Faculty of Engineering & Information Technology

Data Structures & Algorithms

Introduction

Asst. Prof. Dr. Ahmed A.O. Tayeh

Course Organisation

- Theory Course - 3 Credits (EITM2311)
  - Asst. Prof. Dr. Ahmed A.O. Tayeh, atayeh@israa.edu.ps

- Practicum - 1 Credit (EITM2112)
  - Hadeel Altalli, haltali@israa.edu.ps

Contact Details

- If nothing urgent, please contact me via emails
  - atayeh@israa.edu.ps
  - expect a reply in 24 hours
- Office Hours
  - Sunday 8 – 11 AM
  - Sunday 1 – 3 PM
  - Monday 8 – 10 AM
- Do not hesitate to discuss things during lecture breaks
- If questions are related to course topics, raise them during the lectures so others can learn

Grading

- Theory Course
  - midterm exam 30%
  - exercises, presence & project 20%
  - final exam 50%
- Practicum
  - will be shared later

Prerequisites

- Programming I (EITM1302 & EITM1103)
- Programming II (EITM1307 & EITM1108)
- Java fundamentals
- OOP fundamentals

Course Objectives

- Understand basic data structures and algorithms

- Solve problems with the right algorithm
  - use the right data structure
  - design a solution (algorithm)
    - maximum efficiency
    - less memory

"Get your data structures correct first, and the rest of the program will write itself" David Jones

Course Objectives…

- Use data structures in complex real-world problems
  - ***Shipping Port***: containers, ships, vans, employees, storage, transfer of containers, customs, etc..

Source :” https://www.reuters.com/world/china/chinese-ports-choke-over-zero-tolerance-covid-19-policy-2021-08-17”

Course Objectives…

- Use data structures in complex real-world problems
  - ***Smart Hospital***: departments, paths between departments, doctors, patients, emergency triage, operations, medications

Source :" https://www.expresshealthcare.in/covid19-updates/how-covid-19-is-transforming-hospital-design/422712"

Course Objectives…

- Practise & implement data structures & algorithms
  - implement them yourself
  - go beyond the examples you take in this course
  - write code, write code, write code

Practicum: Exercises

- Course topics is further investigated in exercise sessions

- Weekly exercise sessions
  - Assistant: Hadeel Altalli

- Additional content may be covered in exercise sessions
  - strongly recommended to attend all exercise sessions!
  - exam covers content of lectures and exercises

Practicum: Guidelines

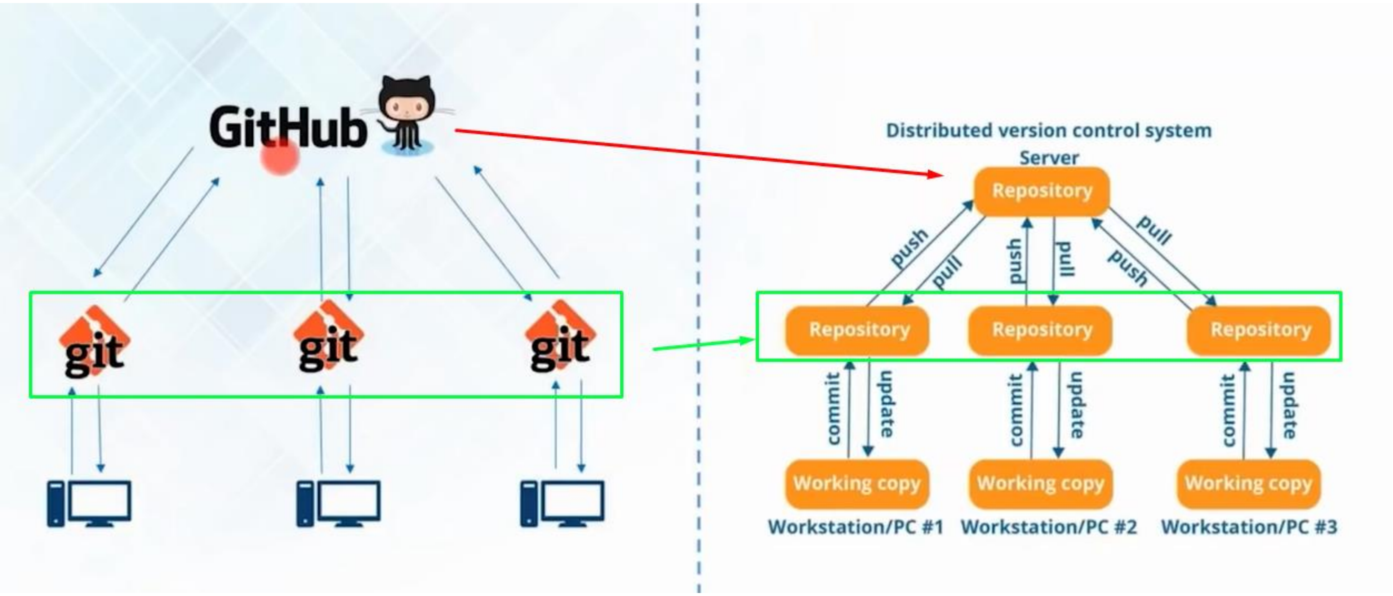
- Examples and exercises are given in Java

- Use any IDE you prefer
  - Eclipse
  - NetBeans
  - IntelliJ IDEA

- You can use also online compilers
  - https://www.jdoodle.com/online-java-compiler/

Practicum: Guidelines…

- GitHub is used for the course content and exercises
  - you must create a GitHub account
  - you need to share your code with us via your GitHub account



Source: https://stackoverflow.com/questions/13321556/difference-between-git-and-github

Study Material

- Slides are the main study material
- Lecture discussions, examples and notes should be considered
- Reference Book "*Data Structures & Algorithms*" *6th edition by Michael T. Goodrich, Roberto Tamassia and Michael H. Goldwasser*
  - anything that is not covered during the lectures, are not part of the exam
- Study material uploaded before each lecture at the study portal and the course repository GitHub account
  - https://github.com/atayeh-israa-university/dataStructures-2023

Course Topics

- Fundamental Data Structures
- Stacks, Queues and Double-Ended Queues
- List and Iterator Abstract Data Structures
- Algorithm Analysis
- Recursion
- Trees
- Priority Queues
- Maps, Hash Table and Skip Lists
- Search Trees
- Sorting and Selection

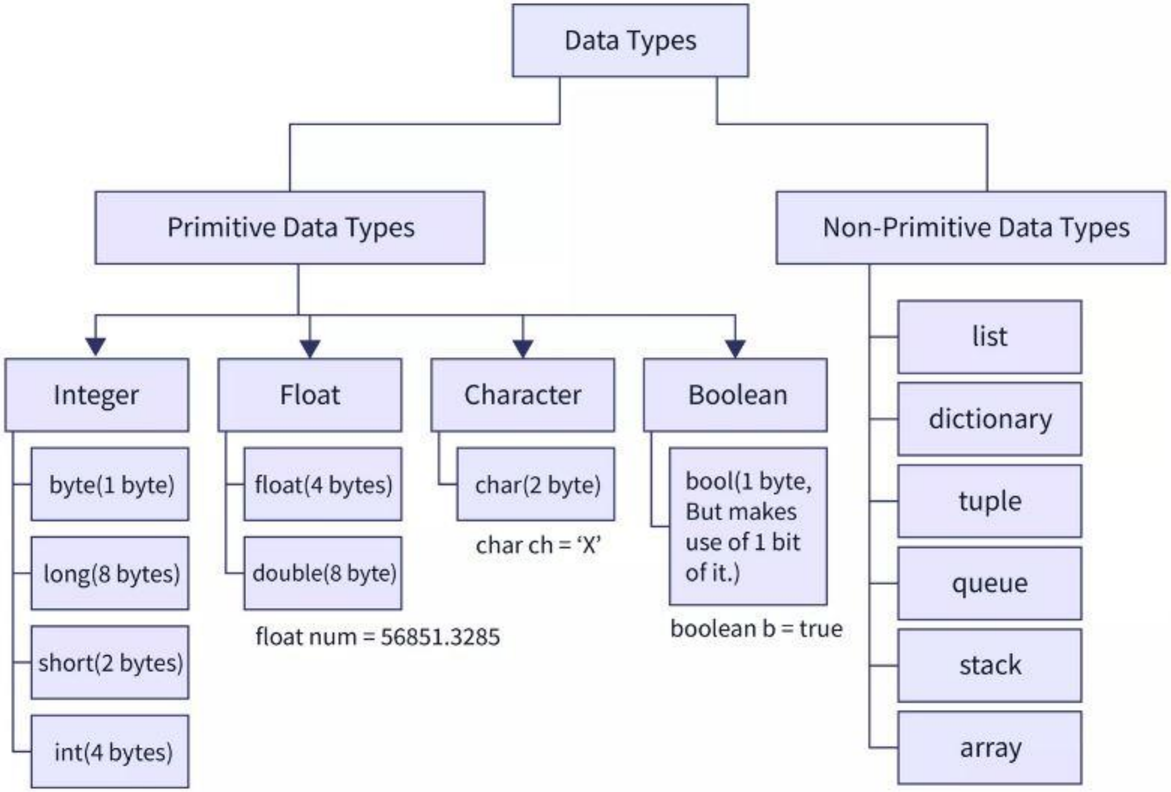
Fundamental Data Structures

Data Structures

- Abstract way for organising data in computer memory so it can be used efficiently

- Data can be organised in many ways
  - some data standard data structures have proved useful in many cases
  - “*one size fits all*” data structure does not exist
  - choose a data structure based on your problem needs and operations

Data Structures…



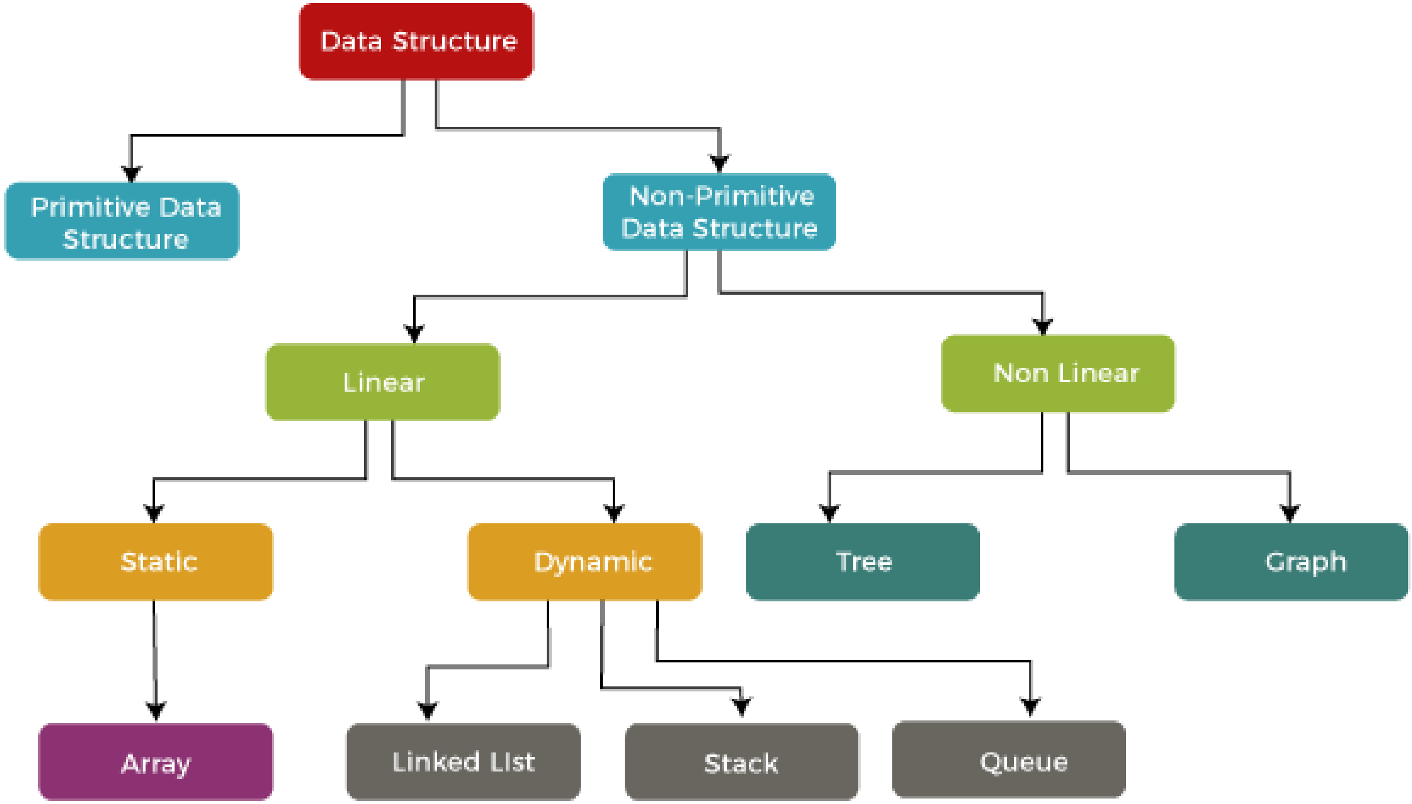
Source: "https://www.scaler.com/topics/primitive-data-structure/"

Data Structures…

- **Primitive Data Structure**
  - allows storing single data type value
  - always contains some value
  - do now allow NULL values
  - size depends on type of the structure
  - Integer, Boolean, character, float, etc
  - common operations
    - Creation
    - Selection
    - Updating
    - Destroy or Delete

- **Non-Primitive Data Structure**
  - stores multiple data type values
    - homogeneous (same type)
    - heterogeneous (different type)
  - can store NULL values in Primitive data structures
  - size is not fixed
  - Array, Linked List, Stack, Queue
  - common operations
    - Insertion
    - Selection
    - Searching
    - Sorting
    - Merging
    - Destroy or Delete

Data Structures…



Source:” https://www.javatpoint.com/primitive-vs-non-primitive-data-structure”

Data Structures…

- **Linear Data Structures**
  - homogeneous elements
  - sequences and liner series
  - easy to implement
    - memory is organised in a linear fashion
  - Array, Stack, Queue & Linked Lists

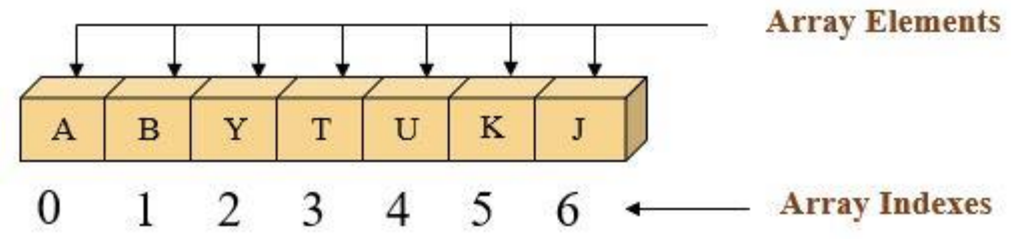
- **Non-Linear Data Structures**
  - data items are connected to several other data items
  - hierarchical relationship or parent child relationship
  - not arranged in a sequential structure
  - Trees and Graphs

Data Structures…

- **Abstract Data Types (ADTs)**
  - conceptual model of information structure
  - specifies the components, their structuring relationships
  - specifies a list of operations (behaviour) that are allowed to be performed
  - just specification based on how they are used, no design or implementation details is included
  - cannot analyse the time and memory complexity of an ADT!
- **Data Structure**
  - a concrete implementation of a data type (ADT)
  - possible to analyse the time and memory complexity
  - can be implemented in several ways and implementation may vary from language to another

Data Structures…

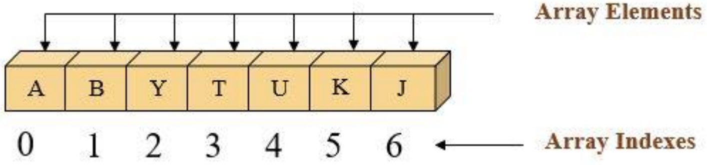
- Examples of ADTs
  - Array
  - Queue
  - Stack



- Array – ADT
  - holds collection of elements
  - implemented as an Array Data Structure
    - Java: *String [] students = {"Hassan", "Anas", "Khadija"};*
    - PHP: *$cars = array("Volvo", "BMW", "Toyota");*
    - Dart: *List<int> numbers = [1, 2, 3, 4, 5];*
  - elements accessible by index
    - *Java: System.out.println(students[1]);   // Output Anas*
    - *PHP: echo $cars[0];   // Output Volvo*
    - *Dart: print(numbers[2]);   // Output: 3*

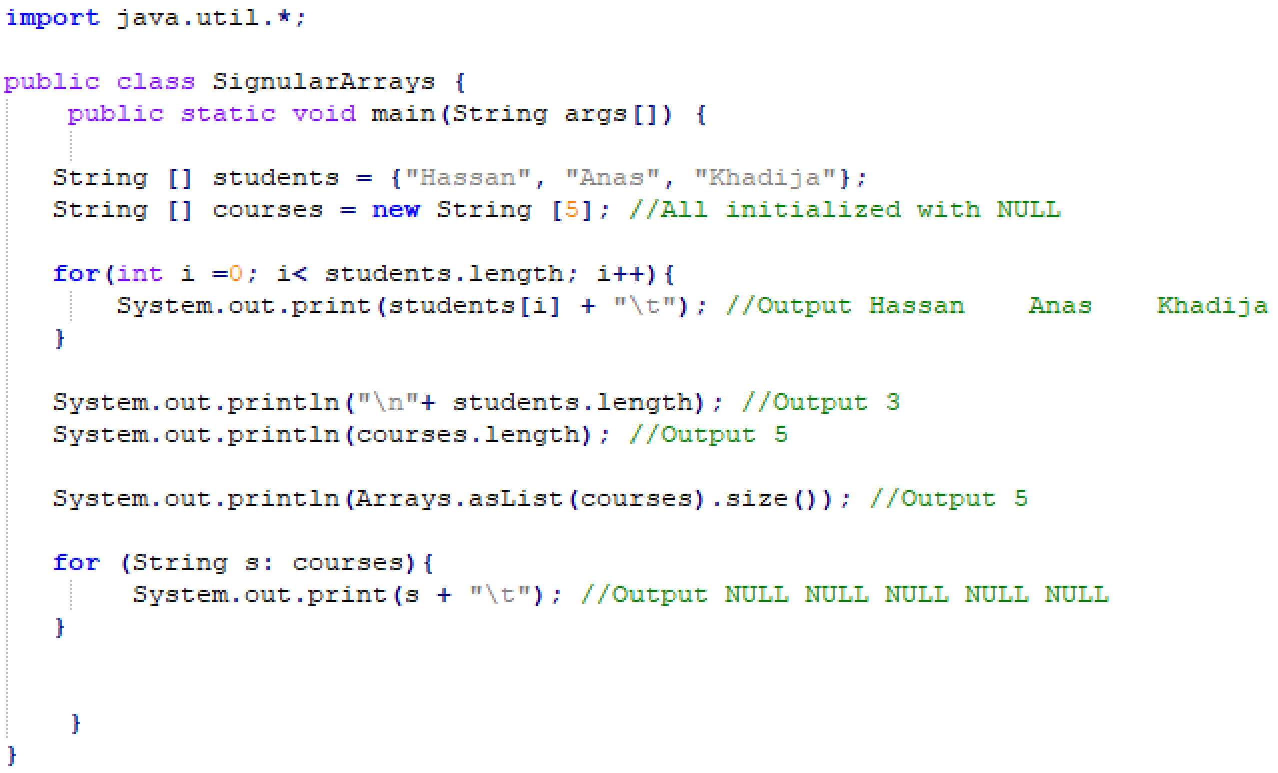
Arrays

- A sequenced collection of variables all of the same type
  - all integer, all float-point, etc
- Stores related data
  - students
  - university courses



- Has fixed size
  - gets a fixed size at initialisation time
  - should consider other data structure when dynamicity is needed
- Stored in successive memory locations
- *Array length*: maximum capacity
- *Array size*: actual number of stored elements

Arrays…

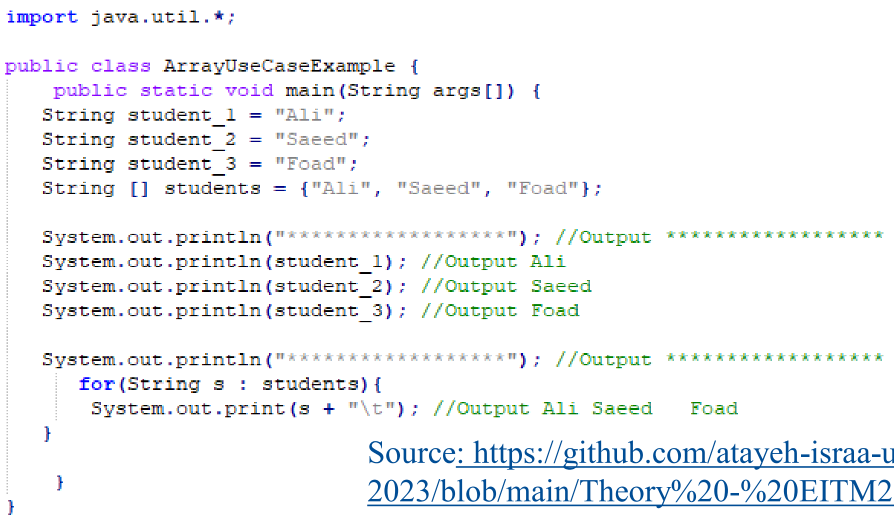


```
import java.util.*;

public class SignularArrays {
    public static void main(String args[]) {

   String [] students = {"Hassan", "Anas", "Khadija"};
   String [] courses = new String [5]; //All initialized with NULL

   for(int i =0; i< students.length; i++){
       System.out.print(students[i] + "\t"); //Output Hassan    Anas    Khadija
   }

   System.out.println("\n"+ students.length); //Output 3
   System.out.println(courses.length); //Output 5

   System.out.println(Arrays.asList(courses).size()); //Output 5

   for (String s: courses){
        System.out.print(s + "\t"); //Output NULL NULL NULL NULL NULL
   }


    }
}
```

https://github.com/atayeh-israa-university/dataStructures/blob/main/Theory%20-%20EITM2311/Arrays/SignularArrays.java

Arrays…

- Use Case: We need to store THREE students



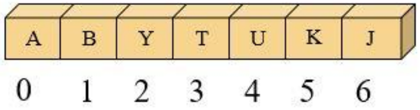
```java
import java.util.*;

public class ArrayUseCaseExample {
     public static void main(String args[]) {
    String student_1 = "Ali";
    String student_2 = "Saeed";
    String student_3 = "Foad";
    String [] students = {"Ali", "Saeed", "Foad"};

    System.out.println("*******************"); //Output *******************
    System.out.println(student_1); //Output Ali
    System.out.println(student_2); //Output Saeed
    System.out.println(student_3); //Output Foad

    System.out.println("*******************"); //Output *******************
       for(String s : students){
        System.out.print(s + "\t"); //Output Ali Saeed   Foad
    }

      }
}
```

Source: https://github.com/atayeh-israa-university/dataStructures-2023/blob/main/Theory%20-%20EITM2311/Arrays/ArrayUseCaseExample.java

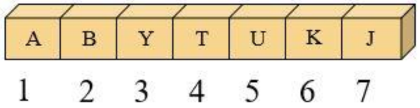
- Imagine a THOUSAND students?
  - how to iterate and do a single or more operation for all students?
  - code quality and maintainability!

Arrays…

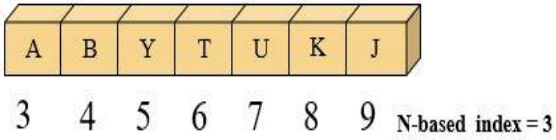
- Zero-based indexing
  - first (base) element is indexed by subscript (position) of 0
  - Java and PHP
  - Array [0] = “A”



- One-based indexing
  - first (base) element is indexed by subscript (position) of 1
  - Fortran and COBOL
  - Array [1] = “A”



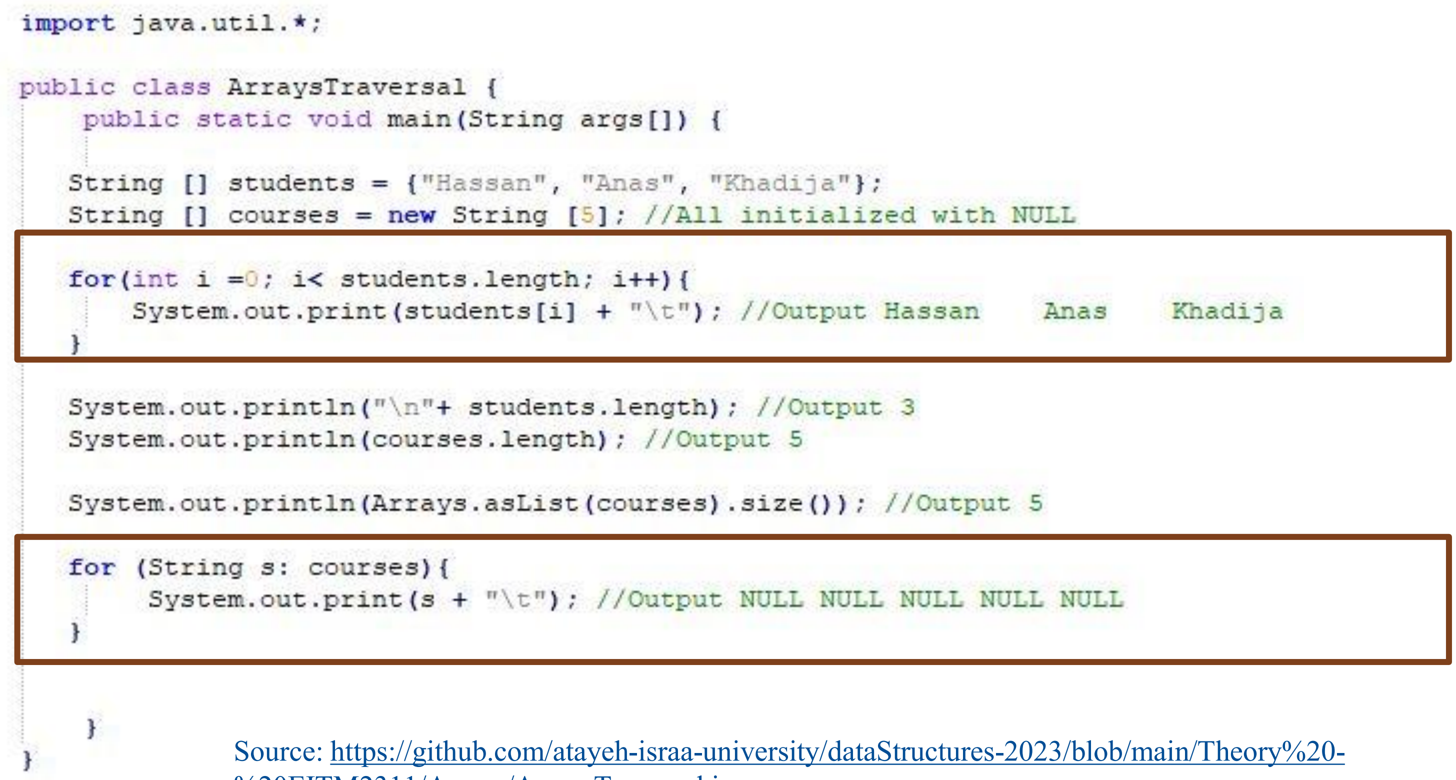
- N-based indexing
  - first (base) index can be freely chosen
  - Ada programming language
  - Array [3] = “A”



Array Operations

- **Traversal:** traverse all the elements one after another
- **Insertion:** add an element at a given position
- **Deletion:** delete an element at a given position
- **Searching:** search an element using a given index or value
- **Updating:** update an element at a given index
- **Sorting:** arrange elements in the array in a specific order
  - *try to create Java code to sort an array of integers*
- **Merging:** merge two arrays into one
  - *try to create Java code to merge two arrays*

Array Operations: Traversal



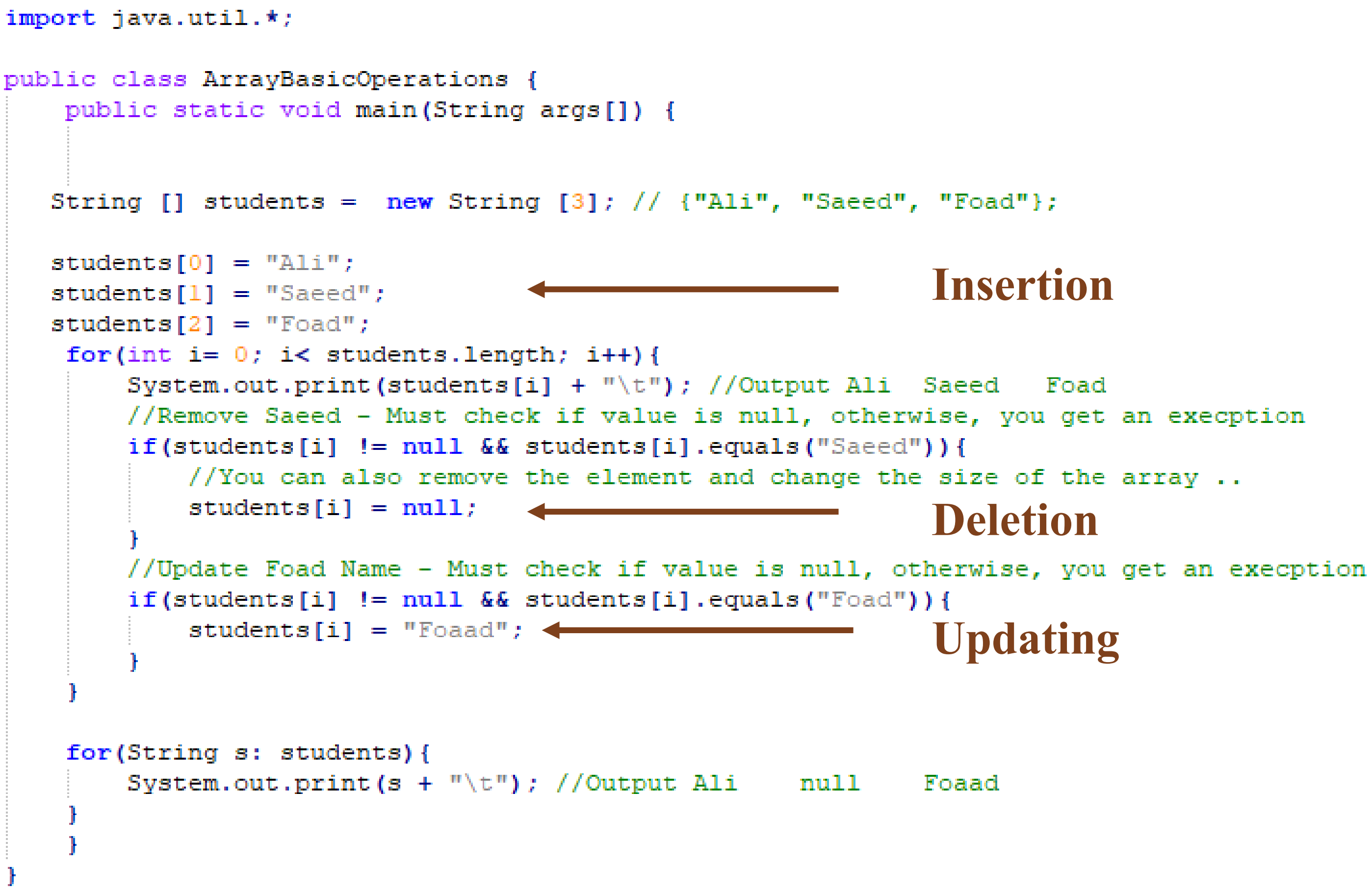
```
import java.util.*;

public class ArraysTraversal {
     public static void main(String args[]) {

   String [] students = {"Hassan", "Anas", "Khadija"};
   String [] courses = new String [5]; //All initialized with NULL

   for(int i =0; i< students.length; i++){
       System.out.print(students[i] + "\t"); //Output Hassan    Anas    Khadija
   }

   System.out.println("\n"+ students.length); //Output 3
   System.out.println(courses.length); //Output 5

   System.out.println(Arrays.asList(courses).size()); //Output 5

   for (String s: courses){
        System.out.print(s + "\t"); //Output NULL NULL NULL NULL NULL
   }


    }
}
```

for Loop

for-each Loop

Source: https://github.com/atayeh-israa-university/dataStructures-2023/blob/main/Theory%20-%20EITM2311/Arrays/ArraysTraversal.java

Array Operations: Insertion, Deletion, Updating



```
import java.util.*;

public class ArrayBasicOperations {
    public static void main(String args[]) {


   String [] students =  new String [3]; // {"Ali", "Saeed", "Foad"};

   students[0] = "Ali";
   students[1] = "Saeed";
   students[2] = "Foad";
    for(int i= 0; i< students.length; i++){
        System.out.print(students[i] + "\t"); //Output Ali   Saeed    Foad
        //Remove Saeed - Must check if value is null, otherwise, you get an execption
        if(students[i] != null && students[i].equals("Saeed")){
            //You can also remove the element and change the size of the array ..
            students[i] = null;
        }
        //Update Foad Name - Must check if value is null, otherwise, you get an execption
        if(students[i] != null && students[i].equals("Foad")){
            students[i] = "Foaad";
        }
    }

    for(String s: students){
        System.out.print(s + "\t"); //Output Ali    null    Foaad
    }
    }
}
```

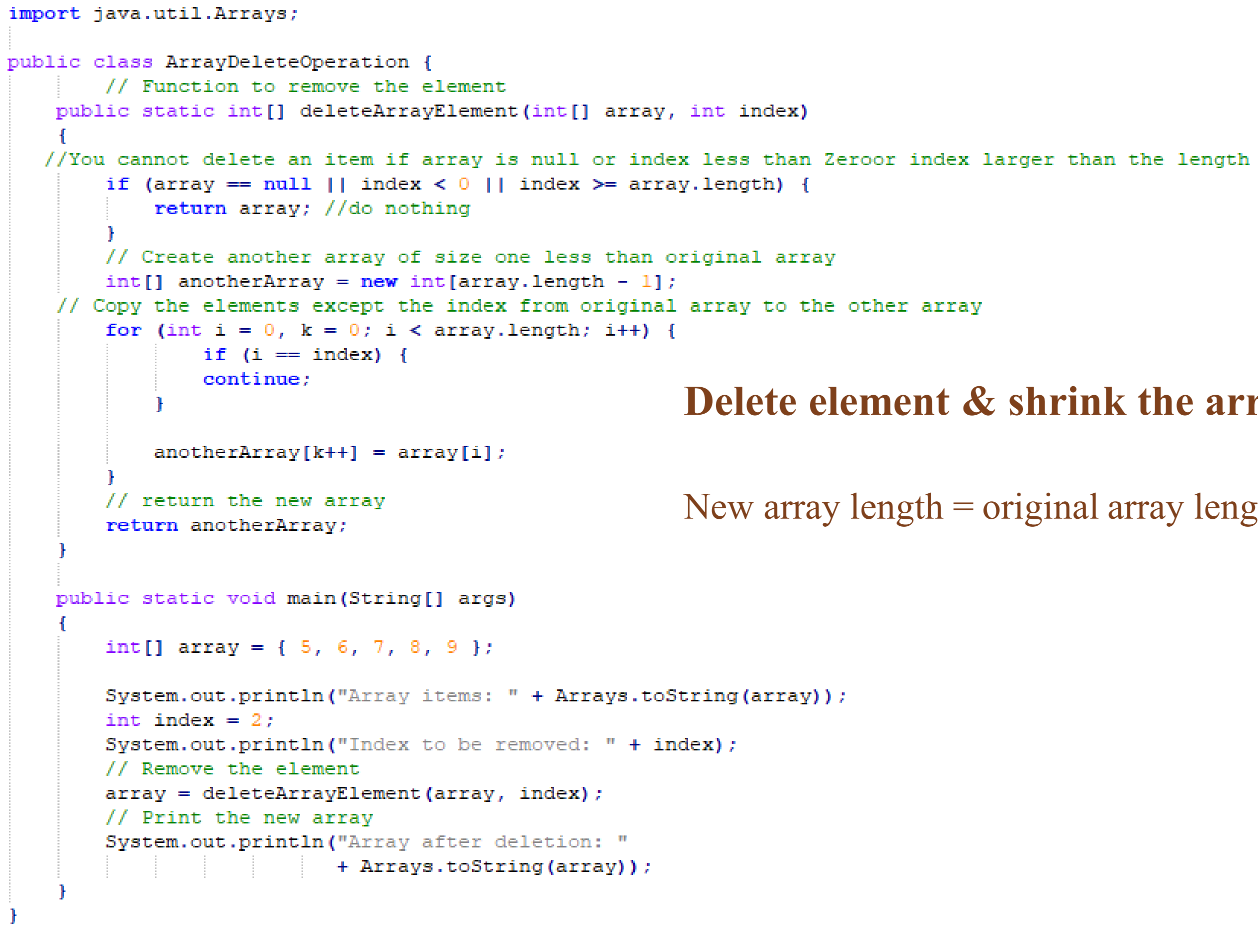
Insertion

Deletion

Updating

Source: https://github.com/atayeh-israa-university/dataStructures-2023/blob/main/Theory%20-%20EITM2311/Arrays/ArrayBasicOperations.java

Array Operations: Insertion, Deletion, Updating



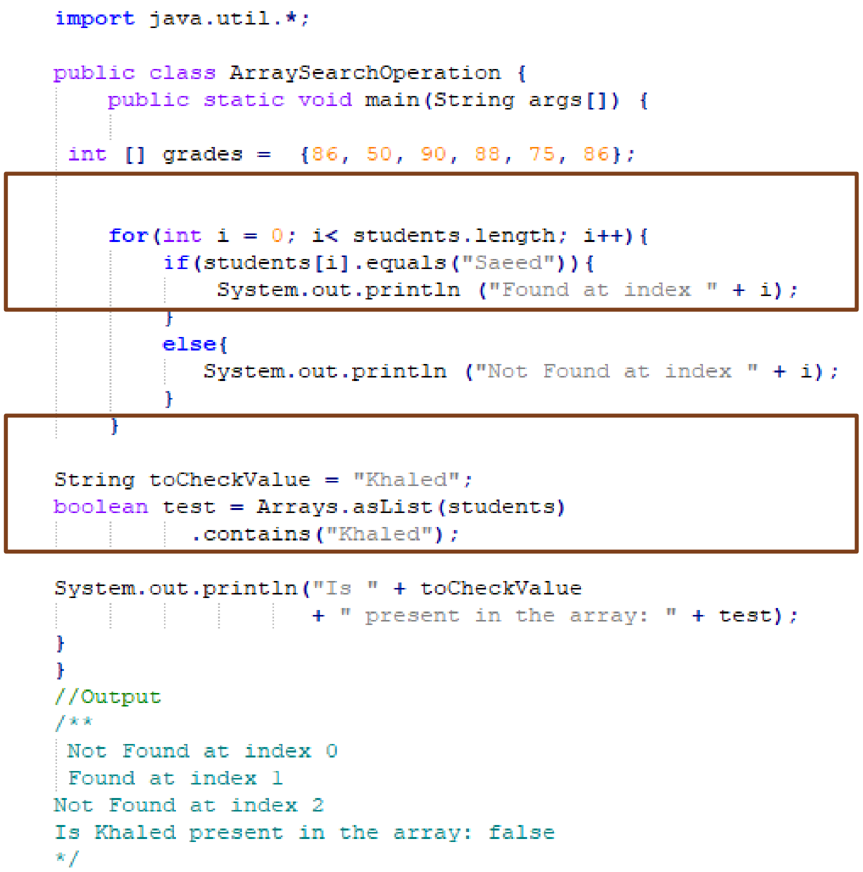
```
import java.util.Arrays;

public class ArrayDeleteOperation {
        // Function to remove the element
    public static int[] deleteArrayElement(int[] array, int index)
    {
   //You cannot delete an item if array is null or index less than Zeroor index larger than the length
        if (array == null || index < 0 || index >= array.length) {
            return array; //do nothing
        }
        // Create another array of size one less than original array
        int[] anotherArray = new int[array.length - 1];
    // Copy the elements except the index from original array to the other array
        for (int i = 0, k = 0; i < array.length; i++) {
                if (i == index) {
                continue;
            }

            anotherArray[k++] = array[i];
        }
        // return the new array
        return anotherArray;
    }

    public static void main(String[] args)
    {
        int[] array = { 5, 6, 7, 8, 9 };

        System.out.println("Array items: " + Arrays.toString(array));
        int index = 2;
        System.out.println("Index to be removed: " + index);
        // Remove the element
        array = deleteArrayElement(array, index);
        // Print the new array
        System.out.println("Array after deletion: "
                                   + Arrays.toString(array));
    }
}
```

Delete element & shrink the array size

New array length = original array length - 1

Source: https://github.com/atayeh-israa-university/dataStructures-2023/blob/main/Theory%20-%20EITM2311/Arrays/ArrayDeleteOperation.java

Array Operations: Searching



```
import java.util.*;

public class ArraySearchOperation {
    public static void main(String args[]) {

 int [] grades =  {86, 50, 90, 88, 75, 86};


    for(int i = 0; i< students.length; i++){
        if(students[i].equals("Saeed")){
            System.out.println ("Found at index " + i);
        }
        else{
           System.out.println ("Not Found at index " + i);
        }
    }

String toCheckValue = "Khaled";
boolean test = Arrays.asList(students)
                .contains("Khaled");

System.out.println("Is " + toCheckValue
                  + " present in the array: " + test);
}
}
//Output
/**
 Not Found at index 0
 Found at index 1
Not Found at index 2
Is Khaled present in the array: false
*/
```

Search elements in various ways

Benefit from List utility functions

Search via Streams (out of scope)

Time complexity & Performance for each search algorithm!

(To be discussed later)

Thank You!