**Advanced Placement Computer Science**

**Shenendehowa HS mr Hanley**

**Unit 6: Arrays/ArrayList**

**Lesson: One and Two D Arrays**

***Last Updated:*** *12/20/2023*

 ****

Arrays are a powerful tool, they allow a programmer to store data using

To create an array of primitives

You create an array of doubles to store the conversion rates for 10 currencies to USD

To access individual elements inside the array

NOTE: Valid indices are from to ( )

To create an array of references

To create objects that the references point to

To figure out the size of the array

Create an array of 100 doubles and write a loop that will assign the square roots of the integers from 1-100 into your array. Call your array squares

Create an array of 256 Color objects and write a loop that will assign an array of 256 Colors using all of the green intensities but leave the red and blue intensities at 0. (to refresh your memory, a Color object is created by using Color newCol = new Color(1,5,10);)

//where 1 = red intensity from 0-255, 5 = green intensity from 0-255, 10 = blue intensity from 0-255

Arrays can also be created by specifying the initial values and NOT specifying the size

int [ ] daysInMonth = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

What will System.out.println(daysInMonth[5]) output to the console?

Also, you can create an anonymous array object on the fly:

new boolean [ ] { true, false, false, true , true};

What is .length for the anonymous array above?

Passing arrays to methods

public class PunjabiMC {

 //preconditions: values.length > 0

 //postconditions: returns the average of the numbers in the array

 public double average(double[ ] values){ //finish the method

 }}

public class Tester {

 public static void main (String [ ] args) {

 //calling a method and passing an array to it

 PunjabiMC pm = new PunjabiMC();

 double [ ] myNums = {5, 6, 7, 8};

 double avg = pm.average(myNums);

 System.out.println(avg);

}

}

Two dimensional arrays

You can also create two and n dimensional arrays

To create a 2 dimensional array

int [ ] [ ] maze = new int [40] [25]; Figuring out the size of the dimensions of the array

For the first dimension

For the second dimension

If there was a third dimension

If there was a fourth dimension

//You try; assign all values in the maze to 1

Now, Let’s display the maze to the screen

Arrays of primitives simply allocate contiguous memory (if its available) when the new command is issued.

You will recall this chart

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TYPE | CONTAINS | SIZE | MAX | MIN |
| boolean | boolean value | 8 bits | true  | false |
| char | Unicode char | 16 bits  | \uFFFF  | \u0000 |
| byte | signed integer | 8 bits | 127 | -128 |
| short | signed integer | 16 bits  | 32767 | -32768 |
| int | signed integer | 32 bits  | 2147483647 | -2147483648 |
| long | signed integer | 64 bits  | 8.223372 E +18  | -9.223372 E +18 |
| float | floating point | 32 bits  | 3.402823 E +38 1.402398 E -45  | -3.402823 E +38 -1.402398 E -45 |
| double  | floating point  | 64 bits  | 1.797693 E +308  4.940656 E -324  | -1.797693 E +308  -4.940656 E -324 |

How many bytes are allocated for the following arrays?

int myArray[ ] = new int[15];

float yourArray[ ] = new float[10];

double [ ] [ ]grid = new double[10][20];

What happens when you attempt to create an array of references?

You need to loop through all of the references and use the new operator

Use an Array of Students as example

public class Student{

 public String name;

 public int numQuizzes, numPoints; //for simplicity, no private vars!

}

Create an array of 3000 Students called roster (loop through and make 3000 objects)

Set the third student’s name to “Chester”, numQuizzes to 3 and numPoints to 297

Writing a method that returns an array

Write a method that returns an array of Strings and takes a String as a parameter that will break the parameter into its individual letters.

For example

String [] x = convertToLetters(“Bob”);

should return

[0] “B”

[1] “o”

[2] “b”

Some very cool methods to use with Arrays.

**Arrays.sort()**: sorts according to the compareTo method or integer/float/char value

int[]myArray1 = {3,-7,8,10,100,14,0,0,3};

cHere is array");

for(int i=0;i<myArray1.length; i++)

 System.out.print(myArray1[i]+" ");

System.out.println();

Arrays.sort(myArray1);

System.out.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

System.out.println("After Sorting...");

for(int i=0;i<myArray1.length; i++)

 System.out.print(myArray1[i]+" ");

System.out.println();

OUTPUT:

Here is array

3 -7 8 10 100 14 0 0 3

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

After Sorting...

-7 0 0 3 3 8 10 14 100

Also works with Strings and Doubles and Chars

String myFriend = "Luke the Duke";

char[] myCharArray = myFriend.toCharArray();

Arrays.sort(myCharArray);

for (int i = 0; i < myCharArray.length; i++)

 System.out.print(myCharArray[i] + " ");

System.out.println("");

OUTPUT: (2 Spaces in the beginning as space is ASCII 32)

 D L e e e h k k t u u

**Arrays.fill()**

int myArray2[] = new int[10];

Arrays.fill(myArray2,-1);

for (int i = 0; i < myArray2.length; i++)

 System.out.print(myArray2[i] + " ");

System.out.println("");

OUTPUT:

-1 -1 -1 -1 -1 -1 -1 -1 -1 -1

**Arrays.copyOf()**

System.out.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

int[] myArray3 = {3, -7, 8, 10, 100, 14, 0, 0, 3};

int[] myArray4 = Arrays.copyOf(myArray3, 5); //5 is the new size

for (int i = 0; i < myArray4.length; i++)

 System.out.print(myArray4[i] + " ");

System.out.println("");

OUTPUT:

3 -7 8 10 100

**Arrays.compare()**

It returns 0 if the array is equal to the other array.

It returns a value less than 0 is returned if the array is lexicographically  less than the other array in

It returns a value greater than 0 if the array is lexicographically greater than the other array (more characters).

A null array is lexicographically less than a non-null array, and the two arrays are considered equal if both are null so that it will print 0 in this case.

System.out.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

System.out.println(Arrays.compare(myArray3, myArray4));

System.out.println(Arrays.compare(myArray4, myArray3));

OUTPUT:

4

-4

**String.split()** Returns an array of Strings based on a delimiter

String x = "It was a dark and rainy night...";

String[] words = x.split(" ");

System.out.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

System.out.println("After Splitting...");

for (int i = 0; i < words.length; i++)

 System.out.println(words[i] + " ");

OUTPUT:

After Splitting...

It

was

a

dark

and

rainy

night...

**System.arraycopy()** copies a certain number of elements from one array to another

//ArrayCopy

public static void arraycopy(Object source\_arr, int sourcePos,

 Object dest\_arr, int destPos, int len)

Parameters :

source\_arr : array to be copied from

sourcePos : starting position in source array from where to copy

dest\_arr : array to be copied in

destPos : starting position in destination array, where to copy in

len : total no. of components to be copied.

String[] countries1 = {"Australia","Chile","Mozambique","Madagascar"};

String[] countries2 = {"USA", "Canada", "Haiti", "RSA", "Malaysia"};

System.arraycopy(countries1, 1, countries2, 3, 2);

System.out.println("The new array 2...");

for (int i = 0; i < countries2.length; i++)

 System.out.println(countries2[i] + " ");

OUTPUT:

The new array 2...

USA

Canada

Haiti

Chile

Mozambique