Methods and Parameters

Passing Parameters to Methods

```
Public class PassParameters {

  public static void main (String[] args) {
    int a,b;
    doSomething(a,b);

  }
  public static doSomething(int x, int y) {

  }

}
```

Actual parameters

```
Public class PassParameters {

  public static void main (String[] args) {
    int a,b;
    doSomething(a,b);

  }
  public static doSomething(int x, int y) {

  }

}
```

Formal parameters

Parameter Passing

- Primitive type parameters are passed by value
- Pass by value means a copy of the value of the parameter is given to the method
- Each variable in the main program and the method has its own memory location

Before the call:

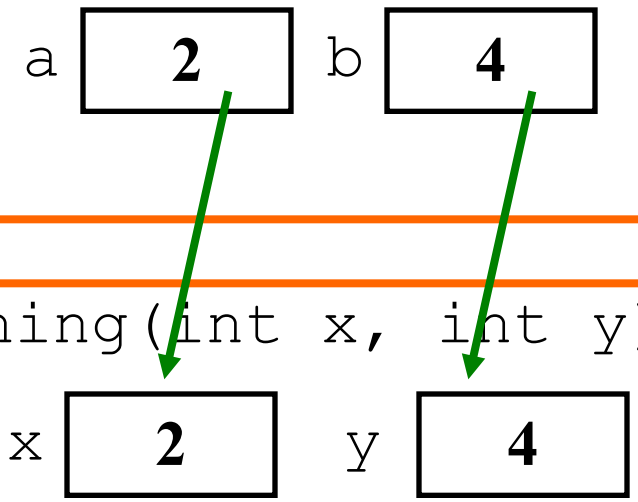
```
public class PassParameters {

  public static void main (String[] args) {
    int a=2,b=4;
    doSomething(a,b);
  }
  public static doSomething(int x, int y) {

  }

}
```

a **2** b **4**

x **?** y **?**

After the call:

```
public class PassParameters {

  public static void main (String[] args) {
    int a=2,b=4;
    doSomething(a,b);

  }
  public static doSomething(int x, int y) {

  }

}
```



Because the method works on *copies* no changes happen in the main program

```
public class PassParameters {

  public static void main (String[] args) {
    int a=2,b=4;
    doSomething(a,b);
  }
  public static doSomething(int x, int y) {
    x=10;
    y=20;
  }
}
```

a **2** b **4**

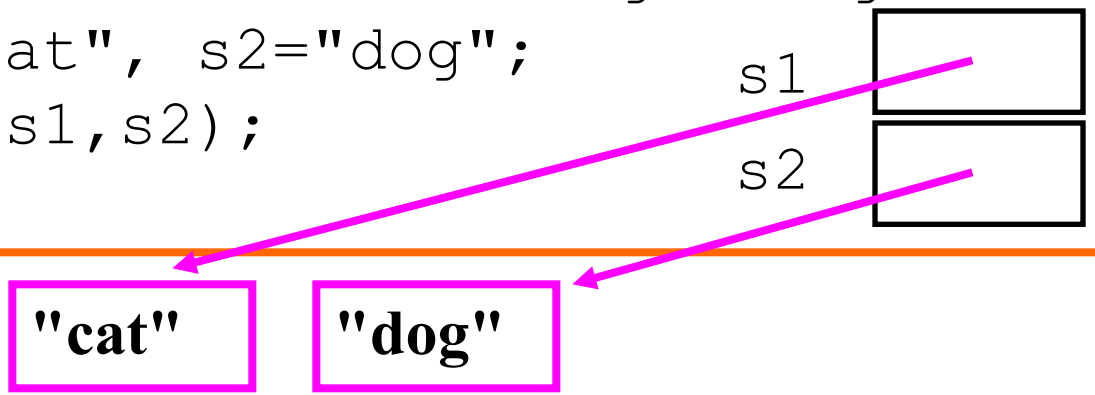
x **10** y **20**

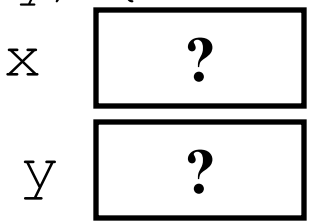
Parameter Passing

- Object type parameters are passed by reference.
- Pass by reference means a reference to the the parameter is given to the method.
- Each variable in the main program and the method has its own memory location for the reference

Before the call:

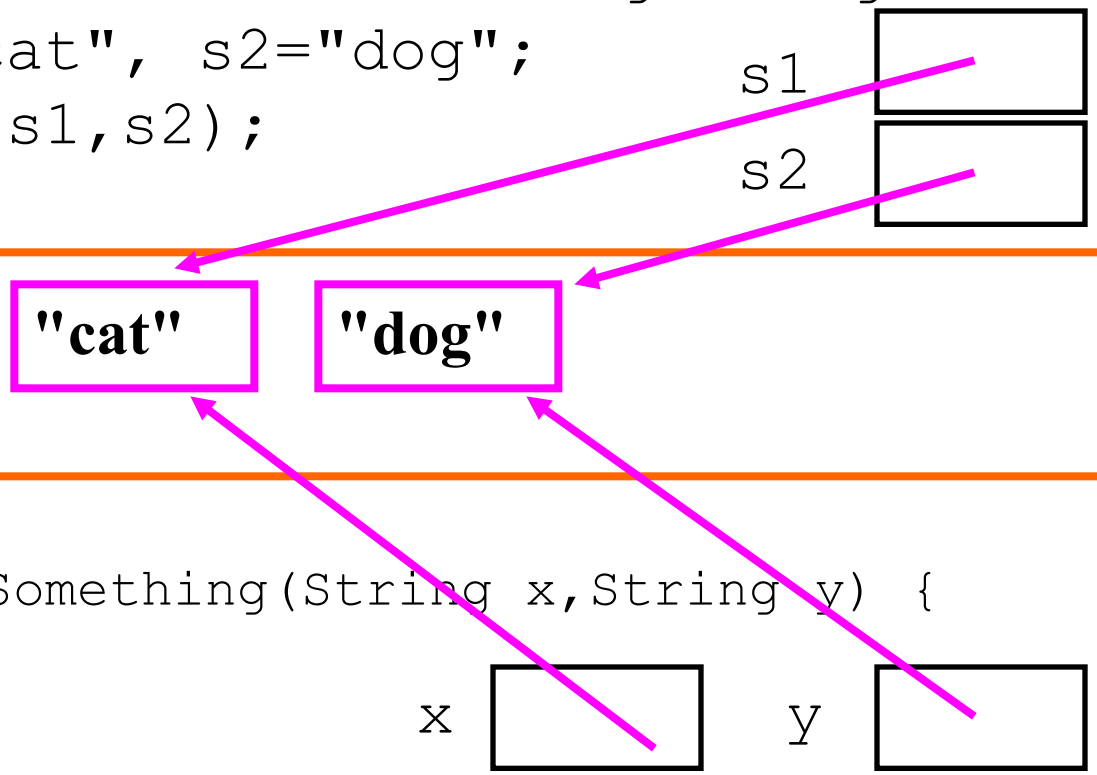
```
public class PassParameters {

  public static void main (String[] args) {
    String s1="cat", s2="dog";
    doSomething(s1,s2);

  }

  public static doSomething(String x,String y) {

  }

}
```





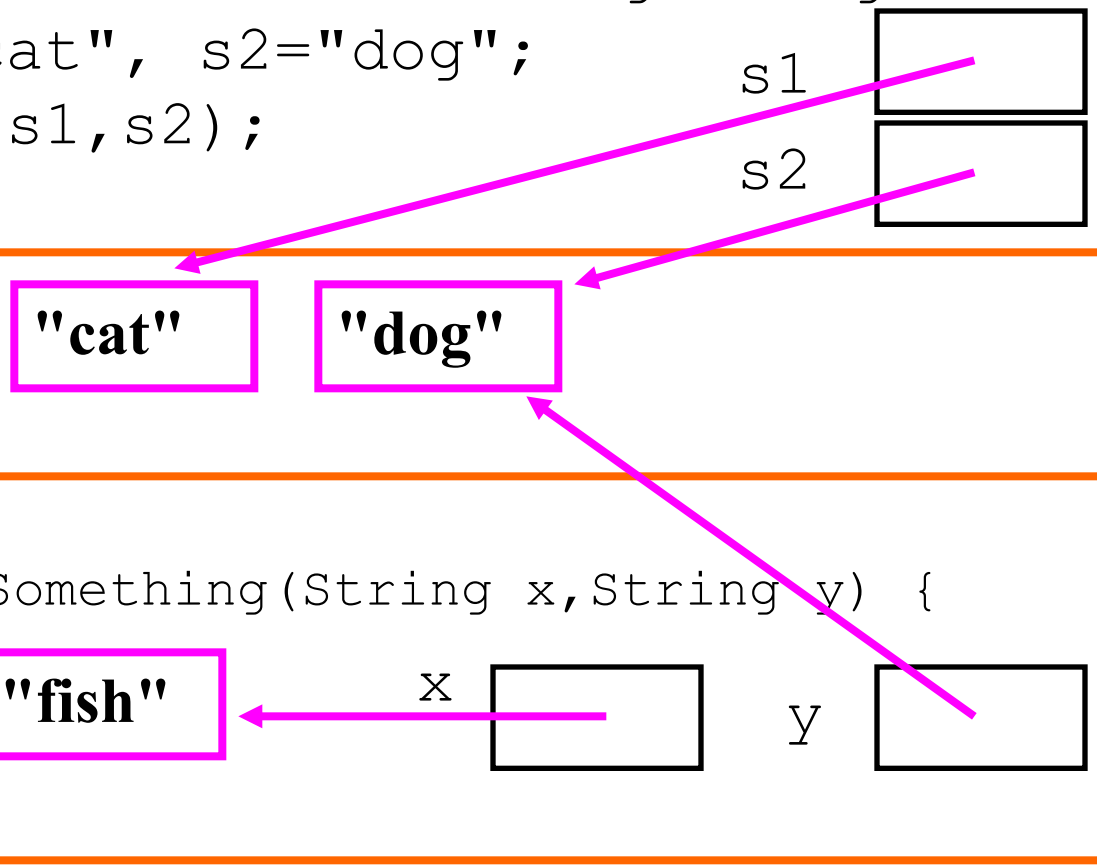
After the call:

```
public class PassParameters {

  public static void main (String[] args) {
    String s1="cat", s2="dog";
    doSomething(s1,s2);

  }

  public static doSomething(String x,String y) {

  }

}
```

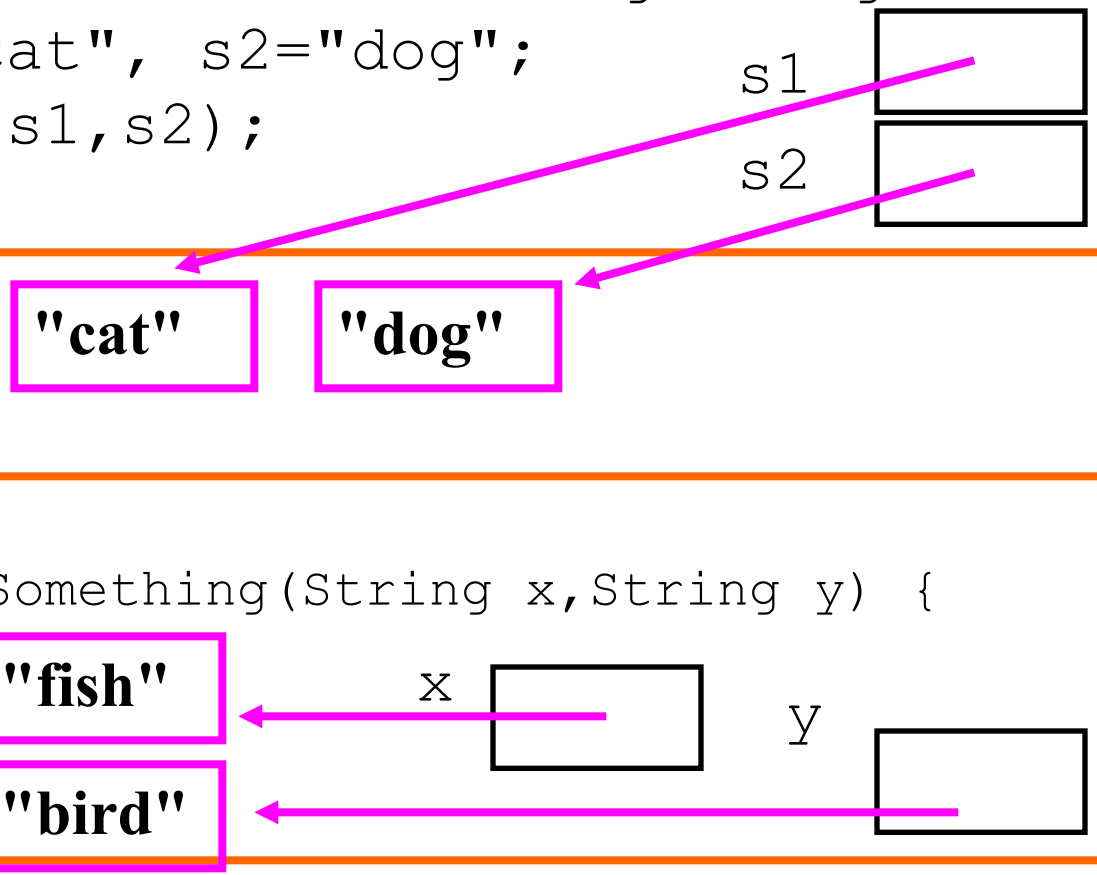


```
public class StringTest {
  public static void main(String[] args) {
    String s1="cat", s2="dog";
    System.out.println("s1 is "+s1+", s2 is "+s2);
    doSomething(s1,s2);
    System.out.println("s1 is "+s1+", s2 is "+s2);
  }
  public static void doSomething(String x, String y) {
    System.out.println(x);
    System.out.println(y);
    x = "fish";
    y = "bird";
    }
}
```

```
public class PassParameters {

  public static void main (String[] args) {
    String s1="cat", s2="dog";
    doSomething(s1,s2);
  }

  public static doSomething(String x,String y) {
    x="fish"
  }
}
```



```
public class PassParameters {

  public static void main (String[] args) {
    String s1="cat", s2="dog";
    doSomething(s1,s2);
  }

  public static doSomething(String x,String y) {
    y="bird"
  }
}
```



```
public class StringTest {
  public static void main(String[] args) {
    String s1="cat", s2="dog";
    System.out.println("s1 is "+s1+", s2 is "+s2);
    doSomething(s1,s2);
    System.out.println("s1 is "+s1+", s2 is "+s2);
  }
  public static void doSomething(String x, String y) {
    System.out.println(x);
    System.out.println(y);
    x = "fish";
    y = "bird";
    }
}
```

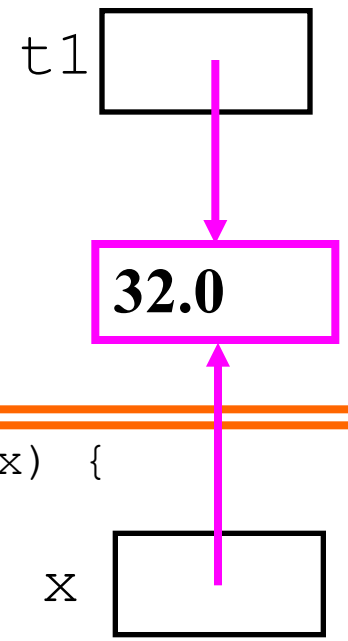
```
s1 is cat, s2 is dog
cat
dog
s1 is cat, s2 is dog
```

Changing an Object

- Since we have a reference to an object in a method, any changes made to that object are permanent.
- Changes happen to the object whose reference is used in a method call.

```
temp.setTemperature(32.0);

numbers[3]=27;
```

```
public class TemperatureTest {
  public static void main(String[] args) {
    Temperature t1= new Temperature(32.0f);
    System.out.println("t1 is "+t1);
    doSomething(t1);
    System.out.println("t1 is "+t1);
  }
  public static void doSomething(Temperature x) {
     x.setTemperature(98.6f);
  }
}
```

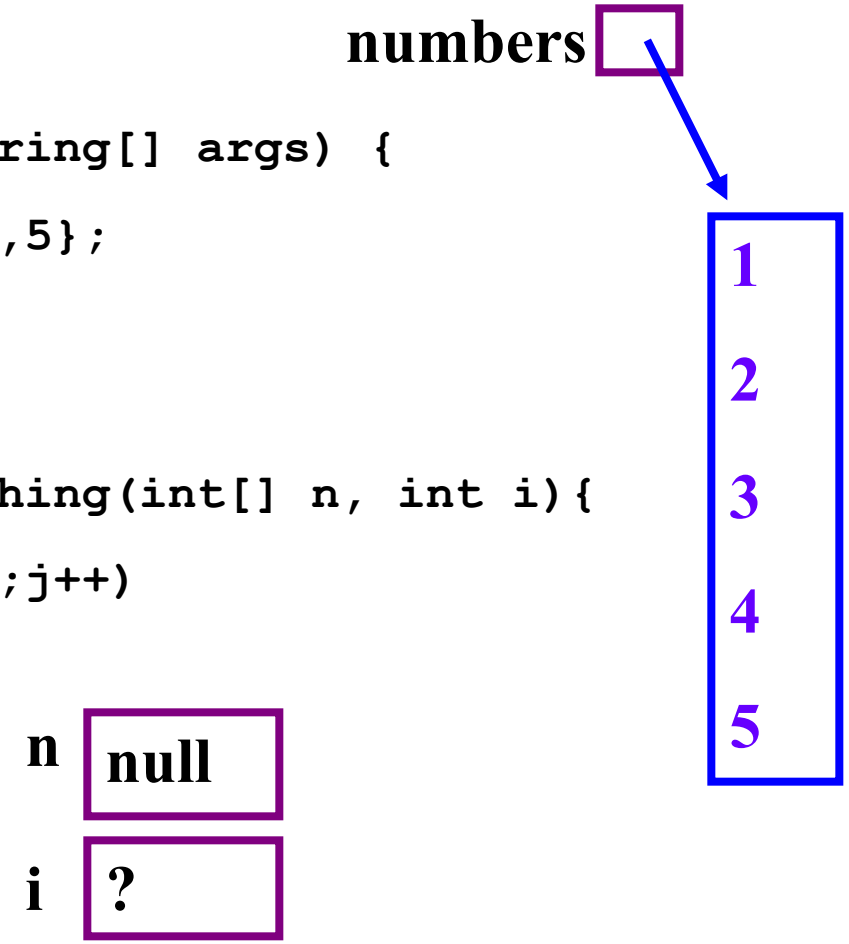



```
t1 is 32.0
t1 is 98.6
```

```
public class ArrayTest {
  public static void main(String[] args) {
    int[] numbers = {1,2,3,4,5};
    doSomething(numbers,2);
}
  public static void doSomething(int[] n, int i){
    for (int j=0; j<n.length;j++)
      n[j] += i;
  }
}
```

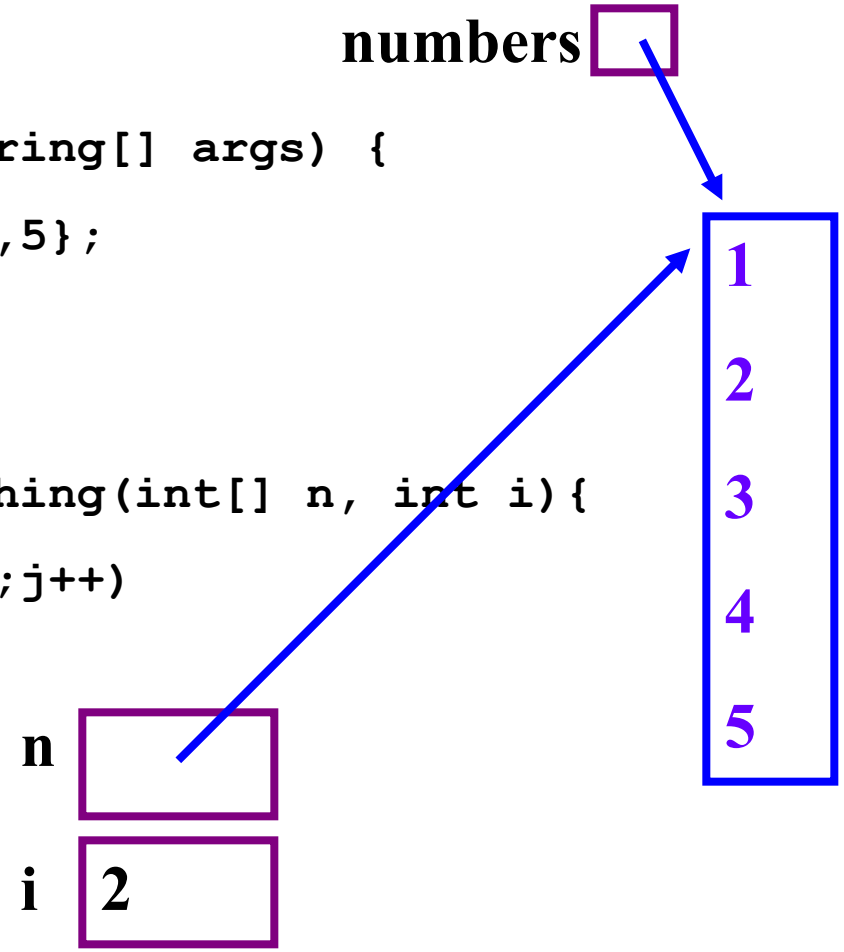
Before the call:

```
public class ArrayTest {
  public static void main(String[] args) {
    int[] numbers = {1,2,3,4,5};
    doSomething(numbers,2);
}
  public static void doSomething(int[] n, int i){
    for (int j=0; j<n.length;j++)
      n[j] += i;
  }
}
```



After the call:

```
public class ArrayTest {
  public static void main(String[] args) {
    int[] numbers = {1,2,3,4,5};
    doSomething(numbers,2);
}
  public static void doSomething(int[] n, int i){
    for (int j=0; j<n.length;j++)
      n[j] += i;
  }
}
```



So what is the output?

```
public class ArrayTest {
  public static void main(String[] args) {
    int[] numbers = {1,2,3,4,5};
    for (int k=0;k<numbers.length;k++)
        System.out.print(numbers[k]+" ");
    doSomething(numbers,2);
    for (int k=0;k<numbers.length;k++)
        System.out.print(numbers[k]+" ");
  }
  public static void doSomething(int[] n, int i){
    for (int j=0; j<n.length;j++)
      n[j] += i;
  }
}
```

The output is:

```
1 2 3 4 5 3 4 5 6 7
```

```
public class ArrayTest {
  public static void main(String[] args) {
    int[] numbers = {1,2,3,4,5};
    for (int k=0;k<numbers.length;k++)
        System.out.print(numbers[k]+" ");
    doSomething(numbers,2);
    for (int k=0;k<numbers.length;k++)
        System.out.print(numbers[k]+" ");
  }
  public static void doSomething(int[] n, int i){
    for (int j=0; j<n.length;j++)
      n[j] += i;
  }
}
```

From Selection sort:

```
private static void selectionSort
                                (short[] array, int length) {
for ( int i = 0; i < length - 1; i++ ) {
  int indexLowest = i;
  for ( int j = i + 1; j < length; j++ )
    if ( array[j] < array[indexLowest] )
       indexLowest = j;
       if ( array[indexLowest] < array[i] ) {
         short temp = array[indexLowest];
         array[indexLowest] = array[i];
         array[i] = temp;
       }
   } // for i
} // method selectionSort
```

From Selection sort:

```
private static void selectionSort
                                (short[] array, int length) {
for ( int i = 0; i < length - 1; i++ ) {
  int indexLowest = i;
  for ( int j = i + 1; j < length; j++ )
    if ( array[j] < array[indexLowest] )
       indexLowest = j;
       if ( array[indexLowest] < array[i] ) {
         short temp = array[indexLowest];
         array[indexLowest] = array[i];
         array[i] = temp;
       }
   } // for i
} // method selectionSort
```

Swap values at indexLowest and i

```
private static void selectionSort
                                (short[] array, int length) {
for ( int i = 0; i < length - 1; i++ ) {
  int indexLowest = i;
  for ( int j = i + 1; j < length; j++ )
    if ( array[j] < array[indexLowest] )
       indexLowest = j;
       if ( array[indexLowest] < array[i] ) {
         swap( ????, ????);
       }
   } // for i
} // method selectionSort
```

```
private static void selectionSort
                                (short[] array, int length) {
for ( int i = 0; i < length - 1; i++ ) {
  int indexLowest = i;
  for ( int j = i + 1; j < length; j++ )
    if ( array[j] < array[indexLowest] )
       indexLowest = j;
       if ( array[indexLowest] < array[i] ) {
         swap( array[indexLowest],array[i]);
       }
   } // for i
} // method selectionSort
```

```
swap( array[indexLowest],array[i]);
```

```
public static void swap (int x, int y) {
   int temp;
   temp = x;
   x=y;
   y=temp;
}
```

```
private static void selectionSort
                                (short[] array, int length) {
for ( int i = 0; i < length - 1; i++ ) {
  int indexLowest = i;
  for ( int j = i + 1; j < length; j++ )
    if ( array[j] < array[indexLowest] )
       indexLowest = j;
       if ( array[indexLowest] < array[i] ) {
         swap( array,indexLowest,i);
       }
   } // for i
} // method selectionSort
```

```
  swap( array,indexLowest,i);


public static void swap
                (int[] a, int x, int y) {
  int temp;
  temp = a[x];
  a[x]=a[y];
  a[y]=temp;
}
```