

# Time in Java



Measuring Time using Java libraries

## By the end of this video you will be able to...

- How to use Java timing libraries to measure execution time

```
public static void main(String [] args) {  
    // set some size n  
    double array[] = new double[n];  
  
    // fill the array with contents (random)  
  
    selectionSort(array);  
  
}
```

**How long did  
selectionSort run?**

```
public static void main(String [] args) {  
    // set some size n  
    double array[] = new double[n];  
  
    // fill the array with contents (random)  
  
    selectionSort(array);  
  
}
```

**How long did  
selectionSort run?**

```
public static void main(String [] args) {  
    // set some size n  
    double array[] = new double[n];  
    // fill the array with contents (random)  
    selectionSort(array);  
}
```

**How long did  
selectionSort run?**

```
public static void main(String [] args) {  
    // set some size n  
    double array[] = new double[n];  
  
    // fill the array with contents (random)  
    selectionSort(array);  
}
```

**We want just this time**

# Class System

`static long`

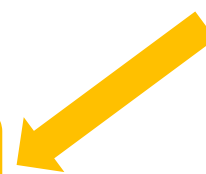
`nanoTime()`

Returns the current value of the running Java Virtual Machine's high-resolution time source, in nanoseconds.


```
public static void main(String [] args) {  
    // set some size n  
    double array[] = new double[n];  
  
    // fill the array with contents (random)  
  
    long startTime = System.nanoTime();  
  
    selectionSort(array);  
  
}
```



```
public static void main(String [] args) {  
    // set some size n  
    double array[] = new double[n];  
  
    // fill the array with contents (random)  
  
    long startTime = System.nanoTime();  
  
    selectionSort(array);  
  
    long endTime = System.nanoTime();  
  
}
```



```
public static void main(String [] args) {  
    // set some size n  
    double array[] = new double[n];  
  
    // fill the array with contents (random)  
  
    long startTime = System.nanoTime();  
  
    selectionSort(array);  
  
    long endTime = System.nanoTime();  
  
    double estTime = (endTime-startTime) /  
                     1000000000.0;  
    System.out.println(estTime);  
}
```



# Idea for Analyzing our Sorts

For increasing sizes of  $n$

Print  $n$

Create a randomized array of size  $n$   
Time selection sort, print outcome

Create a randomized array of size  $n$   
Time quick sort, print outcome

# Idea for Analyzing our Sorts

For increasing sizes of  $n$

Print  $n$

Create a randomized array of size  $n$   
Time selection sort, print outcome

Create a randomized array of size  $n$   
Time quick sort, print outcome

# Idea for Analyzing our Sorts

For increasing sizes of  $n$

## Print $n$

Create a randomized array of size  $n$   
Time selection sort, print outcome

Create a randomized array of size  $n$   
Time quick sort, print outcome

# Idea for Analyzing our Sorts

For increasing sizes of  $n$

Print  $n$

**Create a randomized array of size  $n$**

Time selection sort, print outcome

Create a randomized array of size  $n$

Time quick sort, print outcome

# Idea for Analyzing our Sorts

For increasing sizes of  $n$

Print  $n$

Create a randomized array of size  $n$

**Time selection sort, print outcome**

Create a randomized array of size  $n$

Time quick sort, print outcome

# Idea for Analyzing our Sorts

For increasing sizes of  $n$

Print  $n$

Create a randomized array of size  $n$   
Time selection sort, print outcome

Create a randomized array of size  $n$   
Time quick sort, print outcome



# Results

| n      | Selection (s) | Quick (s)   |
|--------|---------------|-------------|
| 10000  | 0.112887621   | 0.001323534 |
| 20000  | 0.397227565   | 0.001568662 |
| 30000  | 0.580318935   | 0.002420492 |
| 40000  | 1.020979179   | 0.003304295 |
| 50000  | 1.605557659   | 0.004232703 |
| 60000  | 2.340087449   | 0.004983088 |
| 70000  | 3.264979954   | 0.006035047 |
| 80000  | 4.097073897   | 0.006989112 |
| 90000  | 5.285101776   | 0.007900941 |
| 100000 | 6.57904119    | 0.008538038 |

**Quicksort is  
fantastic**

# Results

| n      | Selection (s) | Quick (s)   |
|--------|---------------|-------------|
| 10000  | 0.112887621   | 0.001323534 |
| 20000  | 0.397227565   | 0.001568662 |
| 30000  | 0.580318935   | 0.002420492 |
| 40000  | 1.020979179   | 0.003304295 |
| 50000  | 1.605557659   | 0.004232703 |
| 60000  | 2.340087449   | 0.004983088 |
| 70000  | 3.264979954   | 0.006035047 |
| 80000  | 4.097073897   | 0.006989112 |
| 90000  | 5.285101776   | 0.007900941 |
| 100000 | 6.57904119    | 0.008538038 |

**Quicksort is  
fantastic**

**We can do  
more...**