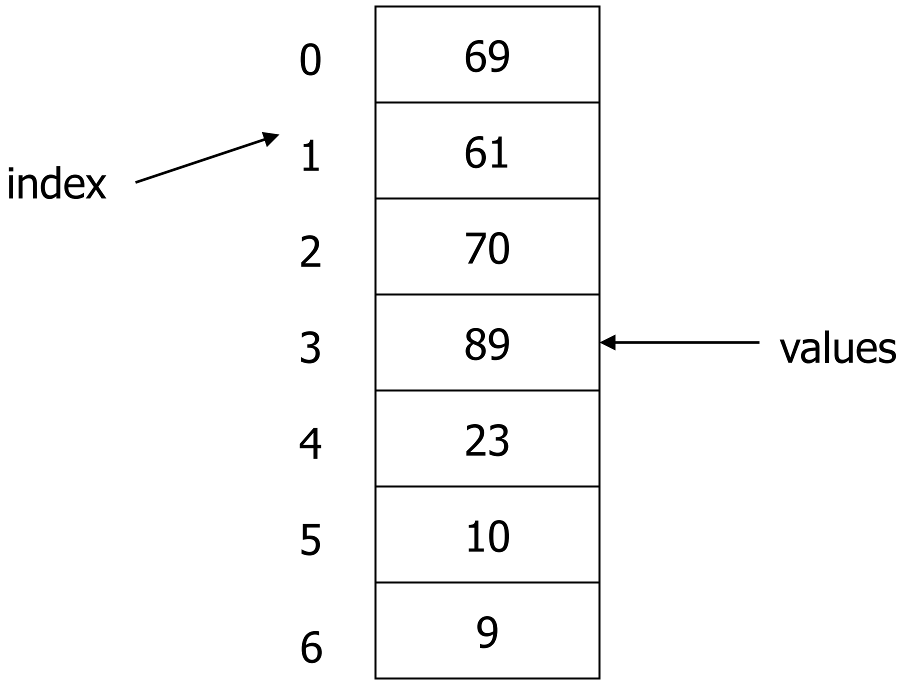
Arrays, Strings

Arrays - Introduction

- An array is a group of contiguous or related data items that share a common name.
- Used when programs have to handle large amount of data
- Each value is stored at a specific position
- Position is called a index or superscript. Base index = 0
- The ability to use a single name to represent a collection of items and refer to an item by specifying the item number enables us to develop concise and efficient programs. For example, a loop with index as the control variable can be used to read the entire array, perform calculations, and print out the results.

Arrays - Introduction



Declaration of Arrays

- Like any other variables, arrays must declared and created before they can be used. Creation of arrays involve three steps:
  - Declare the array
  - Create storage area in primary memory.
  - Put values into the array (i.e., Memory location)
- Declaration of Arrays:
  - Form 1:

    Type arrayname[]
  - Form 2:
    - Type [] arrayname;

  - Examples:

    ```
    int[] students;
    int students[];
    ```
  - Note: we don't specify the size of arrays in the declaration.

Creation of Arrays

- After declaring arrays, we need to allocate memory for storage array items.
- In Java, this is carried out by using “new” operator, as follows:
  - Arrayname = **new** type[size];
- Examples:
  - students = new int[7];

Initialisation of Arrays

- Once arrays are created, they need to be initialised with some values before access their content. A general form of initialisation is:
  - Arrayname [index/subscript] = value;
- Example:
  - students[0] = 50;
  - students[1] = 40;
- Like C, Java creates arrays starting with subscript 0 and ends with value one less than the size specified.
- Unlike C, Java protects arrays from overruns and under runs. Trying to access an array beyond its boundaries will generate an error message.

Arrays – Length

- Arrays are fixed length
- Length is specified at create time
- In java, all arrays store the allocated size in a variable named “length”.
- We can access the length of arrays as arrayName.length:

```
e.g.  int x = students.length;     //  x = 7
```

- Accessed using the index

```
e.g.   int x = students [1];          //  x = 40
```

Arrays – Example

```
// StudentArray.java: store integers in arrays and access
public class StudentArray{
        public static void main(String[] args) {
                int[] students;
                students = new int[7];
                System.out.println("Array Length = " + students.length);

                for ( int  i=0;  i < students.length;  i++)
                        students[i] = 2*i;
                System.out.println("Values Stored in Array:");
                for ( int  i=0;  i < students.length;  i++)
                        System.out.println(students[i]);
        }
}
```

:w

Arrays – Initializing at Declaration

- Arrays can also be initialised like standard variables at the time of their declaration.
  - Type arrayname[] = {list of values};
- Example:

```
int[] students = {55, 69, 70, 30, 80};
```

- Creates and initializes the array of integers of length 5.
- In this case it is not necessary to use the *new* operator.

Arrays – Example

```
// StudentArray.java: store integers in arrays and access
public class StudentArray{
        public static void main(String[] args) {
        int[] students = {55, 69, 70, 30, 80};

        System.out.println("Array Length = " + students.length);
        System.out.println("Values Stored in Array:");
         for ( int  i=0;  i < students.length;  i++)
                System.out.println(students[i]);
        }
}
```

Two Dimensional Arrays

- Two dimensional arrays allows us to store data that are recorded in table. For example:
- Table contains 12 items, we can think of this as a matrix consisting of 4 rows and 3 columns.

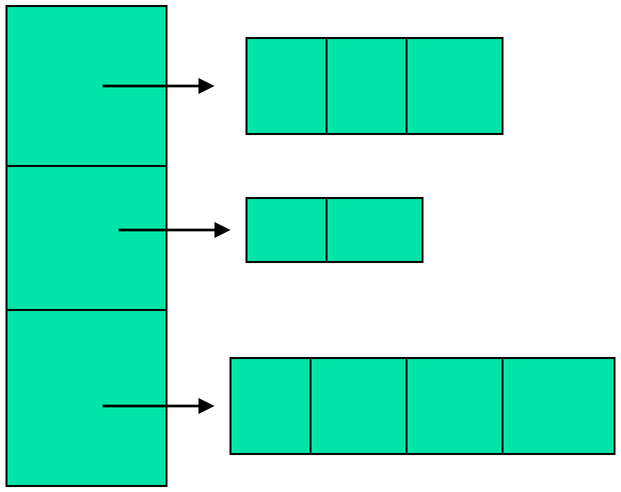
Sold / Person	Item1	Item2	Item3
Salesgirl #1	10	15	30
Salesgirl #2	14	30	33
Salesgirl #3	200	32	1
Salesgirl #4	10	200	4

2D arrays manipulations

- Declaration:
  - int myArray [][];
- Creation:
  - myArray = new int[4][3]; // OR
  - int myArray [][] = new int[4][3];
- Initialisation:
  - Single Value;
    - myArray[0][0] = 10;
  - Multiple values:
    - int tableA[2][3] = {{10, 15, 30}, {14, 30, 33}};
    - int tableA[][] = {{10, 15, 30}, {14, 30, 33}};

Variable Size Arrays

- Java treats multidimensional arrays as “arrays of arrays”. It is possible to declare a 2D arrays as follows:
  - int a[][] = new int [3][];
  - a[0]= new int [3];
  - a[1]= new int [2];
  - a[2]= new int [4];



Try: Write a program to Add to Matrix

- Define 2 dimensional matrix variables:
  - Say: int a[][], b[][];
- Define their size to be 2x3
- Initialise like some values
- Create a matrix c to storage sum value
  - c[0][0] = a[0][0] + b[0][0]
- Print the contents of result matrix.

Arrays of Objects

- Arrays can be used to store objects

```
Circle[] circleArray;
circleArray = new Circle[25];
```

- The above statement creates an array that can store references to 25 Circle objects.
- Circle objects are not created.

Arrays of Objects

- Create the Circle objects and stores them in the array.
  - //declare an array for Circle

```
    Circle circleArray[] = new Circle[25];
    int r = 0;
// create circle objects and store in array
    for (r=0;  r <25; r++)
      circleArray[r] = new Circle(r);
```

String Operations in Java

Introduction

- String manipulation is the most common operation performed in Java programs. The easiest way to represent a String (a sequence of characters) is by using an array of characters.
    - Example:
    - char place[] = new char[4];
    - place[0] = 'J';
    - place[1] = 'a';
    - place[2] = 'v';
    - place[3] = 'a';
- Although character arrays have the advantage of being able to query their length, they themselves are too primitive and don't support a range of common string operations. For example, copying a string, searching for specific pattern etc.
- Recognising the importance and common usage of String manipulation in large software projects, Java supports String as one of the fundamental data type at the language level. Strings related book keeping operations (e.g., end of string) are handled automatically.

String Operations in Java

- Following are some useful classes that Java provides for String operations.
  - String Class
  - StringBuffer Class
  - StringTokenizer Class

String Class

- String class provides many operations for manipulating strings.
  - Constructors
  - Utility
  - Comparisons
  - Conversions
- String objects are read-only (immutable)

Strings Basics

- Declaration and Creation:
  - **String** stringName;
  - stringName = **new String** (“string value”);
  - Example:
    - String city;
    - city = new String (“Bangalore”);
  - Length of string can be accessed by invoking length() method defined in String class:
    - int len = city.length();

String operations and Arrays

- Java Strings can be concatenated using the + operator.
  - String city = “New” + “York”;
  - String city1 = “Delhi”;
  - String city2 = “New “+city1;
- Strings Arrays
  - String city[] = new String[5];
  - city[0] = new String(“Melbourne”);
  - city[1] = new String(“Sydney”);
  - …
  - String megacities[] = {“Brisbane”, “Sydney”, “Melbourne”, “Adelaide”, “Perth”};

String class - Constructors

public String()	Constructs an empty String.
Public String(String value)	Constructs a new string copying the specified string.

String – Some useful operations

public int length()	Returns the length of the string.
public charAt(int index)	Returns the character at the specified location (*index)*
public int compareTo(String anotherString) public int compareToIgnoreCase(String anotherString)	Compare the Strings.
reigonMatch(int start, String other, int ostart, int count)	Compares a region of the Strings with the specified start.

String – Some useful operations

public String replace(char oldChar, char newChar)	Returns a new string with all instances of the *oldChar* replaced with *newChar.*
public trim()	Trims leading and trailing white spaces.
public String toLowerCase() public String toUpperCase()	Changes as specified.

String Class - example

```
// StringDemo.java: some operations on strings
class StringDemo {
        public static void main(String[] args)
        {
                String s = new String("Have a nice Day");

                // String Length =  15
                System.out.println("String Length = " + s.length() );

                // Modified String =  Have a Good Day
                System.out.println("Modified String = " + s.replace('n', 'N'));

                // Converted to Uppercse =  HAVE A NICE DAY"
                System.out.println("Converted to Uppercase = " + s.toUpperCase());

                // Converted to Lowercase =  have a nice day"
                System.out.println("Converted to Lowercase = " + s.toLowerCase());

        }
}
```

StringDemo Output

```
[raj@mundroo] Arrays [1:130] java StringDemo

String Length = 15
Modified String = Have a Nice Day
Converted to Uppercase = HAVE A NICE DAY
Converted to Lowercase = have a nice day

[raj@mundroo] Arrays [1:131]
```

Summary

- Arrays allows grouping of sequence of related items.
- Java supports powerful features for declaring, creating, and manipulating arrays in efficient ways.
- Each items of arrays of arrays can have same or variable size.
- Java provides enhanced support for manipulating strings and manipulating them appears similar to manipulating standard data type variables.