

INTRODUCTION TO
COMPUTER SCIENCE
Rutgers University

# 5. Arrays



https://introcs.cs.rutgers.edu

https://www.socrative.com/

- Login
- Student
- ROOM: TREESCS111
- TEACHER'S QUESTION:
- *What do you need to be reviewed? How are you doing????? What would you like to tell me?*



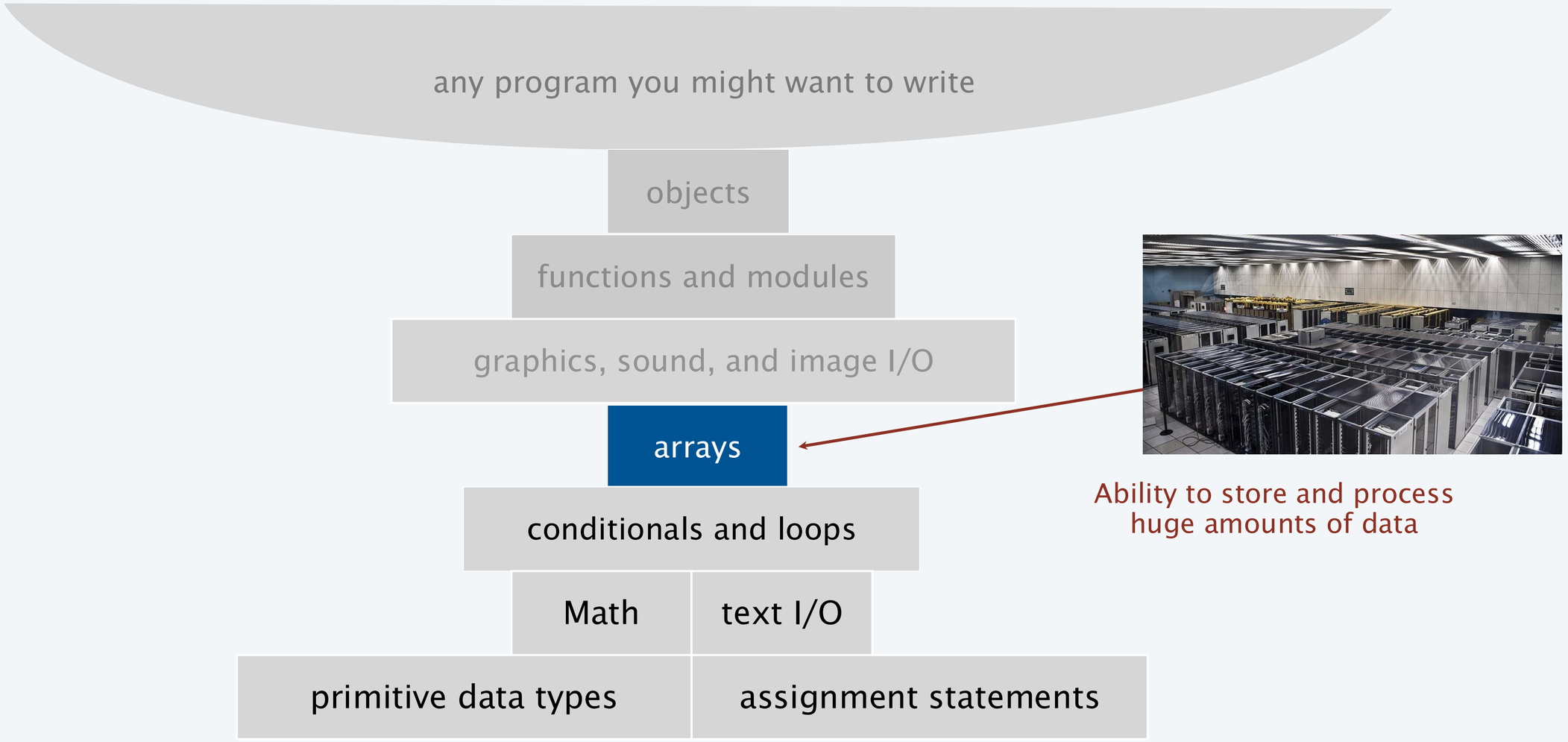
INTRODUCTION TO
COMPUTER SCIENCE
Rutgers University

# 5. Arrays

- Array data structure
- Declare, create and initialize
- Memory representation and allocation
- Array bounds
- Examples of array-processing code
- Playing Cards
- Two-dimensional arrays

adopted and modified from Slides by Sedgewick and Wayne

# Basic Building Blocks of Programming



# Dr. Seuss: "Too Many Daves"

https://stuff.mit.edu/people/dpolicar/writing/poetry/poems/tooManyDaves.html

https://www.youtube.com/watch?v=8YLCQodll30 (Video)

WHAT IN THE WORLD DOES THIS HAVE TO DO WITH OUR LESSON ON **ARRAYS** TODAY?????

# Data Structures

A data structure is an arrangement of data that enables efficient processing by a program.

An array is an *indexed* sequence of values of the same type.

Examples.

- 52 playing cards in a deck.
- 100 thousand students in an online class.
- 1 billion pixels in a digital image.
- 4 billion nucleotides in a DNA strand.
- 73 billion Google queries per year.
- 86 billion neurons in the brain.
- 50 trillion cells in the human body.
- $6.02 \times 10^{23}$ particles in a mole.

| *index* | *value* |
|---|---|
| 0 | 2♥ |
| 1 | 6♠ |
| 2 | A♦ |
| 3 | A♥ |
| ... | |
| 49 | 3♣ |
| 50 | K♣ |
| 51 | 4♠ |



Main purpose. Facilitate storage and manipulation of data.



INTRODUCTION TO COMPUTER SCIENCE

Rutgers University

# Arrays

- An OVERVIEW with simple examples

# What we have been using: *args* *Arrays1a.java*

```java
public class Arrays1a{
    Run | Debug
    public static void main(String[] args){
        // input of 3 values causes Runtime Error
        for(int i = 0; i < 5; i++){
            System.out.print(args[i]+ " ");
        }
        System.out.println();

    }
}
```

*args.length = 3*

| *index* | *value* |
|---|---|
| 0 | 5 |
| 1 | 4 |
| 2 | 3 |

← RUNTIME ERROR

```
$ javac Arrays1a.java
$ java Arrays1a 5 4 3
5 4 3 Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 3 out of bounds for length 3
        at Arrays1a.main(Arrays1a.java:5)
```

# What we have been using: *args* *Arrays1aFix.java*

```java
public class Arrays1aFix{
    Run | Debug
    public static void main(String[] args){

        for(int i = 0; i < args.length; i++){
            System.out.print(args[i]+ " ");
        }
        System.out.println();

    }
}
```

*args.length = 7*

| *index* | *value* |
|---|---|
| 0 | 5 |
| 1 | 4 |
| 2 | 3 |
| 3 | 7 |
| 4 | 8 |
| 5 | 10 |
| 6 | 2 |

```
$ javac Arrays1aFix.java
$ java Arrays1aFix 5 4 3 7 8 10 2
5 4 3 7 8 10 2
```

***args.length*** is the length of the array ***args***



# 5. Arrays

- Array data structure
- Declare, create and initialize
- Memory representation and allocation
- Array bounds
- Typical array-processing code
- Playing Cards
- Two-dimensional arrays

# Declare, create and initialize



**10 values, without arrays**

```
double a0 = 0.0;
double a1 = 0.0;
double a2 = 0.0;
double a3 = 0.0;
double a4 = 0.0;
double a5 = 0.0;
double a6 = 0.0;
double a7 = 0.0;
double a8 = 0.0;
double a9 = 0.0;
...
a4 = 3.0;
...
a8 = 8.0;
...
double x = a4 + a8;
```



tedious and error-prone code

**10 values, with an array**

```
double[] a;
a = new double[10];
...
a[4] = 3.0;
...
a[8] = 8.0;
...
double x = a[4] + a[8];
```



an easy alternative

**1 million values, with an array**

```
double[] a;
a = new double[1000000];
...
a[234567] = 3.0;
...
a[876543] = 8.0;
...
double x = a[234567] + a[876543];
```



scales to handle huge amounts of data

# create and initialize



**100 values, without explicit initialization**

```
int[]  a;
a = new int[100];
```

In Java, all number array elements are initialized to 0
However, this might not be true for other languages.

```
boolean[]  b = new boolean[5]
```

In Java, all boolean array elements are initialized to false.

**5 values, initialize with a list**

```
double[] a =  {1, 2.0, 3, 4, 5} ;
```

Initialize an array with a list

The number of elements in array a is a.length**.** ***a.length = 5***
The first element of the array a has index 0. ***a[0] = 1***
The last element of the array a has index a.length – 1 ***a[4] = 5***

# Let's take it slowly! *Arrays1b.java*

```java
public class Arrays1b{
    public static void main(String[] args){
        int[] nums = new int[args.length];

        for(int i = 0; i < nums.length; i++){
            nums[i] = Integer.parseInt(args[i]);
        }
        int sum = 0;
        for(int i = 0; i < nums.length; i++){
            System.out.print(nums[i]+ " ");
            sum += nums[i];
        }
        System.out.println();
        System.out.println("The sum is: " + sum);
    }
}
```

Declare an array ***nums*** with 5 elements.
Array **nums**: 0 0 0 0 0

Reassigning 5 values to array **nums** from input.

Traverse and print elements of **nums**, accumulating the values in variable **sum**.

Output the sum.

```
$ javac Arrays1b.java
$ java Arrays1b 5 4 3 2 1
5 4 3 2 1
The sum is: 15
```

# Memory representation of an array

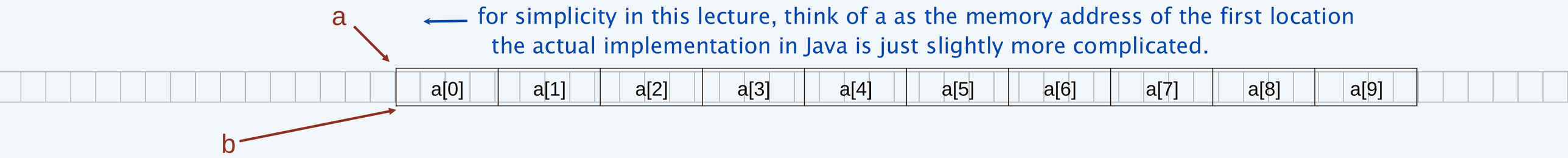


An array is an indexed sequence of values of the same type.

A computer's memory is *also* an indexed sequence of memory locations.

- Each primitive type value occupies a fixed number of locations.
- *Array values are stored in contiguous locations.*

← stay tuned for many details




Critical concepts

- Indices start at 0.
- Given i, the operation of accessing the value a[i] is extremely efficient.
- The assignment b = a makes the variables b and a refer to the same array.

it does *not* copy the array,
as with primitive types
(stay tuned for details)

## *Arrays1c.java*

```java
public class Arrays1c{
    public static void main(String[] args){
        int[] a = {1, 2, 3, 4, 5};
        System.out.println("array a");
        for(int i = 0; i < a.length; i++){
            System.out.print(a[i]+ " ");
        }
        System.out.println();

        int[] b = a;
        System.out.println("array b");
        for(int i = 0; i < b.length; i++){
            System.out.print(b[i]+ " ");
        }
        System.out.println();
        a[0]= 22;
        System.out.println("b array again");
        for(int i = 0; i < b.length; i++){
            System.out.print(b[i]+ " ");
        }
        System.out.println();
    }
}
```

```
$ java Arrays1c
array a
1 2 3 4 5
array b
1 2 3 4 5
b array again
22 2 3 4 5
```

# Java language support for arrays



## Basic support

| *operation* | *typical code* |
|---|---|
| Declare an array | double[] a; |
| Create an array of a given length | a = new double[1000]; |
| Refer to an array entry by index | a[i] = b[j] + c[k]; |
| Refer to the length of an array | a.length; |

## Initialization options

| *operation* | *typical code* |
|---|---|
| Default initialization to 0 for numeric types | a = new double[1000]; |
| Declare, create and initialize in one statement | double[] a = new double[1000]; |
| Initialize to literal values | double[] x = { 0.3, 0.6, 0.1 }; |

no need to use a loop like

```
for (int i = 0; i < 1000; i++)
  a[i] = 0.0;
```

BUT cost of creating an array is proportional to its length.

# In Java

In Java, a newly created array is **always** filled with a known, default value:

- `0` for numbers
- `false` for boolean

- the null(empty) char for chars (with Unicode number zero for char)

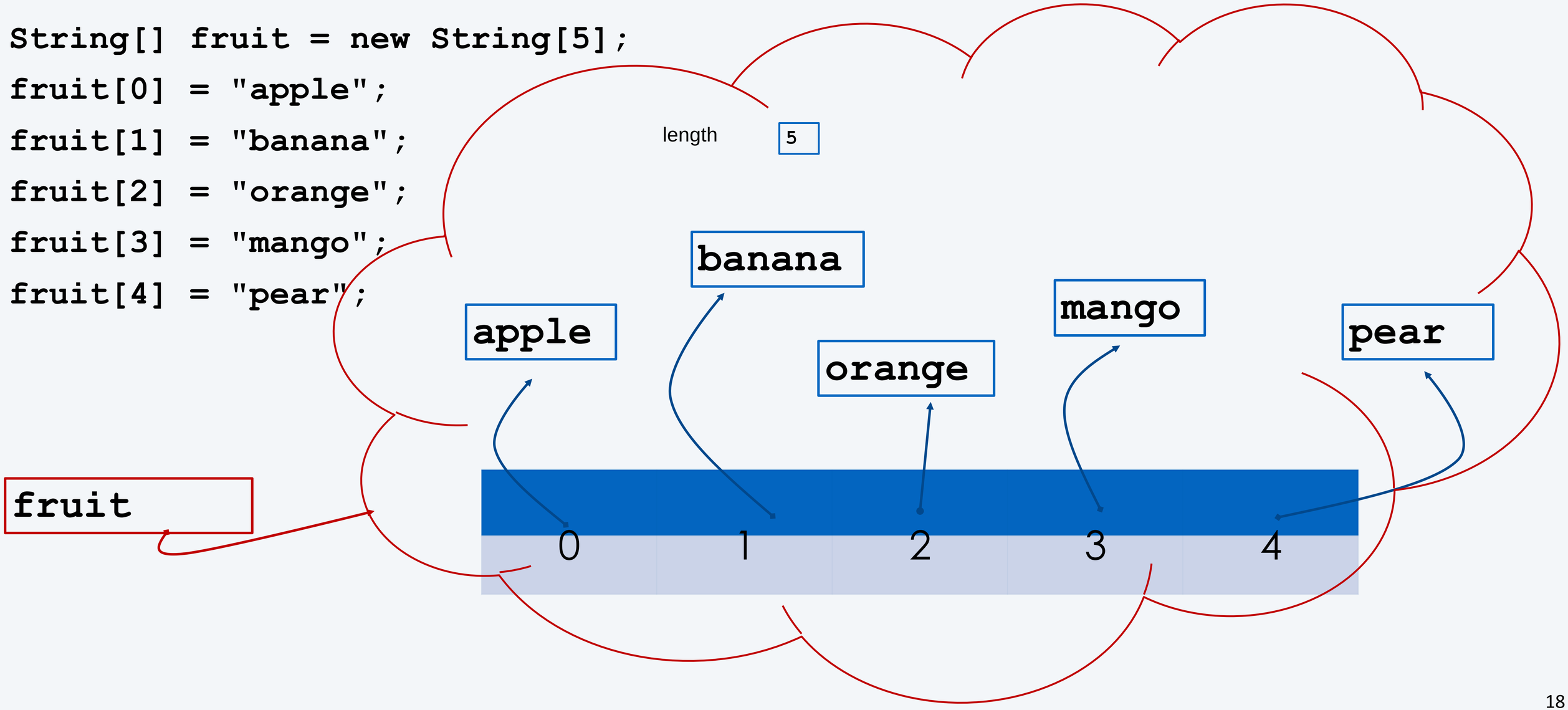
```
char[] myCharArr = new char[4];
for ( int i = 0; i < myCharArr.length; i++ ) {
  System.out.println ( i + "*" + myCharArr[i] + "*");
  }
```

- **`null`** for objects.

```
0**
1**
2**
3**
```

# Arrays of Strings (objects) *Strings are NOT primitive types*

```
String[] fruit = new String[5];
fruit[0] = "apple";
fruit[1] = "banana";
fruit[2] = "orange";
fruit[3] = "mango";
fruit[4] = "pear";
```



# Copying an array



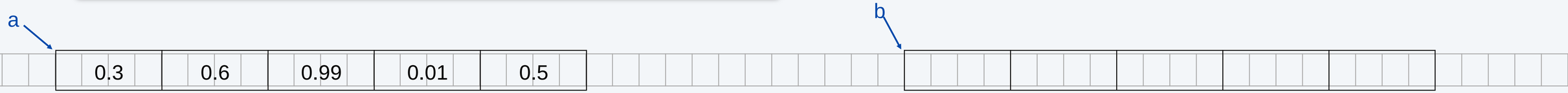
To copy an array, create a new array , then copy all the values.

```
double[] b = new double[a.length];
for (int i = 0; i < a.length; i++)
   b[i] = a[i];
```




Important note: The code b = a does *not* copy an array (it makes b and a refer to the same array).

```
double[] b = new double[a.length];
b = a;
```



# Programming with arrays: typical examples

**Access command-line args in system array**

```
int stake  = Integer.parseInt(args[0]);
int goal   = Integer.parseInt(args[1]);
int trials = Integer.parseInt(args[2]);
```

**Create an array with N random values**

```
double[] a = new double[N];
for (int i = 0; i < a.length; i++)
   a[i] = Math.random();
```

**Compute the average N array values**

```
double sum = 0.0;
for (int i = 0; i < a.length; i++)
   sum += a[i];
double average = sum / N;
```

**Copy to another array**

```
double[] b = new double[a.length];
for (int i = 0; i < b.length; i++)
   b[i] = a[i];
```

**Print array values, one per line**

```
for (int i = 0; i < a.length; i++)
   System.out.println(a[i]);
```

**Find the maximum of array values**

```
double max = a[0];
for (int i = 1; i < a.length; i++)
   if (a[i] > max) max = a[i];
```

# Pop quiz 1 on arrays VERY IMPORTANT



Q. What does the following code print?

```
public class PQarray1
{
    public static void main(String[] args)
    {
        int[] a = new int[6];
        int[] b = new int[a.length];

        b = a;
        for (int i = 1; i < b.length; i++)
            b[i] = i;

        for (int i = 0; i < a.length; i++)
            System.out.print(a[i] + " ");
        System.out.println();

        for (int i = 0; i < b.length; i++)
            System.out.print(b[i] + " ");
        System.out.println();
    }
}
```

# Pop quiz 1 on arrays – VERY IMPORTANT



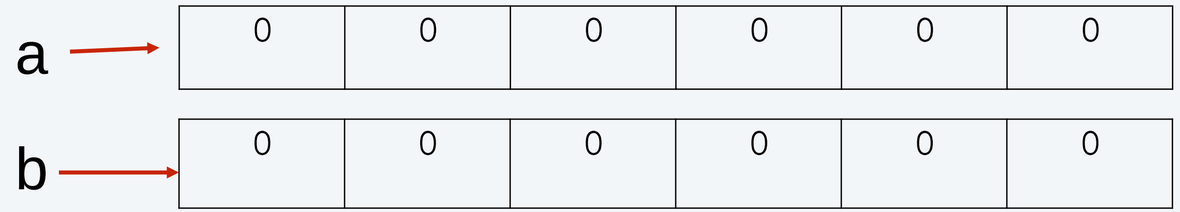
## Q. What does the following code print?

```
public class PQarray1
{
    public static void main(String[] args)
    {
        int[] a = new int[6];
        int[] b = new int[a.length];

        b = a;
        for (int i = 1; i < b.length; i++)
            b[i] = i;

        for (int i = 0; i < a.length; i++)
            System.out.print(a[i] + " ");
        System.out.println();

        for (int i = 0; i < b.length; i++)
            System.out.print(b[i] + " ");
        System.out.println();
    }
}
```



# Pop quiz 1 on arrays

LO 5.1d

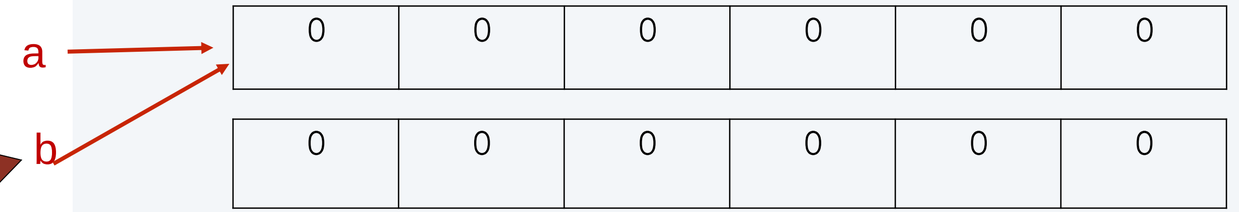
## Q. What does the following code print?

```
public class PQarray1
{
    public static void main(String[] args)
    {
        int[] a = new int[6];
        int[] b = new int[a.length];

        b = a;
        for (int i = 1; i < b.length; i++)
            b[i] = i;

        for (int i = 0; i < a.length; i++)
            System.out.print(a[i] + " ");
        System.out.println();

        for (int i = 0; i < b.length; i++)
            System.out.print(b[i] + " ");
        System.out.println();
    }
}
```




After this, b and a refer to the same array

# Pop quiz 1 on arrays

LO 5.1d

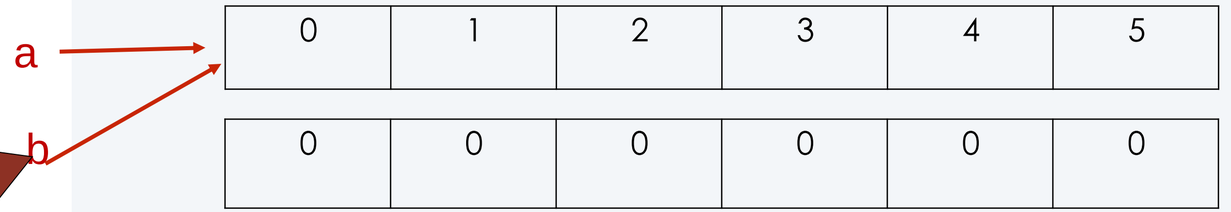
## Q. What does the following code print?

```
public class PQarray1
{
    public static void main(String[] args)
    {
        int[] a = new int[6];
        int[] b = new int[a.length];

        b = a;
        for (int i = 1; i < b.length; i++)
            b[i] = i;

        for (int i = 0; i < a.length; i++)
            System.out.print(a[i] + " ");
        System.out.println();

        for (int i = 0; i < b.length; i++)
            System.out.print(b[i] + " ");
        System.out.println();
    }
}
```




After this, b and a refer to the same array

A.

```
% java PQ4_1
0 1 2 3 4 5
0 1 2 3 4 5
```

# Programming with arrays: typical bugs

LO 5.1f

**Array index out of bounds**

```
double[] a = new double[10];
for (int i = 1; i <= 10; i++)
   a[i] = Math.random();
```

No a[10] (and a[0] unused)





**Uninitialized array**

```
double[] a;
for (int i = 0; i < 9; i++)
   a[i] = Math.random();
```

Never created the array





**Undeclared variable**

```
a = new double[10];
for (int i = 0; i < 10; i++)
   a[i] = Math.random();
```

What type of data does a refer to?

# Programming with arrays: typical bugs

LO 5.1f

**Array equality: true** if the two specified arrays are *equal* to one another. Two arrays are considered equal if both arrays contain the same number of elements, and all corresponding pairs of elements in the two arrays are equal.

```
System.out.println("Checking a==b");
System.out.println (a==b);
```




Really checking references!

FIX:

```
System.out.println("Checking term by term");
  int i = 0;
  boolean check = true;
  while (i < a.length && check){
    if(a[i] != b[i]){
      check = false;
    }
    i++;
  }
  System.out.println(check);
```



# 5. Arrays

- The first data structure
- Declare, create and initialize
- Memory representation and allocation
- Array bounds
- Typical array-processing code
- Playing Cards
- Two-dimensional arrays

# Array bounds



**Bounds.** An array of size n contains n cells to store data

Example

```
int[] a = new int[10];   <-- declaration

a[0] = 10;               <-- legal

a[9] = 20;               <-- legal

a[-1] = 10;              <-- Illegal. ArrayIndexOutOfBoundsException

a[10] = 10;              <-- Illegal. Illegal. ArrayIndexOutOfBoundsException
```

**Array index out of bounds exception.** Is thrown when accessing a cell illegally



# 5. Arrays

- The first data structure
- Declare, create and initialize
- Memory representation and allocation
- Array bounds
- Examples of array-processing code
- Playing Cards
- Two-dimensional arrays

# Manipulating arrays with for-each

Q. How do we use **for-each** construct (enhanced for loop) with arrays?

**Print array values, one per line**

```
for (int i = 0; i < N; i++)
   System.out.println(a[i]);
```

**Print array values, one per line**

```
for (int elt : a)
   System.out.println(elt);
```

**The meaning:** ***For each element, `elt`, in array `a`, do what's in the loop. Cannot refer to individual elements by index number with this loop.***

# Examples. Manipulating 1D arrays



Q. How do we traverse and display elements in an array, in order and reverse order?

A.

```
// in order
for (int i = 0; i < a.length; i++)
    System.out.println(a[i]);
// reverse order
for (int i = a.length-1; i >= 0; i--)
    System.out.println(a[i]);
```

Q. How do we reverse the elements in an integer array?

A.

```
// reverse array
for (int i = 0; i < a.length/2; i++) {
    int temp = a[i];
    a[i] = a[a.length-i-1];
    a[a.length-i-1] = temp
}
```

SWAP

# Examples. Manipulating 1D arrays



Q. How do we find and report min and max elements in an integer array?

A.

```
int max = Integer.MIN_VALUE;
int min = Integer.MAX_VALUE;
for (int i = 0; i < a.length; i++) {
    if (a[i] > max)
        max = a[i];
    if (a[i] < min)
        min = a[i];
}
System.out.println(min);
System.out.println(max);
```

Q. How do we find and report **index** of the min and max elements in an integer array?

A.

```
int max = Integer.MIN_VALUE;
int min = Integer.MAX_VALUE;
int maxIndex = 0, minIndex=0;
for (int i = 0; i < a.length; i++) {
    if (a[i] > max)
        {max = a[i];  maxIndex = i;}
    if (a[i] < min)
        { min = a[i];   minIndex = i;}
}
System.out.println(min);
System.out.println(max);
```

# Examples. Manipulating 1D arrays - alternate initializations



Q. How do we find and report min and max elements in an integer array?

A.

```
int max = a[0];
int min = a[0];
for (int elt : a) {
    if (elt> max)
        max = elt;
    if (elt < min)
        min =elt;
}
System.out.println(min);
System.out.println(max);
```

CAN USE FOR EACH LOOP HERE

Q. How do we find and report **index** of the min and max elements in an integer array?

A.

```
int max = a[0];
int min = a[0];
int maxIndex = 0, minIndex=0;
for (int i = 0; i < a.length; i++) {
    if (a[i] > max)
        {max = a[i];  maxIndex = i;}
    if (a[i] < min)
        { min = a[i];   minIndex = i;}
}
System.out.println(min);
System.out.println(max);
```

CANNOT USE FOR EACH LOOP HERE

# Examples. Manipulating 1D arrays – Enhanced for loop



Q. How do we find and report min and max elements in an integer array?

```
int max = a[0];
int min = a[0];
for (int i = 0; i < a.length; i++) {
    if (a[i] > max)
        max = a[i];
    if (a[i] < min)
        min = a[i];
}
System.out.println(min);
System.out.println(max);
```

## Using FOR-EACH LOOP

```
int max = a[0];
int min = a[0];
for ( int elt: a ) {
    if (elt > max)
        max = elt;
    if (elt < min)
        min = elt;
}
System.out.println(min);
System.out.println(max);
```

# Examples. Manipulating 1D arrays



Q. How do find the average of numerical values in an integer array?

A.

CAN USE FOR EACH LOOP HERE

```
double sum = 0;
for (int i = 0; i < a.length; i++) {
   sum += a[i]
}
System.out.println(sum/a.length);
```

Q. Why is sum declared to be double?

A. Otherwise sum/a.length is performed as an integer division?

Q. How do we exchange values of two elements in an integer array?

A.

```
// exchange two values at indices i and j
if (i < 0 || i >= a.length) {
    System.out.println("Array index out of bounds");
    return;
}
if (j < 0 || j >= a.length) {
    System.out.println("Array index out of bounds");
    return;
}
int temp = a[i];
a[i] = a[j];
a[j] = temp;
```

CANNOT USE FOR EACH LOOP HERE

SWAP

# Examples. Manipulating 1D arrays

LO 5.1g

## Q. How do shift elements in an array to right?

### A.

```
for  (int i = a.length-1; I >=1; i--) {
      a[i] = a[i-1];
}
```

Q. What is the value of a[1] after the shift?

A. It is the same as a[0]

Q: How do we “wrap around?”

1 2 3 4 5 → 5 1 2 3 4

*Shifting.java*

## Q. How do shift elements in an array to left?

### A.

```
for  (int i = 0; i < a.length-1; i++) {
      a[i] = a[i+1];
}
```

Q. Which element of the original array is lost after the shift?

A. a[0] is lost

Q: How do we “wrap around?”

1 2 3 4 5 → 2 3 4 5 1

Shifting.java

# Examples. Manipulating 1D arrays

LO 5.1g

Q. How do remove negative integers from an array and replace them with 0?

A.

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

Q. How do remove duplicate values from an array and replace them with 0's?

A.

```
Arrays.sort(a);

for (int i=0; i < a.length ; i++) {
    int j = i+1;
    while (j < a.length && a[j] == a[i])
        { a[j] = 0; j++;}
    if (j == a.length)  break;
    i = j;
}
```

# Basic Array "tasks" → *Basics.java*

- Traverse an array
- Find the largest (smallest) **value in an array**
- Find the average of values in an array
- Find the average of the values in an array excluding the largest value and the smallest value in the array
- Count the number of elements above (or below) the average
- Search for a value in the array: display the index of the first occurrence of the value or display “not found” if the value is not in the array.
- Reverse the values in an array.
- Copy an array
- Shift elements in array to the right/left as specifications describe.

# NJ. Pick 6 Lottery (NJ has numbers 1 – 46)

Let's look again closely at picking THREE random numbers from 1 - 9

```
int a[] = new int[9];
for (int i = 0; i < a.length; i++){
     a[i] = i + 1;
}

//Shuffling numbers
for (int i = 0; i < 3; i++){
    int rand = (int)(Math.random()*9);
 //swap nums in positions i and rand
    int temp = a[i];
    a[i] = a[rand];
    a[rand] = temp;
}
```

numbers chosen were 3, *4, 7*

*NoDupes.java*
Original order: **1 2 3 4 5 6 7 8 9**
Choosing 3 numbers without replacement

| rand/index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| original | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 3: | 4 | 2 | 3 | 1 | 5 | 6 | 7 | 8 | 9 |
| 4: | 4 | 5 | 3 | 1 | 2 | 6 | 7 | 8 | 9 |
| 7: | 4 | 5 | 8 | 1 | 2 | 6 | 7 | 3 | 9 |

Shuffled numbers are: 4 5 8 1 2 6 7 3 9