

A New Problem

- Consider the following task:
 - Input N real numbers representing temperature measurements and compute the following:
 - the average, minimum, and maximum temperature
 - the number of temperatures that are above the average and the number of temperatures that are below the average
 - the median temperature

What's the Problem?

- We need to store *all* the temperatures *before* we can perform some of the computation
- How many variables do we need?
- What are we going to name them?

The Solution: Arrays

- An *array* is a sequence of variables of the same data type (any type can be used: e.g., int, double, boolean, String, etc.)
- Each variable in the array is called an *element*
- Array elements are accessed through an *index* (their position in the array)

An Example

```
int N = 5;
double [] temperatures;
temperatures = new double[N];

Scanner in = new Scanner(System.in);
int pos = 0;
while (pos < N)
{
    temperatures[pos] = in.nextDouble();
    pos = pos + 1;
}
```

Declaring Arrays

- Arrays are declared like normal variables with the addition of [] between type and name, e.g.

```
double [] temperatures;  
int [] scores;  
boolean [] answers;  
String [] names;
```

- The declaration does not specify the number of elements
- Declaring an array **does not** reserve (allocate) memory for the array elements

Allocating The Array

- To allocate memory space for the array elements we need to use the **new** operator, e.g.,

```
temperatures = new double[10];  
int numScores = 8;  
scores = new int[numScores];  
answers = new boolean[4];  
names = new String[5];
```

- **new** is followed by the type of the elements and the number of elements between []
- The number of elements can be any expression that evaluates to a positive integer value

Accessing Array Elements

- Elements of an array are accessed by using the name of the array variable followed by the index (position in the array) of the element between [], e.g.,

```
temperatures[2]  
scores[numScores-1]  
answers[0]  
names[3]
```

- The index can range between 0 and the number of elements in the array - 1

Array Length

- Once we declare and allocate an array, we can retrieve the array length (the number of elements in the array) from a variable called *arrayName.length*, e.g.,

```
temperatures.length is 10  
scores.length is 8  
answers.length is 4  
names.length is 5
```

Your Turn

- Declare and allocate an array of 10 integers, call it *a*

Your Turn cont.

- Write a loop that initializes the 10 elements of the array to the integers 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Your Turn cont.

- Write a loop that outputs the elements of the array, one per line

Your Turn cont.

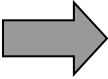
- Write a loop that outputs the elements of the array, one per line, **in reverse order**

Your Turn cont.

- Write a piece of code that checks if the elements in an array of char form a palindrome

For Loops

- A different syntax for a familiar concept
- They work well with arrays

```
for (init; test; update)
{
    for_block
}
    
while (test)
{
    for_block
    update
}
```

Example

- Rewrite the loop that initializes the 10 elements of an array to the integers 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

```
int i = 0;
while (i < 10)
{
    a[i] = i + 1;
    i++;
}
```

Your Turn (Our Old Friend)

- Write a for loop that given a String variable *str* and a character variable *ch*, counts and outputs the number of occurrences of character *ch* in String *str*.

Count Character

Your Turn, Again

- Write a program segment—using for loops—that given two integer variables, *width* and *height*, outputs a rectangle of '+'s of the given width and height.

Output A Rectangle Of '+'s

Arrays And Methods

- Arrays can be passed as parameters to methods and can be returned by methods (functions), e.g.,
 - `private static void printArray(double[] a)`
 - `private static void clearArray(int[] a)`
 - `private static int[] copyArray(int[] a)`
 - `private static int indexOfSmallest(
int startIndex, double[] t)`
 - `private static void sort(double[] temps)`

Print Array

- Given an array of double, output each element on a separate line (in the order in which they occur in the array)

```
private static void printArray(double[] a)
{

}
}
```

Clear Array

- Given an array of int, set each element of the array to 0

```
private static void clearArray(int[] a)
{

}
}
```

Copy Array

- Given an array of int, return a new array which is a copy of the given one (same length, same elements)

```
private static int[] copyArray(int[] a)
{

}
}
```