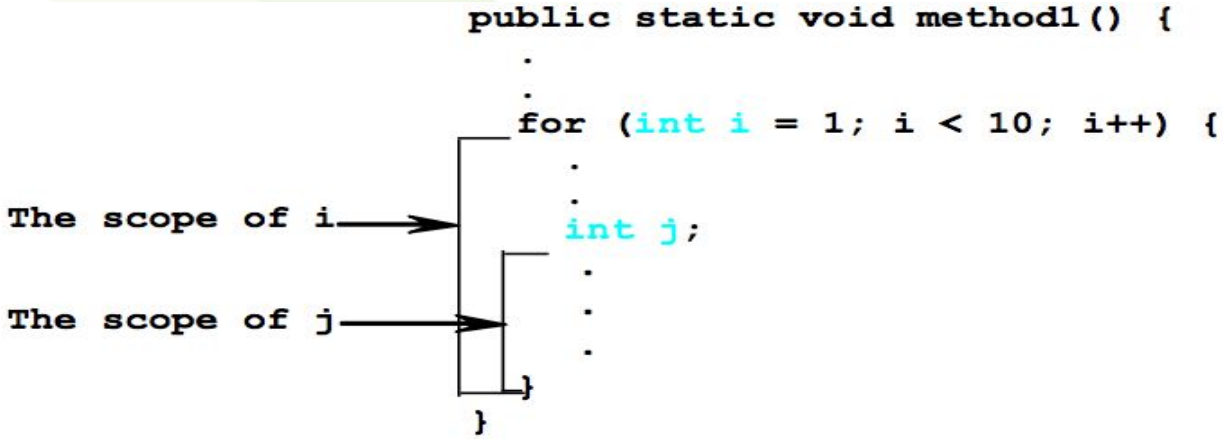
A local variable must be declared before it can be used.

You can declare a local variable with the same name multiple times in different non-nesting blocks in a method, but you cannot declare a local variable twice in nested blocks.

Scope of Local Variables

A variable declared in the initial action part of a for loop header has its scope in the entire loop.

A variable declared inside a for loop body has its scope limited in the loop body from its declaration and to the end of the block that contains the variable.



Scope of Local Variables

```
// Fine with no errors
public static void correctMethod() {
        int x = 1; int y = 1; // i is declared
        for (int i = 1; i < 10; i++) {
                x += i;
                }
        // i is declared again
         for (int i = 1; i < 10; i++) {
                y += i;
        }
}
```

Scope of Local Variables

It is fine to declare i in two non-nesting blocks

```
public static void method1() {
  int x = 1;
  int y = 1;

  for (int i = 1; i < 10; i++) {
    x += i;
  }

  for (int i = 1; i < 10; i++) {
    y += i;
  }
}
```

It is wrong to declare i in two nesting blocks

```
public static void method2() {

  int i = 1;
  int sum = 0;

  for (int i = 1; i < 10; i++)
    sum += i;
  }
}
```

Scope of Local Variables

```
// With errors
public static void incorrectMethod() {
        int x = 1;
        int y = 1;
        for (int i = 1; i < 10; i++) {
         int x = 0;
         x += i;
        }
 }
```

Static methods

Remember the main method header?

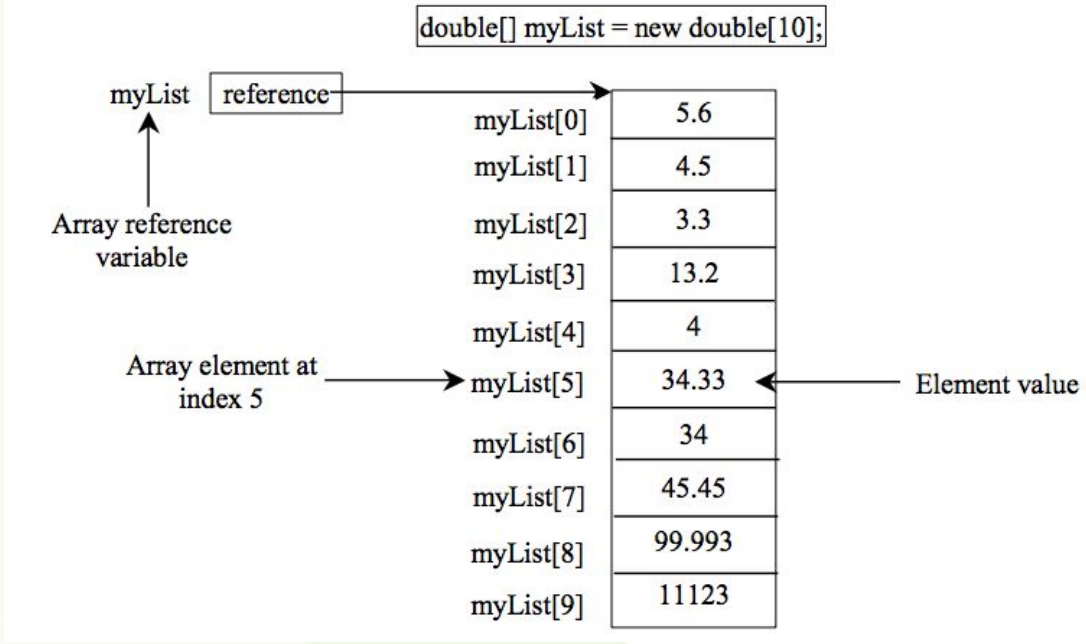
public static void main(String[] args)

What does static mean?

- associates a method with a particular class name
- any method can call a static method either:
  - directly from within same class OR
  - using class name from outside class

Arrays

Array is a data structure that represents a collection of the same types of data.



Declaring Array Variables

datatype[] arrayRefVar;

Example:

double[] myList;

datatype arrayRefVar[];

// This style is allowed, but not preferred Example:

double myList[];

Creating Array

arrayRefVar = new datatype[arraySize];

Example:

myList = new double[10];

myList[0] references the first element in the array.

myList[9] references the last element in the array.

Declaring and Creating in One Step

```
datatype[] arrayRefVar = new datatype[arraySize];

double[] myList = new double[10];

datatype arrayRefVar[] = new datatype[arraySize];

double myList[] = new double[10];
```

Array

Once an array is created, its size is fixed.

It cannot be changed.

You can find its size using:

arrayRefVar.length

Example:

myList.length

returns 10

Array

When an array is created, its elements are assigned the default value of 0

for the numeric primitive data types,

'\u0000' for char types,

and false for boolean types.

Array

When an array is created, its elements are assigned the default value of 0

for the numeric primitive data types,

'\u0000' for char types,

and false for boolean types.

Array

The array elements are accessed through the index.

The array indices are 0-based, i.e.,

it starts from 0 to arrayRefVar.length-1.

Each element in the array is represented using the following syntax, known as an indexed variable:

```
arrayRefVar[index];
```

Array

After an array is created, an indexed variable can be used in the same way as a regular variable.

```
myList[2]= myList[0] + myList[1];
```

Array

Declaring, creating, initializing in one step:

```
double[] myList = {1.9, 2.9, 3.4, 3.5};
```

This shorthand syntax must be in one statement.

Array

```
double[] myList = {1.9, 2.9, 3.4, 3.5};
```

This shorthand notation is equivalent to the following statements:

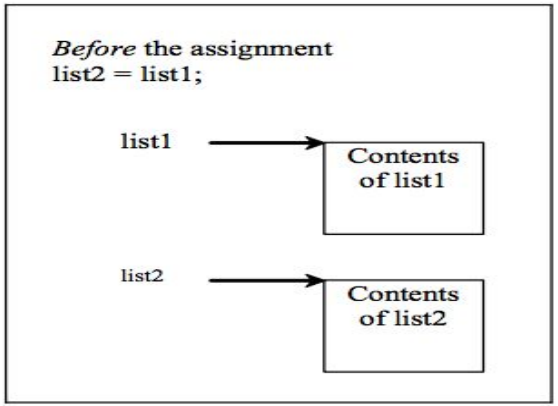
```
double[] myList = new double[4];
    myList[0] = 1.9;
    myList[1] = 2.9;
    myList[2] = 3.4;
    myList[3] = 3.5;
```

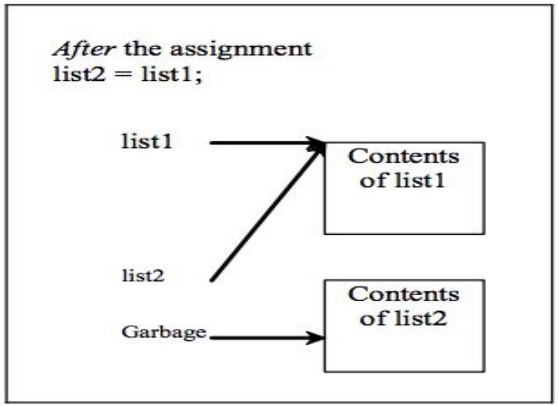
Copying an Array

Often, in a program, you need to duplicate an array or a part of an array.

Using the assignment statement (=), you re-direct the pointer:

list2 = list1; You don't copy with "=" !





Copying an Array

Using a loop:

```
int[] sourceArray={2, 3, 1, 5, 10};
int[] targetArray=new int[sourceArray.length];
for (int i = 0; i < sourceArray.length; i++)
    targetArray[i] = sourceArray[i];
```