Worksheet 6: CS121

SI Arnesh

Study Notes: Arrays & Indexing

1) Arrays

Arrays are 'data structures'. A fancy way to describe something quite simple:

2) Initializing an Array

Initializing an array of ints to store the 10 day weather forecast:

```
int weatherTenDays[] = {50,47,42,56,64,58,55,48,49,50};
```

50	47	42	56	64	58	55	48	49	50

3) Length of the Array

Similar to a String, the 'length' of an array is simply the number of elements. In this case it is 10.

Different from a String however, the way you access the length of an array is not length(), but just length.

```
int arrayLength = weatherTenDays.length; //arrayLength now stores 10
```

4) Indexing an Array

Arrays are indexed just like Strings, starting from 0 and ending at length -1!

50	47	42	56	64	58	55	48	49	50
0	1	2	3	4	5	6	7	8	9

Accessing particular elements of an array is easy: `int day5temp = weatherTenDays[4];`

You can similarly **update** array values at any given time: `weatherTenDays[9]=60;`

Note: Difficulty is indicated by number of '*' signs next to the question. There could be multiple correct answers for every question.

1) Expected output? *

```
public void mystery1()
{
        int pokemonGenerationsRanked[] = {4,3,6,2,1,7,5,8};
        System.out.print(pokemonGenerationsRanked[7]);
}
```

a) 4, 3, 6, 2, 1, 7, 5, 8
b) 8
c) Runtime error
d) Compiler error

2) Expected output? **

```
public void mystery2()
{
        int csCourses[] = {121,186,187};
        csCourses[3]=220;
        System.out.print(csCourses[2]+csCourses[3]);
}
```

a) 407
b) 187220
c) Runtime error
d) Compiler error

3) Expected output? **

```
public void mystery3()
{
        int starWars[] = {4, 5, 6, 1, 2, 3, 7, 8, 9};
        double ratings[] = {8.6, 8.7, 8.3, 6.5, 6.5, 7.5, 7.9, 7.0, 6.8};

        for(int i=0; i<starWars.length; i++)
        {
                if(ratings[i]>7.5)
                {
                        System.out.print(starWars[i]);
                }
        }

        System.out.print(“ and Rogue One, obviously.”);
}
```

a) 45637 and Rogue One, obviously.
b) and Rogue One, obviously.
c) 4567 and Rogue One, obviously.
d) Compiler error.

4) Write a method to take an input array and return the sum of its elements. Hint: a for-each loop can be used for this question. *

```
public int sumOfElements(int[] numbers)
{












}
```

Study Notes: Some More Specifics about an Array

5) **Returning** an array:

You can return arrays as if they are any other datatype!

```
public int[] getCSCourses()
{
        int[] csCourses = {121,186,187,220,230,240,250,305,311};
        return csCourses;
}
```

6) Another way to **initialize** an array:

```
int csCourses[] = new int[9];
```

Creates an array with 9 spaces and sets all the values to default values of the datatype. For an int array, all values are initialized to 0. For a String array, all values are initialized to null. For a **boolean** array, all values are initialized to false.

7) Taking an **array as input** from the user:

```
public int[] takeInput()
{
        Scanner sc = new Scanner(System.in);

        System.out.println("Enter the number of elements.");
        int length = sc.nextInt();

        int inputArray[] = new int[length];

        for(int =0; i<inputArray.length; i++)
        {
                System.out.println("Enter next element:");
                inputArray[i]=sc.nextInt();
        }

        return inputArray;
}
```

This is obviously very useful for handling large amounts of data, e.g: salary databases, student grades, dining hall menus, etc.

Note: Difficulty is indicated by number of '*' signs next to the question. There could be multiple correct answers for every question.

5) Write a method to take an input array and return an array with the same elements in reverse order. **

```
public Object[] reverse(Object[] objects)
{







}
```

6) Write a method to take an input array and return an array with alternating elements from the first array, starting from the first element. **

```
public Object[] alternates(Object[] objects)
{







}
```

7) Write a method to take an input array and return an array with alternating elements from the first array, starting from the first element.

```
public String[] alternates(String[] strings)
{







}
```

Study Notes: Some More Specifics about an Array

8) **Taking** an array as a parameter?

This part of the study notes we will do as questions. First, solve this question:

SN1) What is the output produced by the following lines of code?

```
public Class Foo
{
        public void bar(int x)
        {
                x=x+5;
                System.out.print(x);
        }
        public static void main(String args[])
        {
                Foo obj = new Foo();
                int num = 3;
                obj.bar(num);
                System.out.print(num);
        }
}
```

A) 88
B) 83
C) 53
D) 33

Think about the important takeaway here. Remember ints are primitive and are hence 'called by value'.

Now solve this question. Think about what it is different and what difference that will cause in the output. Remember arrays are ‘passed as reference’.

SN2) What is the output produced by the following lines of code?

```
public Class Foo
{
        public void bar(int[] x)
        {
                for(int i=0;i<x.length;i++)
                {
                        x[i]=x[i]+5;
                        System.out.print(x[i]);
                }
        }
        public static void main(String args[])
        {
                Foo obj = new Foo();
                int nums[] = {3,3,3};
                obj.bar(nums);
                for(int i=0;i<nums.length;i++)
                {
                        System.out.print(nums[i]);
                }
        }
}
```

E) 888888
F) 888333
G) 333333
H) 555333

}

8) Write a boolean method to check whether two int arrays have the same elements. **

```
public boolean equals(int[] arr1, int[] arr2)
{



}
```

9) Write a boolean method to check whether two String arrays have the same elements. **

```
public boolean equals(String[] strArr1, String[] strArr2)
{



}
```

10) Write a method to add numToAdd to every element in an array to obtain a new array. ***

```
public int[] add(int[] originalArray, int numToAdd)
{



}
```

11) Write a method to take a String array and return the number of times a String str occurs in the array. **

```
public int getNumOccurences(String[] strArray, String str)
{



}
```

Study Notes: Overloading vs Overriding

9) **Overloading**

Definition: Having two or more methods with the **same name**, but with **different parameters**.

(If they have the same parameters, it will not compile!)

What does **different parameters** entail?

a) Different Number of Parameters,

E.g: void foo(int x) and void foo(int x, int y)

b) Different Datatype of Parameters, or

E.g: void foo(int x) and void foo(boolean x)

c) Different Ordering of Parameters

E.g: void foo(int x, boolean y) and void foo(boolean x, int y)

Remember that variable names are inconsequential. All we care about is the datatype of the variables and the number of variables.

Study Notes: Stuff to think about before next session:

10) **2-Dimensional Array**

Till now we have only seen one-dimensional arrays. We can have multi-dimensional arrays as well.

Example: boolean ticTacToe[][] = new boolean[3][3];

Indexes	0	1	2
0	false	false	false
1	false	false	false
2	false	false	false

Updating 2-Dimensional Array: ticTacToe**[0][1]** = true;

Sets the element in the **0th row** and the **1st column** to true.

Indexes	0	1	2
0	false	true	false
1	false	false	false
2	false	false	false

Traversing a 2-Dimensional Array:

```
for(int i=0; i<arr.length;i++) // Traverses rows
{
        for(int  j=0;j<arr[0].length;j++)//Traverses  columns,  remember  2-D  array  is  array  of
arrays
        {
                System.out.print(ticTacToe[i][j]+" ");
        }
        System.out.println();
}       // Try and predict the output.
```

Resources for learning more about arrays: https://tinyurl.com/DavidMalanArrays