# ASSIGNMENT 1 FRONT SHEET

| | | | |
|---|---|---|---|
| **Qualification** | **BTEC Level 5 HND Diploma in Computing** | | |
| **Unit number and title** | | | |
| **Submission date** | | **Date Received 1st submission** | |
| **Re-submission Date** | | **Date Received 2nd submission** | |
| **Student Name** | NGUYEN HUU HOANG | **Student ID** | BDAF190022 |
| **Class** | BH-AF-2005-2.3 | **Assessor name** | NGO THI MAI LOAN |

**Student declaration**

I certify that the assignment submission is entirely my own work and I fully understand the consequences of plagiarism. I understand that making a false declaration is a form of malpractice.

| | |
|---|---|
| **Student's signature** | [illegible] |

**Grading grid**

| P1 | P2 | P3 | M1 | M2 | M3 | D1 | D2 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

# ASSIGNMENT 1

**Class**: BH-AF-2005-2.3
**Lecturers**: NGO THI MAI LOAN

[illegible]

facebook.com/hoang.nguyenhuu.7330763/

036 8716 708

hoangnhbdaf190022@fpt.edu.vn

**PRESENTER: NGUYEN HUU HOANG**

Pearson BTEC

## Contents

## P1 - Create A Design Specification For Data Structures?

### 1. WHAT IS DATA ABSTRACTION?

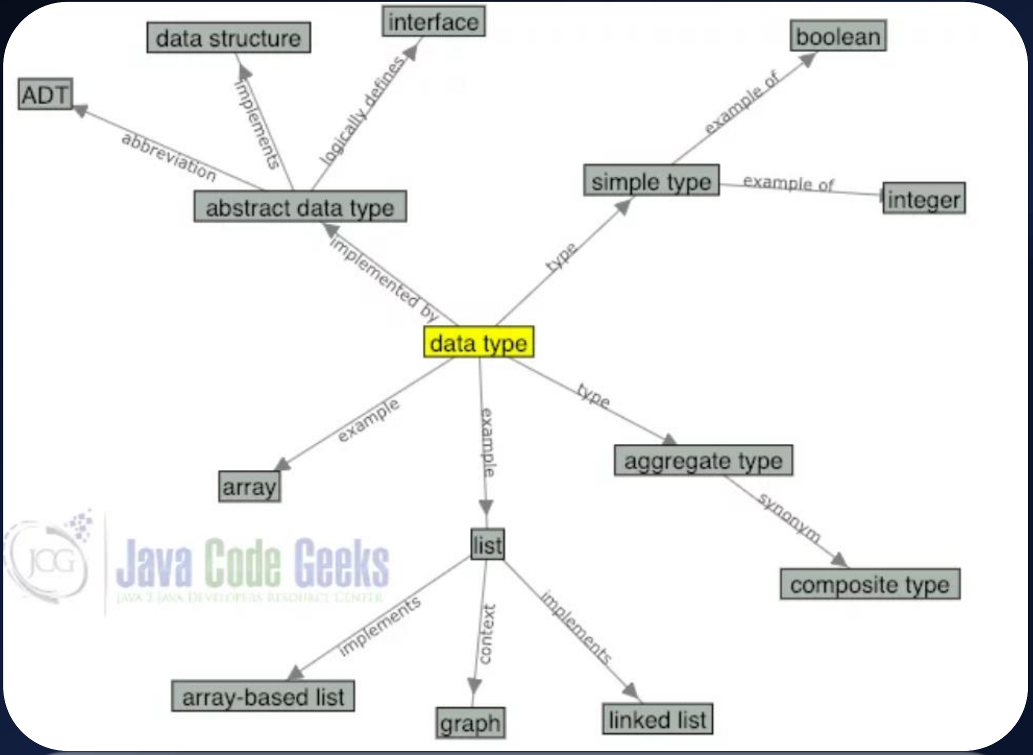
**Data abstraction** is the reduction of a particular body of data to a simplified representation of the whole. Abstraction, in general, is the process of taking away or removing characteristics from something in order to reduce it to a set of essential characteristics.

**For example,** method abstraction in OOP like C++ can use (pre-defined) methods without concern about how they work inside.

## 2. WHAT IS DATA ABSTRACTION TYPE (ADT)?

An abstract data type (ADT) is a specification of a data type in some programming languages, independent of the implementation. The interface for the ADT is defined by the type and set of operations on that type. The behavior of each operation is determined by its inputs and outputs.

ADT is not exactly as defined as how the data type is implemented. These implementation details are hidden from ADT users and protected from outside access - Encapsulation.



## 3. BENEFITS

**Manufacturer (who creates ADT) benefits:**

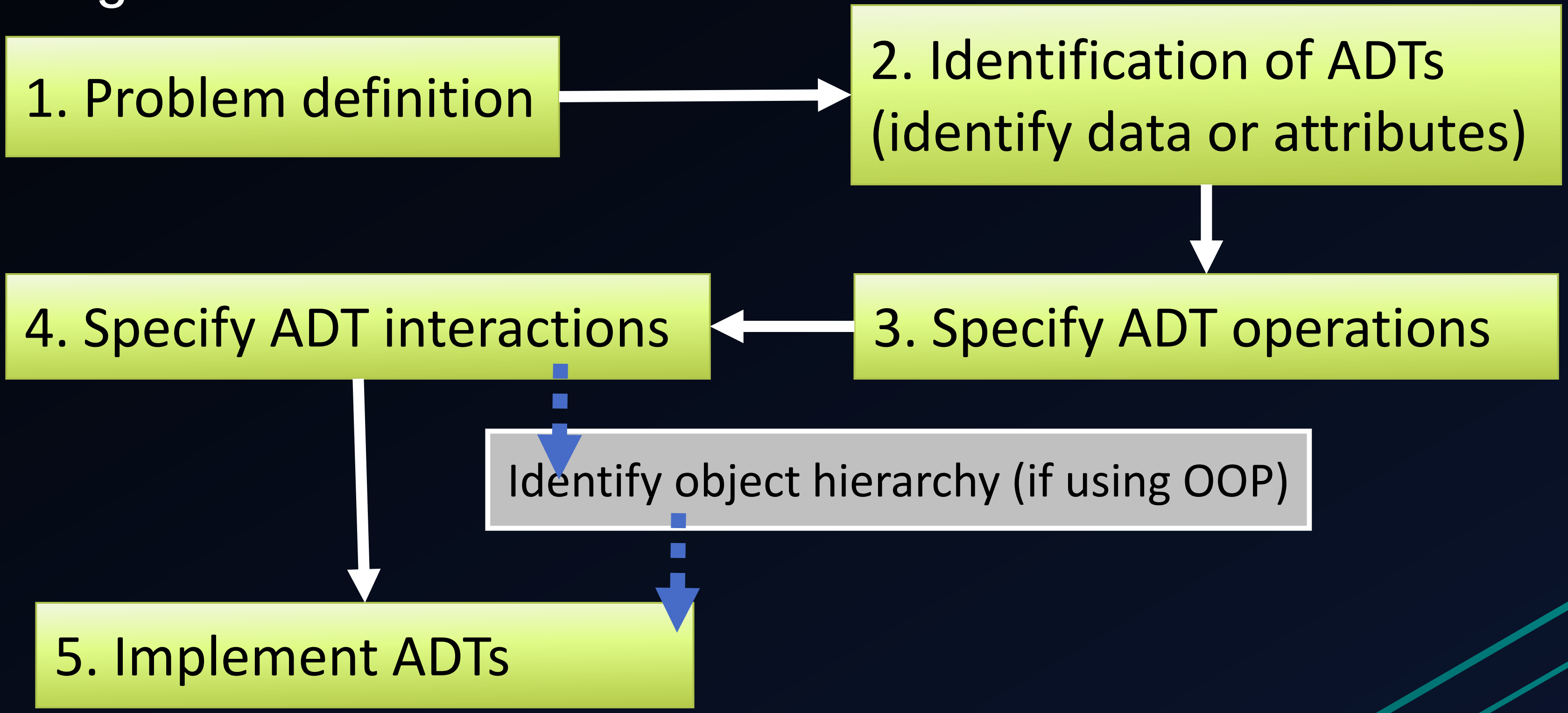
- Easy to modify, maintain
- Profitable
- Reusable

**Client (who uses ADT) benefits:**

- ✓ Simple To Use, Understand
- ✓ Familiar
- ✓ Cheap
- ✓ Component-based

## 4. ADT FROM DEFINITION TO IMPLEMENTATION

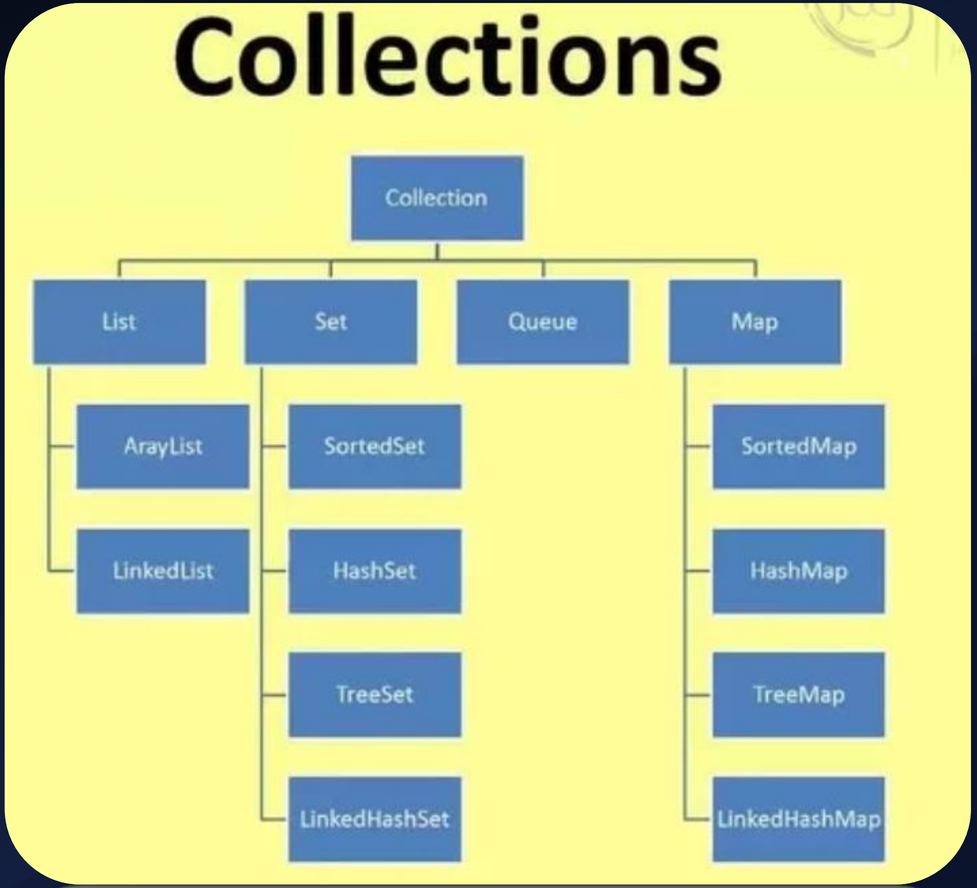
System design with ADTs


1. Problem definition
2. Identification of ADTs (identify data or attributes)
3. Specify ADT operations
4. Specify ADT interactions
Identify object hierarchy (if using OOP)
5. Implement ADTs

## 5. ADTs in Java?

**Java library** has Abstract Data Types such as List, Stack, Queue, Set, Map as inbuilt interfaces which are being implemented using various data structures.

**The JDK** does not provide any direct implementations of this interface. It provides implementations of more specific sub interfaces like List, Set. This interface is typically used to pass collections around and manipulate them where maximum generality is desired.



## COMMONLY USED DATA STRUCTURES

- Array (Data Structure) (Array List)
- Stack
- Queue
- Hash Table
- Linked List
- Tree (Data Structure)
- Graph (Data Structure)

### Application of the queue

- ❑ Production and consumption (applications in parallel operating systems)
- ❑ Cache (e.g. Keypress -> Buffer -> CPU Processing)
- ❑ Processing instructions in the computer (applications in the operating system, compilers), queues of processes waiting to be processed

## P2. What is the Stack Data Type?

The stack is a linear data structure that can only be accessed at one of its ends to store and retrieve data. Example A Stack of Disks: The last disc placed last will be removed first of that stack.

For this reason, a stack is called a LIFO structure: Last in/First out.

## P2. Operations on the Stack Data Type?

- **isEmpty()** — Check to see if the stack is empty.
- **isFull()** — Check to see if the stack is full.
- **Push(element)** — Put the element on the top of the stack.
- **pop()** —Take the topmost element from the stack.
- **peek()** —Return the topmost element in the stack without removing it.

# A series of operations executed on a stack

## 1. PUSH Operation with Array

The process of putting a new data element onto stack is known as a Push Operation. Push operation involves a series of steps −

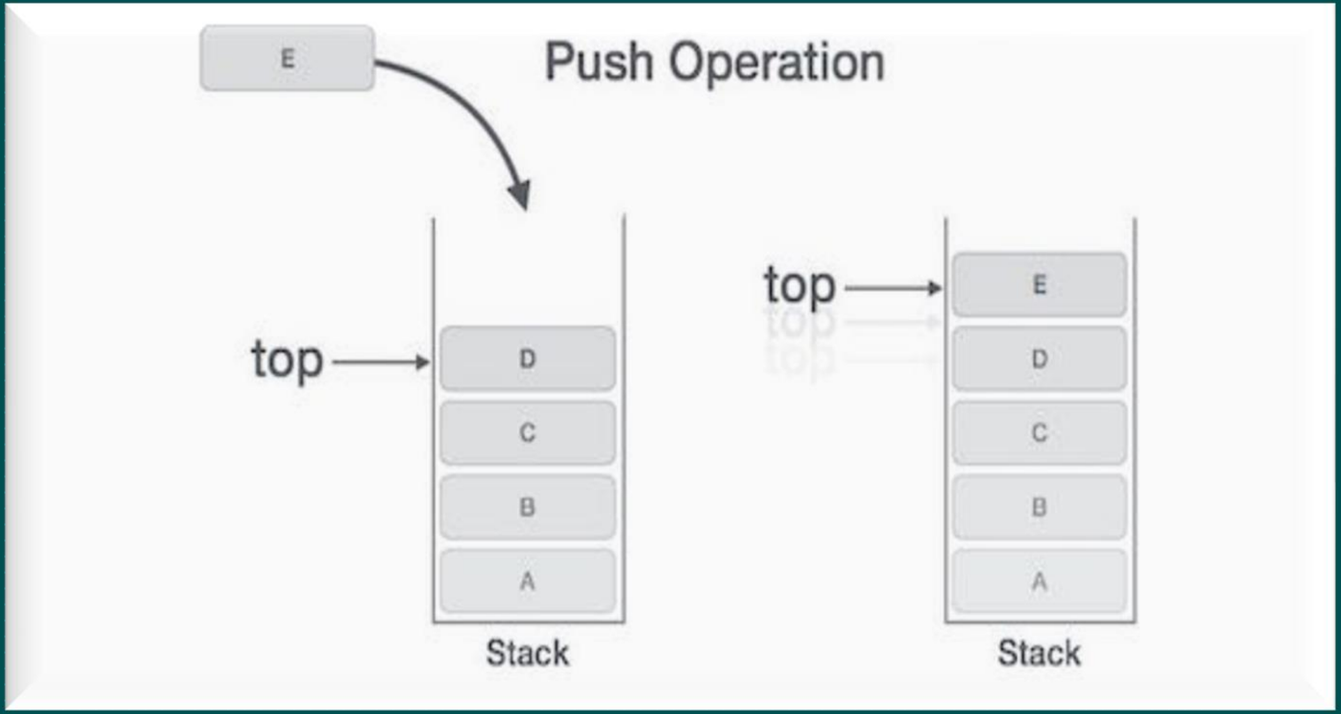
**Step 1:** Checks if the stack is full.

**Step 2**: If the stack is full, produces an error.

**Step 3**: If the stack is not full, increments **top** to point next empty space.

**Step 4**: Adds data element to the stack location, where top is pointing.

**Step 5**: Returns success.



## A series of operations executed on a stack

## 2. POP Operation with Array

Accessing the content while removing it from the stack, is known as a Pop Operation. In an array implementation of pop() operation, the data element is not actually removed, instead **top** is decremented to a lower position in the stack to point to the next value. But in linked-list implementation, pop() actually removes data element and deallocates memory space.

**A Pop operation may involve the following steps −**

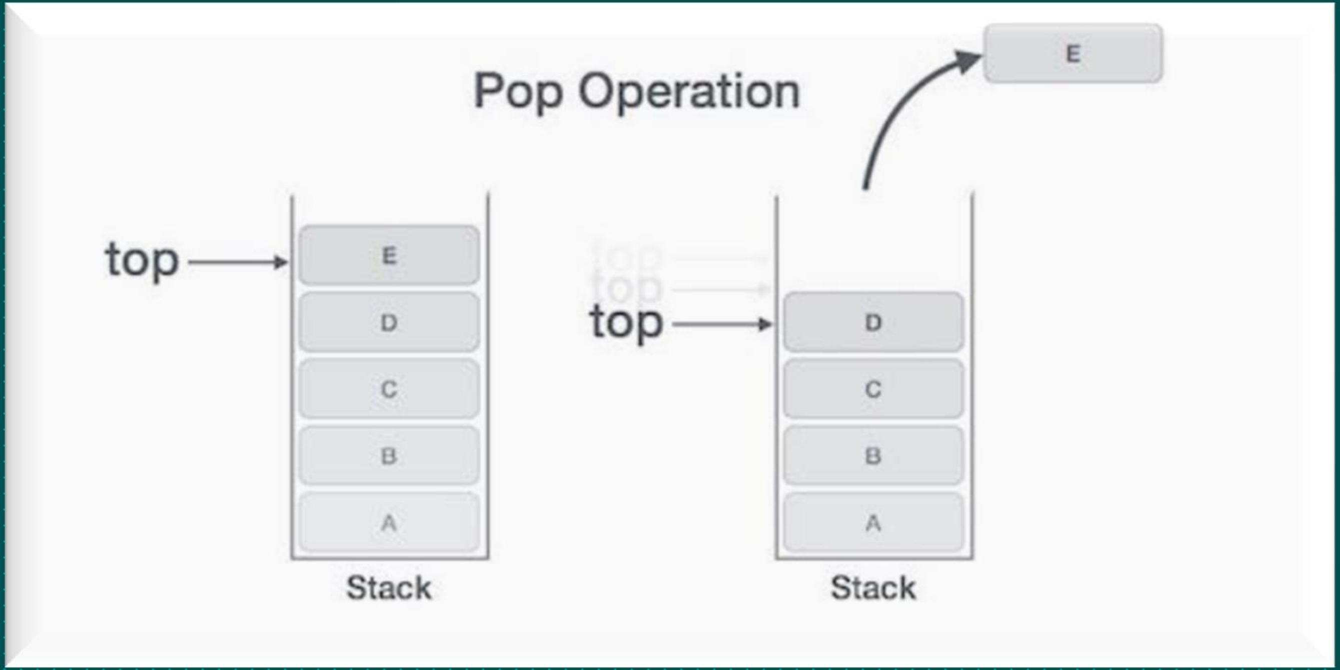
**Step 1:** Checks if the stack is empty.

**Step 2**: If the stack is empty, produces an error.

**Step 3**: If the stack is not empty, accesses the data element at which **top** is pointing.

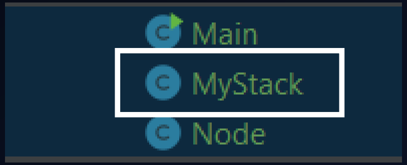
**Step 4**: Decreases the value of top by 1.

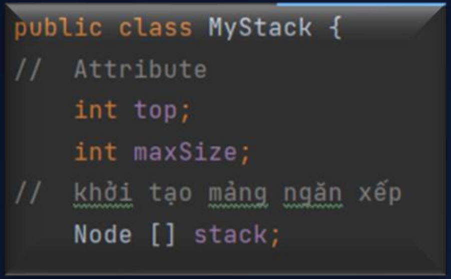
**Step 5**: Returns success.



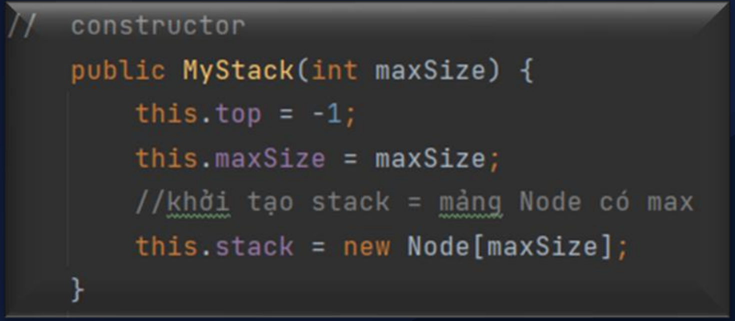
## P3. Specify an Abstract Data Type for the software stack

### Stack with Array

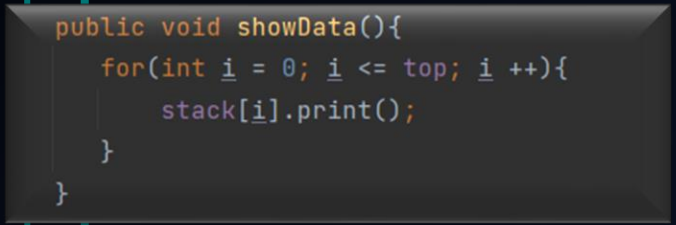




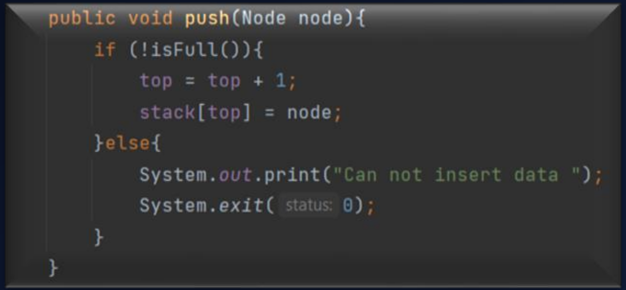
```java
public class MyStack {
//  Attribute
    int top;
    int maxSize;
//  khởi tạo mảng ngăn xếp
    Node [] stack;
```



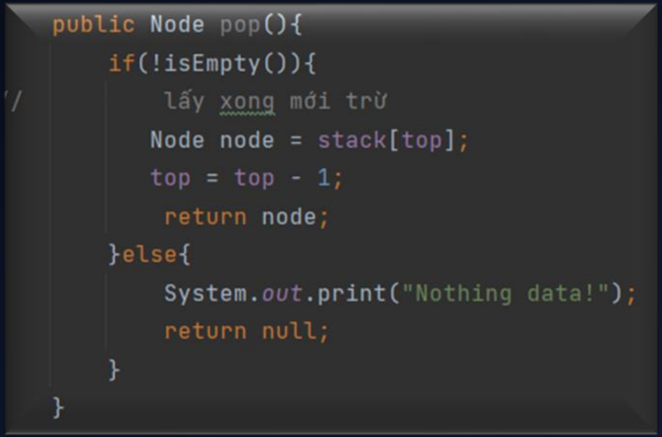
```java
//  constructor
    public MyStack(int maxSize) {
        this.top = -1;
        this.maxSize = maxSize;
        //khởi tạo stack = mảng Node có max
        this.stack = new Node[maxSize];
    }
```



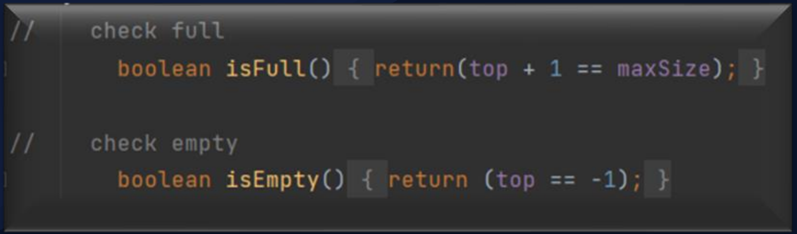
```java
public void showData(){
    for(int i = 0; i <= top; i ++){
        stack[i].print();
    }
}
```



```java
public void push(Node node){
    if (!isFull()){
        top = top + 1;
        stack[top] = node;
    }else{
        System.out.print("Can not insert data ");
        System.exit( status: 0);
    }
}
```

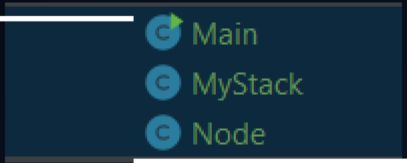


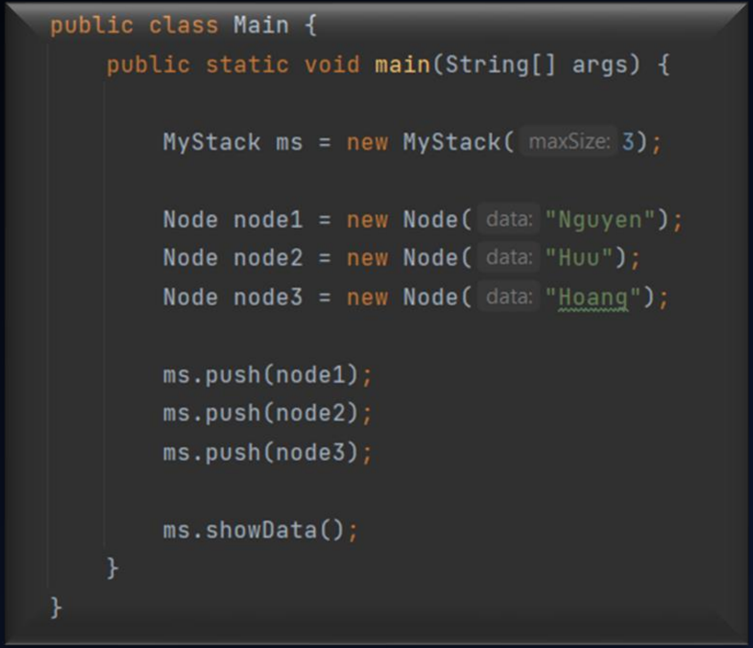
```java
public Node pop(){
    if(!isEmpty()){
/          lấy xong mới trừ
        Node node = stack[top];
        top = top - 1;
         return node;
    }else{
         System.out.print("Nothing data!");
         return null;
    }
}
```



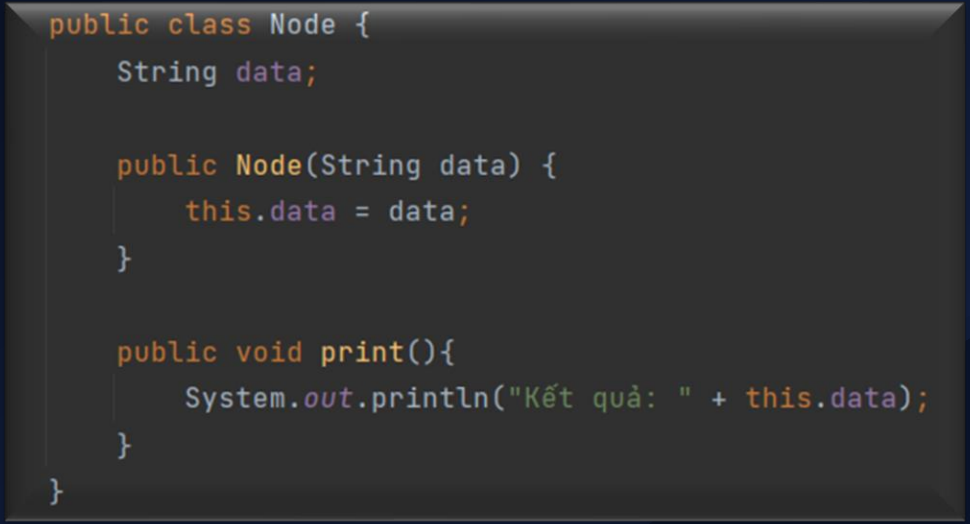
```java
//   check full
     boolean isFull() { return(top + 1 == maxSize); }

//   check empty
     boolean isEmpty() { return (top == -1); }
```

## P3. Specify an Abstract Data Type for the software stack





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

        MyStack ms = new MyStack( maxSize: 3);

        Node node1 = new Node( data: "Nguyen");
        Node node2 = new Node( data: "Huu");
        Node node3 = new Node( data: "Hoang");

        ms.push(node1);
        ms.push(node2);
        ms.push(node3);

        ms.showData();
    }
}
```

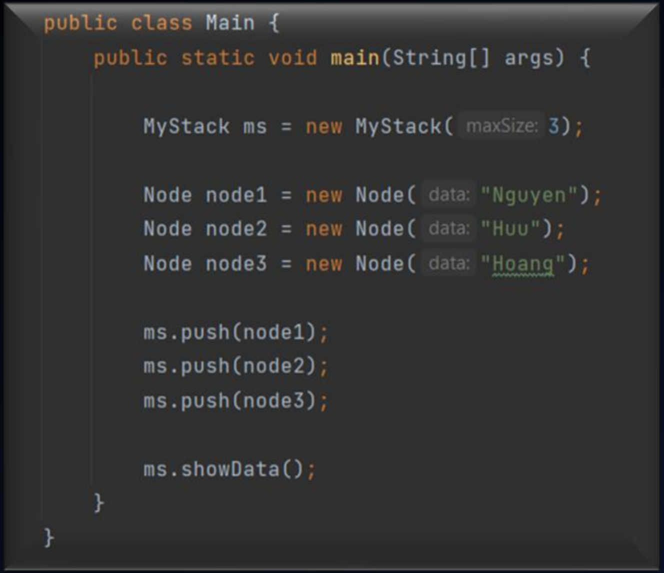


```
public class Node {
    String data;

    public Node(String data) {
        this.data = data;
    }

    public void print(){
        System.out.println("Kết quả: " + this.data);
    }
}
```

In the main class I put 3 data **Nguyen Huu Hoang** and pushed them onto the stack – then displayed them

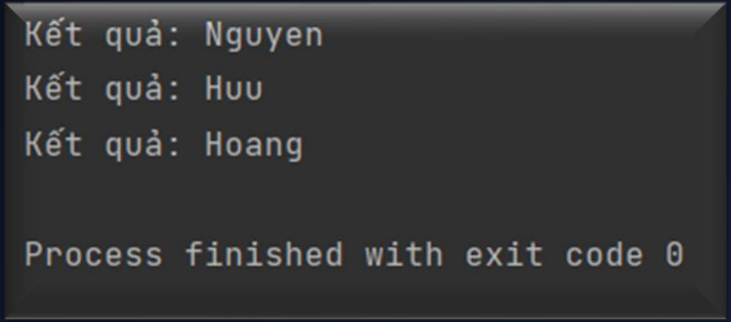
Class Node has attribute, constructor, and 1 display function

# P3. Specify an Abstract Data Type for the software stack

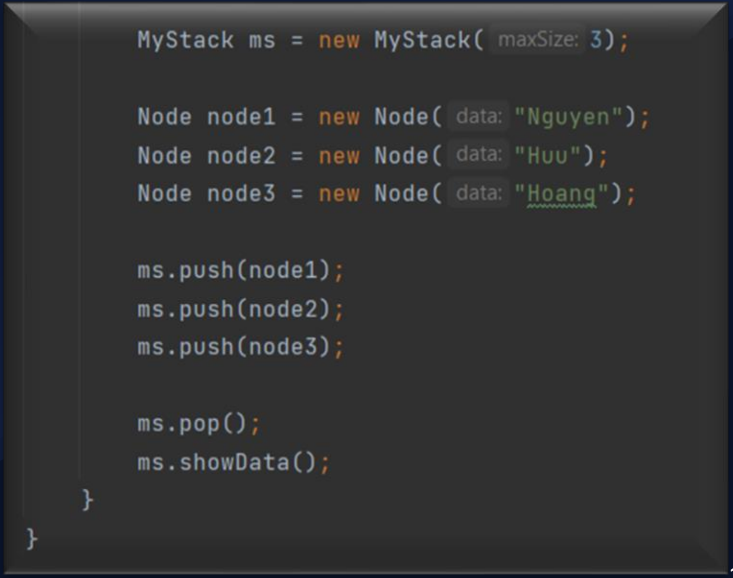


```
public class Main {
    public static void main(String[] args) {

        MyStack ms = new MyStack( maxSize: 3);

        Node node1 = new Node( data: "Nguyen");
        Node node2 = new Node( data: "Huu");
        Node node3 = new Node( data: "Hoang");

        ms.push(node1);
        ms.push(node2);
        ms.push(node3);

        ms.showData();
    }
}
```

Result after push

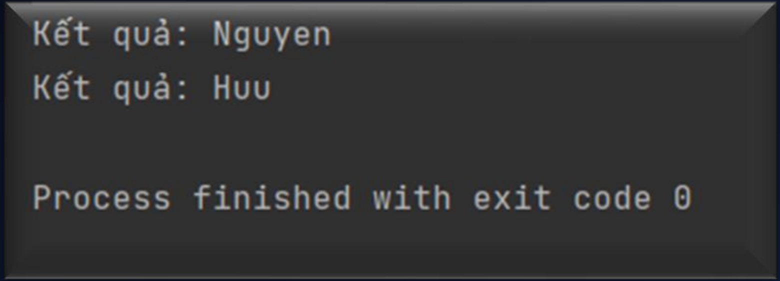


```
Kết quả: Nguyen
Kết quả: Huu
Kết quả: Hoang

Process finished with exit code 0
```



```
        MyStack ms = new MyStack( maxSize: 3);

        Node node1 = new Node( data: "Nguyen");
        Node node2 = new Node( data: "Huu");
        Node node3 = new Node( data: "Hoang");

        ms.push(node1);
        ms.push(node2);
        ms.push(node3);

        ms.pop();
        ms.showData();
    }
}
```

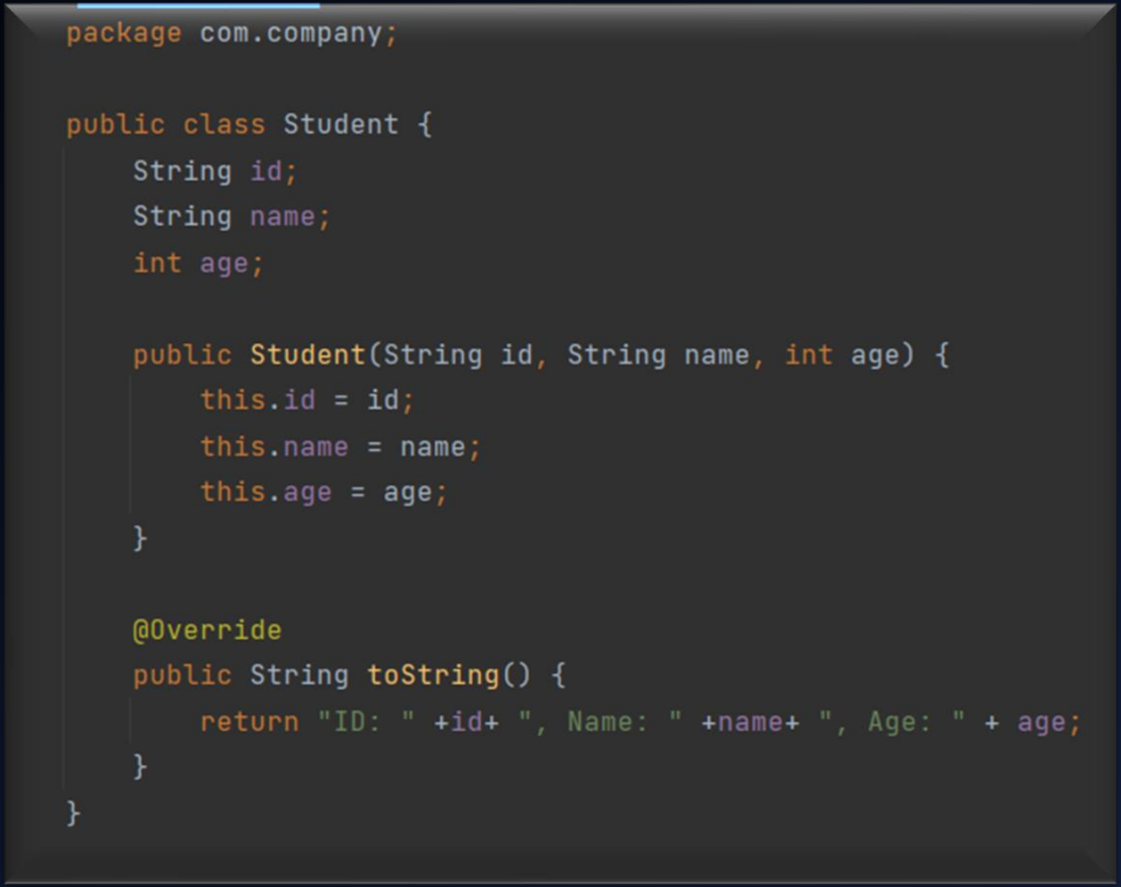
Result after pop



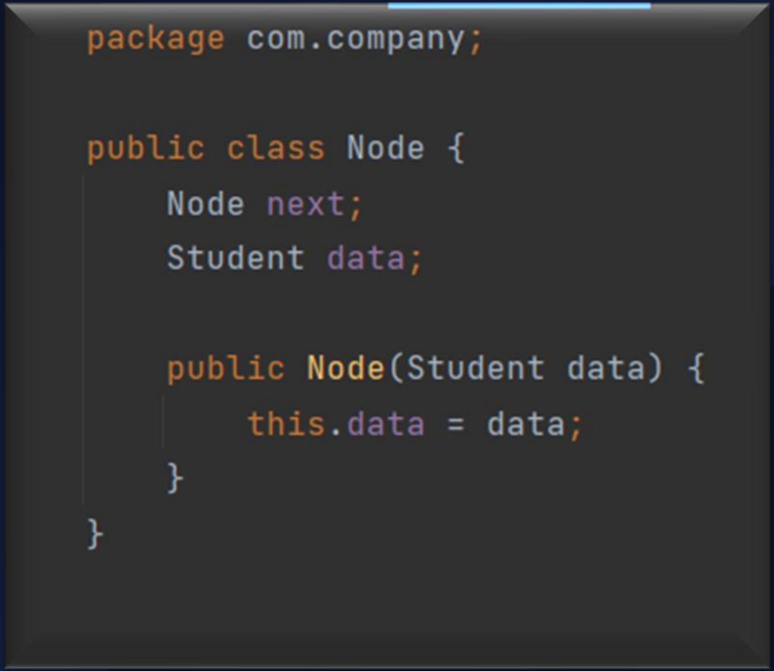
```
Kết quả: Nguyen
Kết quả: Huu

Process finished with exit code 0
```

## P3. Specify an Abstract Data Type for the software stack

### Stack with LinkedList

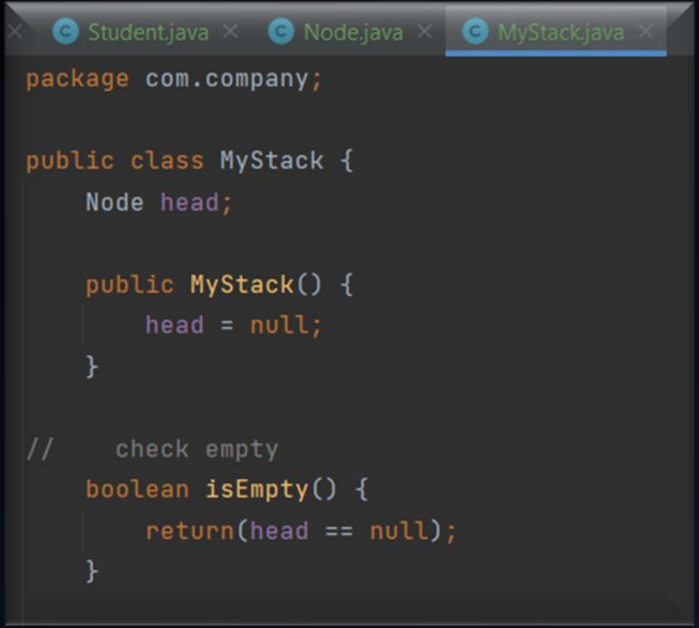


```java
package com.company;

public class Student {
    String id;
    String name;
    int age;

    public Student(String id, String name, int age) {
        this.id = id;
        this.name = name;
        this.age = age;
    }

    @Override
    public String toString() {
        return "ID: " +id+ ", Name: " +name+ ", Age: " + age;
    }
}
```



```java
package com.company;

public class Node {
    Node next;
    Student data;

    public Node(Student data) {
        this.data = data;
    }
}
```

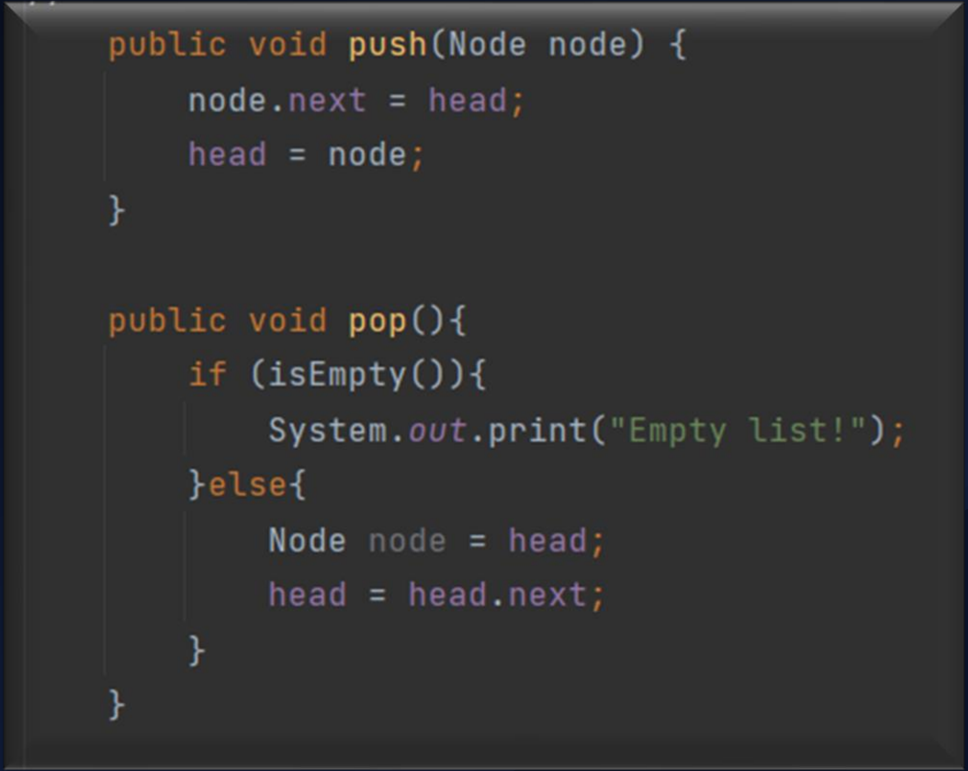
## P3. Specify an Abstract Data Type for the software stack


Student.java × Node.java × MyStack.java ×

```java
package com.company;

public class MyStack {
    Node head;

    public MyStack() {
        head = null;
    }

//    check empty
    boolean isEmpty() {
        return(head == null);
    }
```

isEmpty()

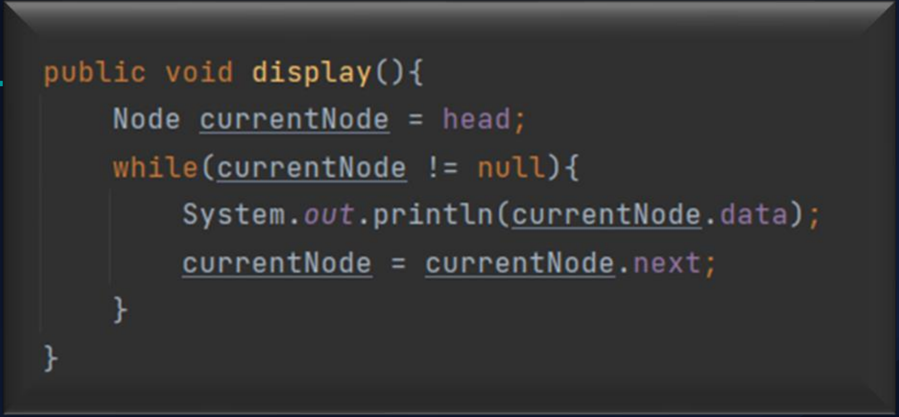
Push()



```java
public void push(Node node) {
    node.next = head;
    head = node;
}

public void pop(){
    if (isEmpty()){
        System.out.print("Empty list!");
    }else{
        Node node = head;
        head = head.next;
    }
}
```

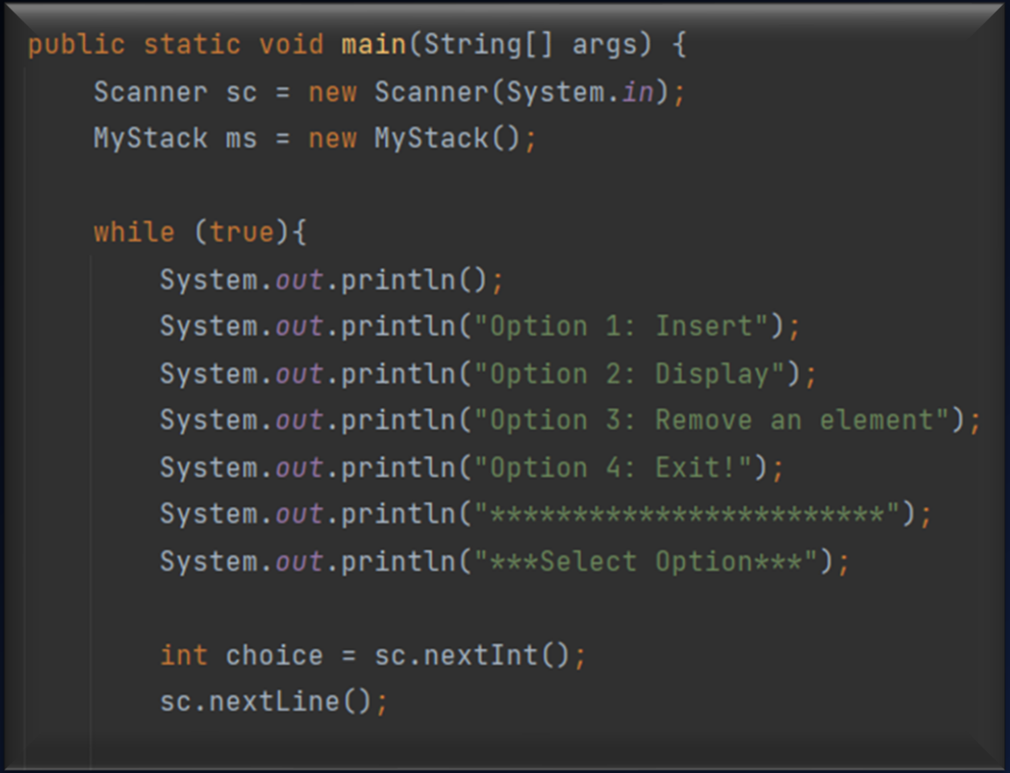
POP()

Display()

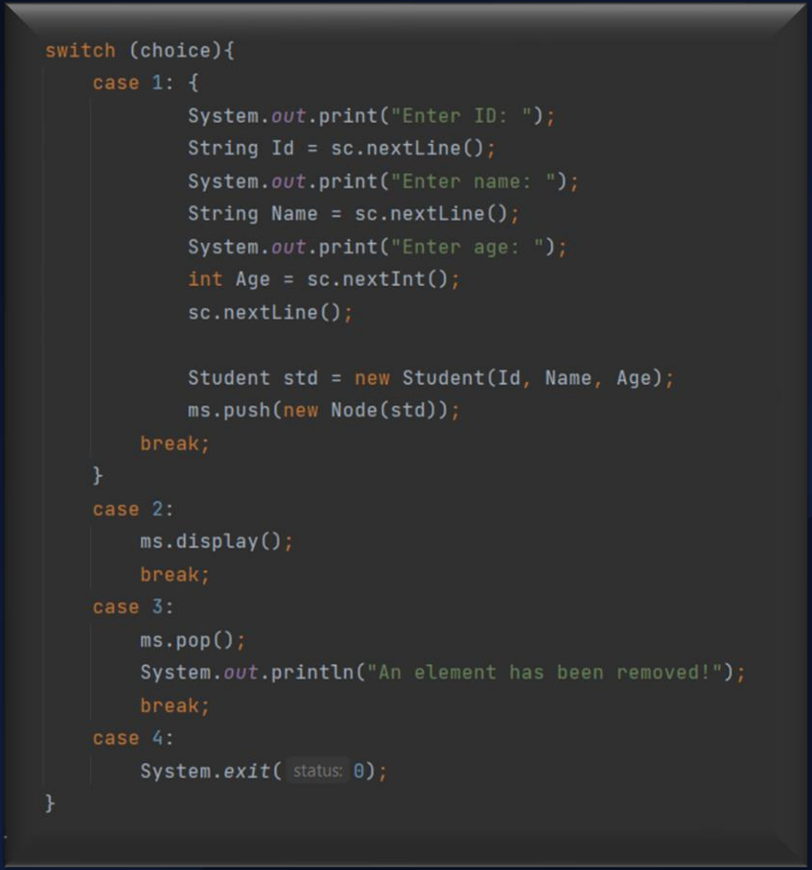


```java
public void display(){
    Node currentNode = head;
    while(currentNode != null){
        System.out.println(currentNode.data);
        currentNode = currentNode.next;
    }
}
```

## P3. Specify an Abstract Data Type for the software stack



```
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    MyStack ms = new MyStack();

    while (true){
        System.out.println();
        System.out.println("Option 1: Insert");
        System.out.println("Option 2: Display");
        System.out.println("Option 3: Remove an element");
        System.out.println("Option 4: Exit!");
        System.out.println("**********************");
        System.out.println("***Select Option***");

        int choice = sc.nextInt();
        sc.nextLine();
```

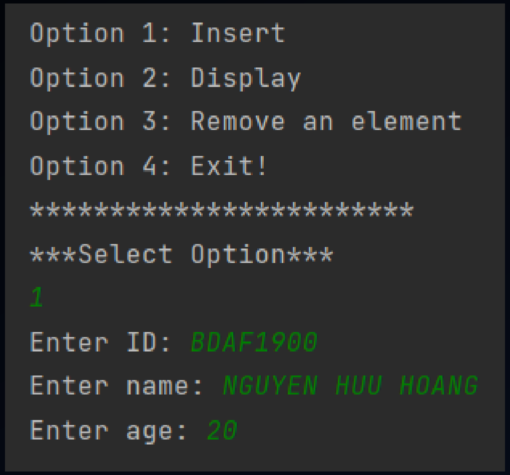


```
switch (choice){
    case 1: {
            System.out.print("Enter ID: ");
            String Id = sc.nextLine();
            System.out.print("Enter name: ");
            String Name = sc.nextLine();
            System.out.print("Enter age: ");
            int Age = sc.nextInt();
            sc.nextLine();

            Student std = new Student(Id, Name, Age);
            ms.push(new Node(std));
        break;
    }
    case 2:
        ms.display();
        break;
    case 3:
        ms.pop();
        System.out.println("An element has been removed!");
        break;
    case 4:
        System.exit( status: 0);
}
```

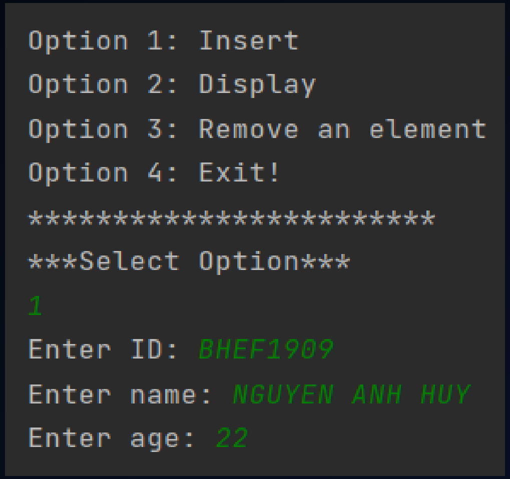
In the main class, we will deploy the program using the Switch case. There are 4 cases for different options such as:
**Insert, display, remove** information and **exit** the program.

## P3. Specify an Abstract Data Type for the software stack

### INSERT

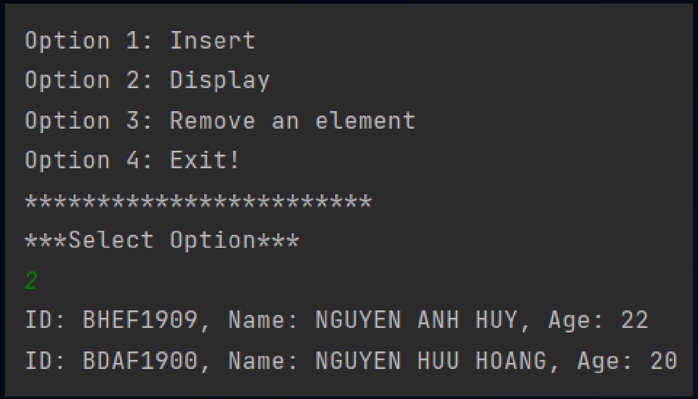


```
Option 1: Insert
Option 2: Display
Option 3: Remove an element
Option 4: Exit!
************************
***Select Option***
1
Enter ID: BDAF1900
Enter name: NGUYEN HUU HOANG
Enter age: 20
```



```
Option 1: Insert
Option 2: Display
Option 3: Remove an element
Option 4: Exit!
************************
***Select Option***
1
Enter ID: BHEF1909
Enter name: NGUYEN ANH HUY
Enter age: 22
```

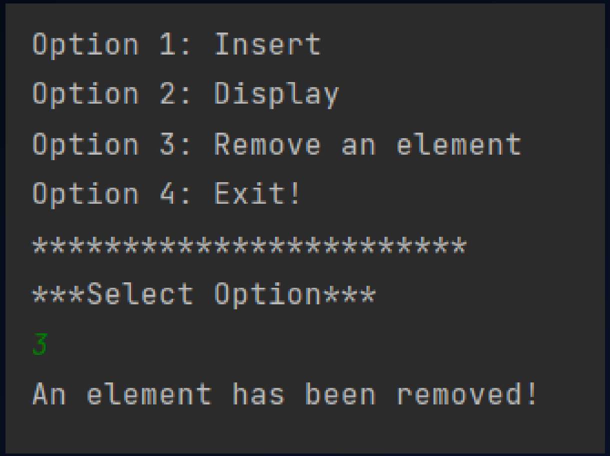
### DISPLAY



```
Option 1: Insert
Option 2: Display
Option 3: Remove an element
Option 4: Exit!
************************
***Select Option***
2
ID: BHEF1909, Name: NGUYEN ANH HUY, Age: 22
ID: BDAF1900, Name: NGUYEN HUU HOANG, Age: 20
```

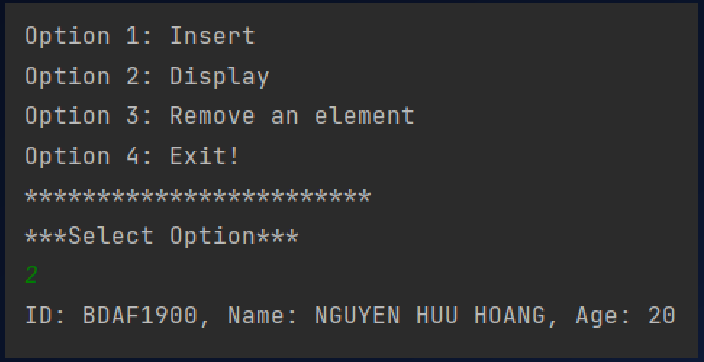
The last input element will be removed

### REMOVE



```
Option 1: Insert
Option 2: Display
Option 3: Remove an element
Option 4: Exit!
************************
***Select Option***
3
An element has been removed!
```

### RESULT



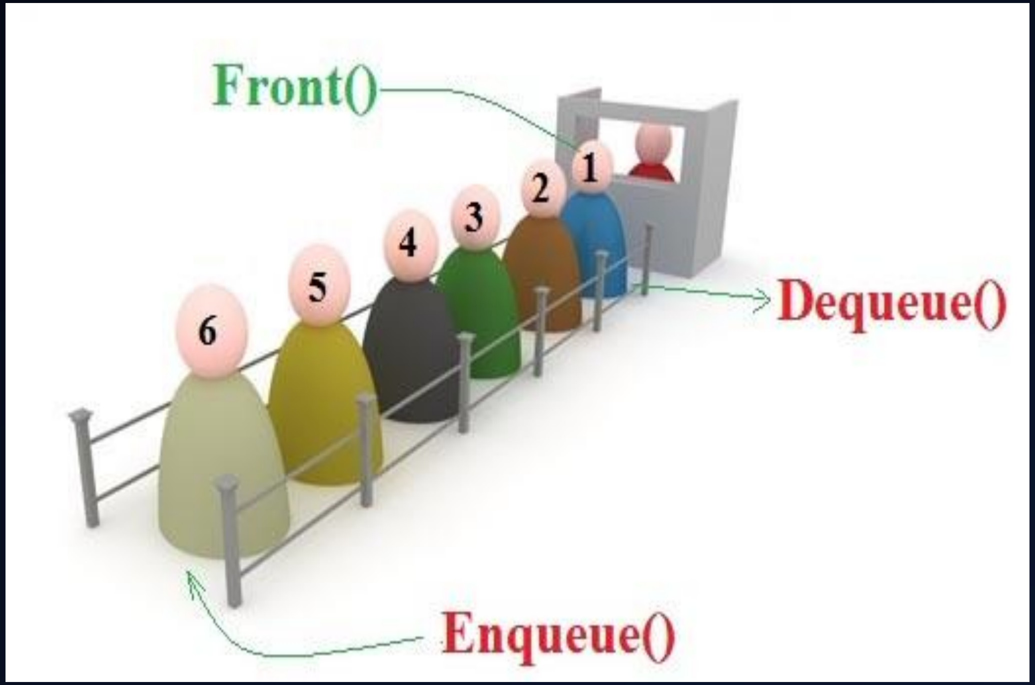
```
Option 1: Insert
Option 2: Display
Option 3: Remove an element
Option 4: Exit!
************************
***Select Option***
2
ID: BDAF1900, Name: NGUYEN HUU HOANG, Age: 20
```

## M1- What is Queue Data Type

### Queue

The queue is a data structure used to hold objects that work according to the FIFO mechanism (short for: First In First Out),

In a queue, objects can be added to the queue at any time, but only the first added object can be removed from the queue. The operation of adding and removing an object from the queue is called "Enqueue" and "Dequeue", respectively. Adding an object always happens at the end of the queue and an element is always removed from the top of the queue



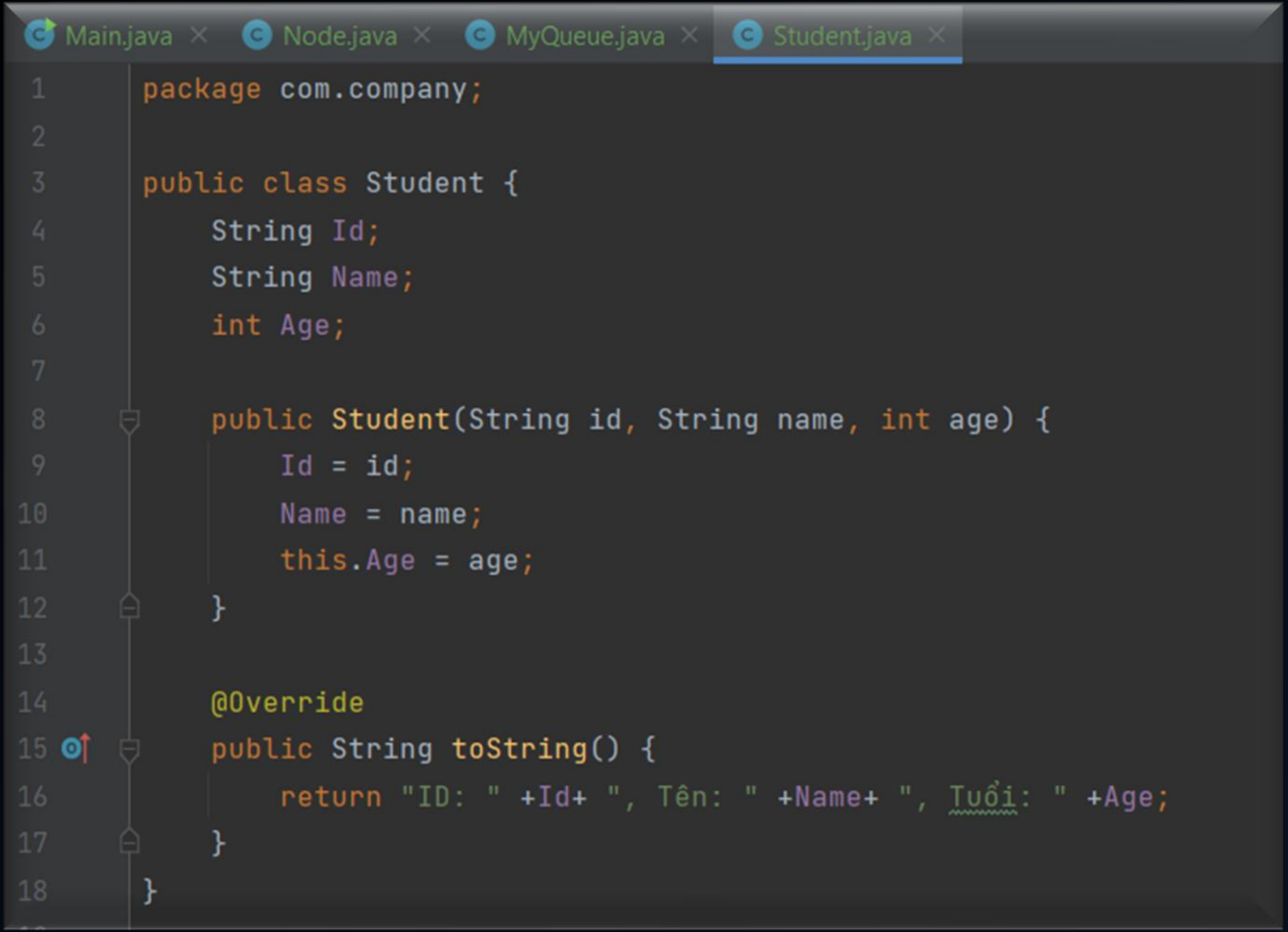
## M1- What is Queue Data Type

Similar to stacks, queues support the following operations:

- **EnQueue():** Adds object to the end of the queue.
- **DeQueue():** Takes the object at the top of the queue out of the queue and returns its value. If the queue is empty then an error will occur.
- **isEmpty():** Check if the queue is empty.
- **isFull():** Check if the queue is full.
- **Front():** Returns the value of the element at the top of the queue without canceling it. If the queue is empty then an error will occur.

The operations of adding, and removing an element must be performed on two different sides of the queue, so the queue operation is performed according to the FIFO principle. Like the stack, a one-dimensional array or a linked list structure can be used to represent a queue structure

## M1- What is Queue Data Type

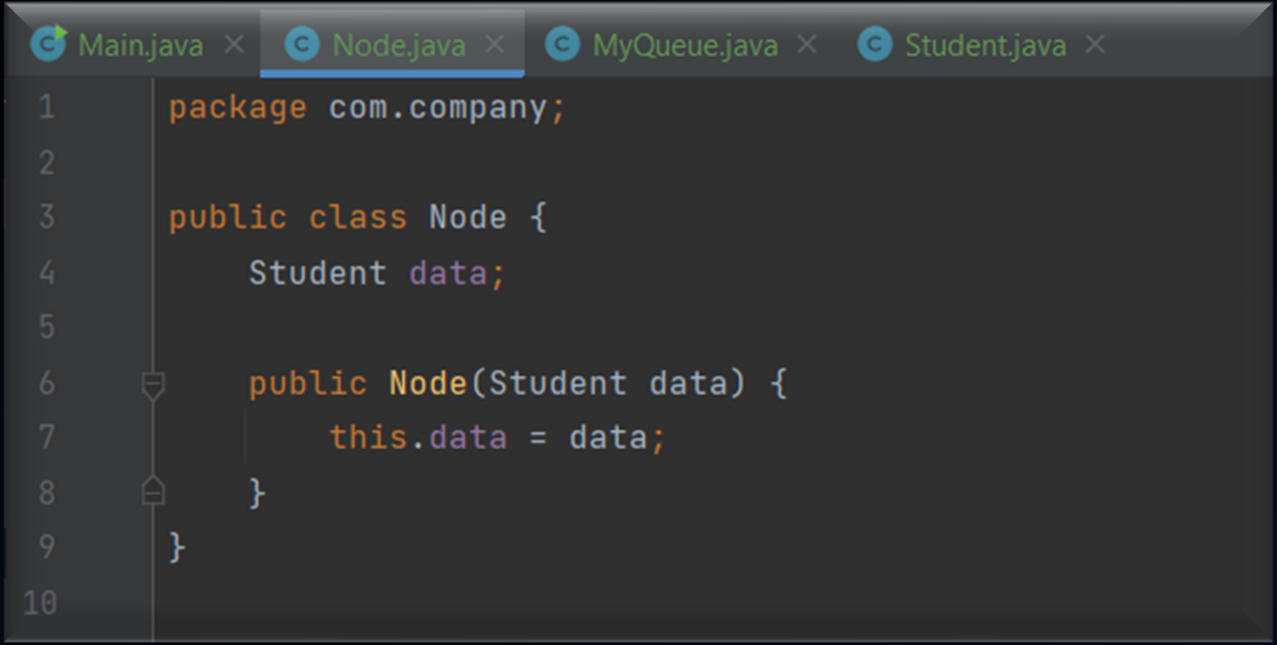


```java
package com.company;

public class Student {
    String Id;
    String Name;
    int Age;

    public Student(String id, String name, int age) {
        Id = id;
        Name = name;
        this.Age = age;
    }

    @Override
    public String toString() {
        return "ID: " +Id+ ", Tên: " +Name+ ", Tuổi: " +Age;
    }
}
```

**Class Student**

- Attributes
- Constructor
- toString

## M1- What is Queue Data Type

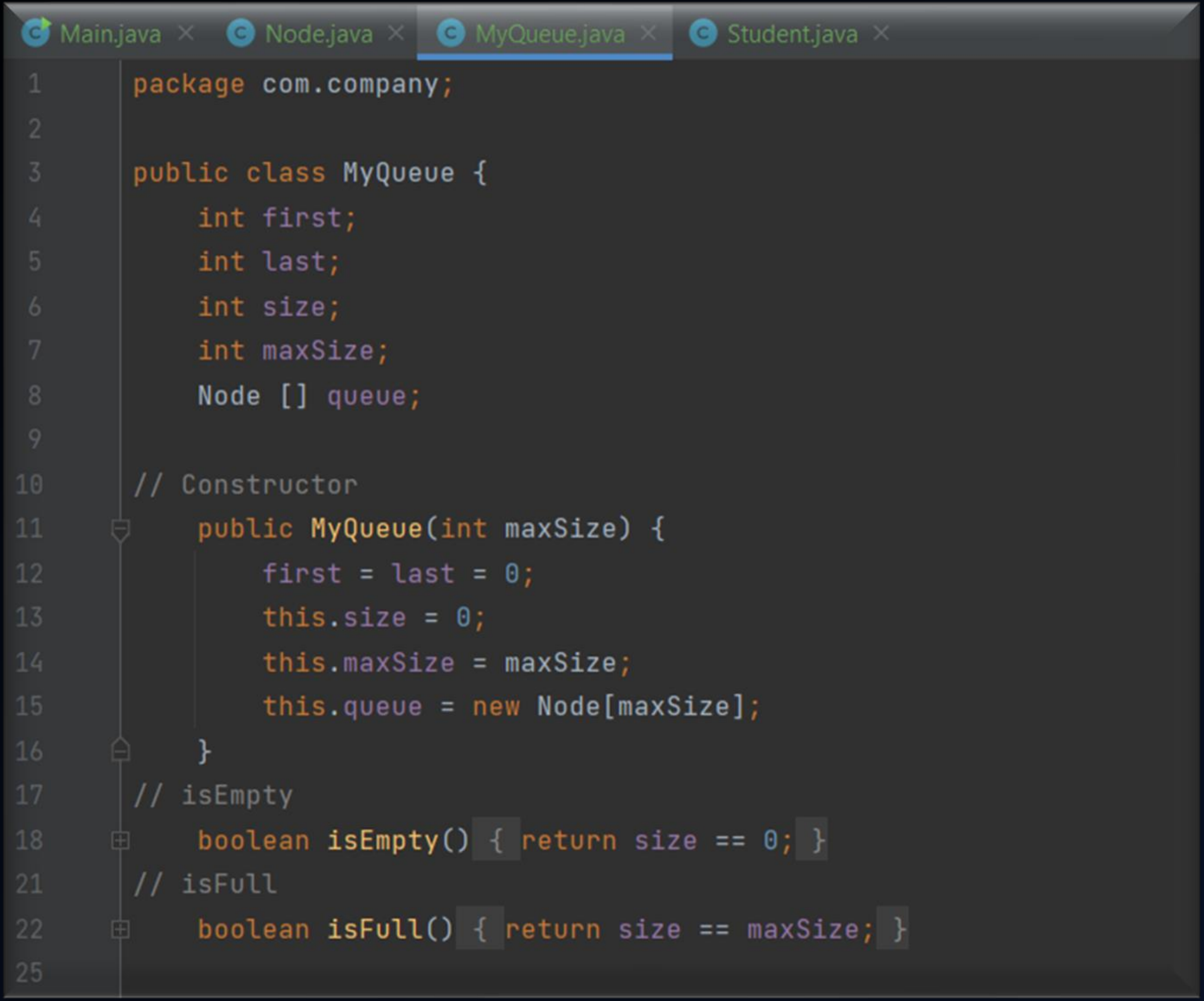


```java
package com.company;

public class Node {
    Student data;

    public Node(Student data) {
        this.data = data;
    }
}

```

**Class Node**

- Attributes: Student data
- Constructor

# M1- What is Queue Data Type



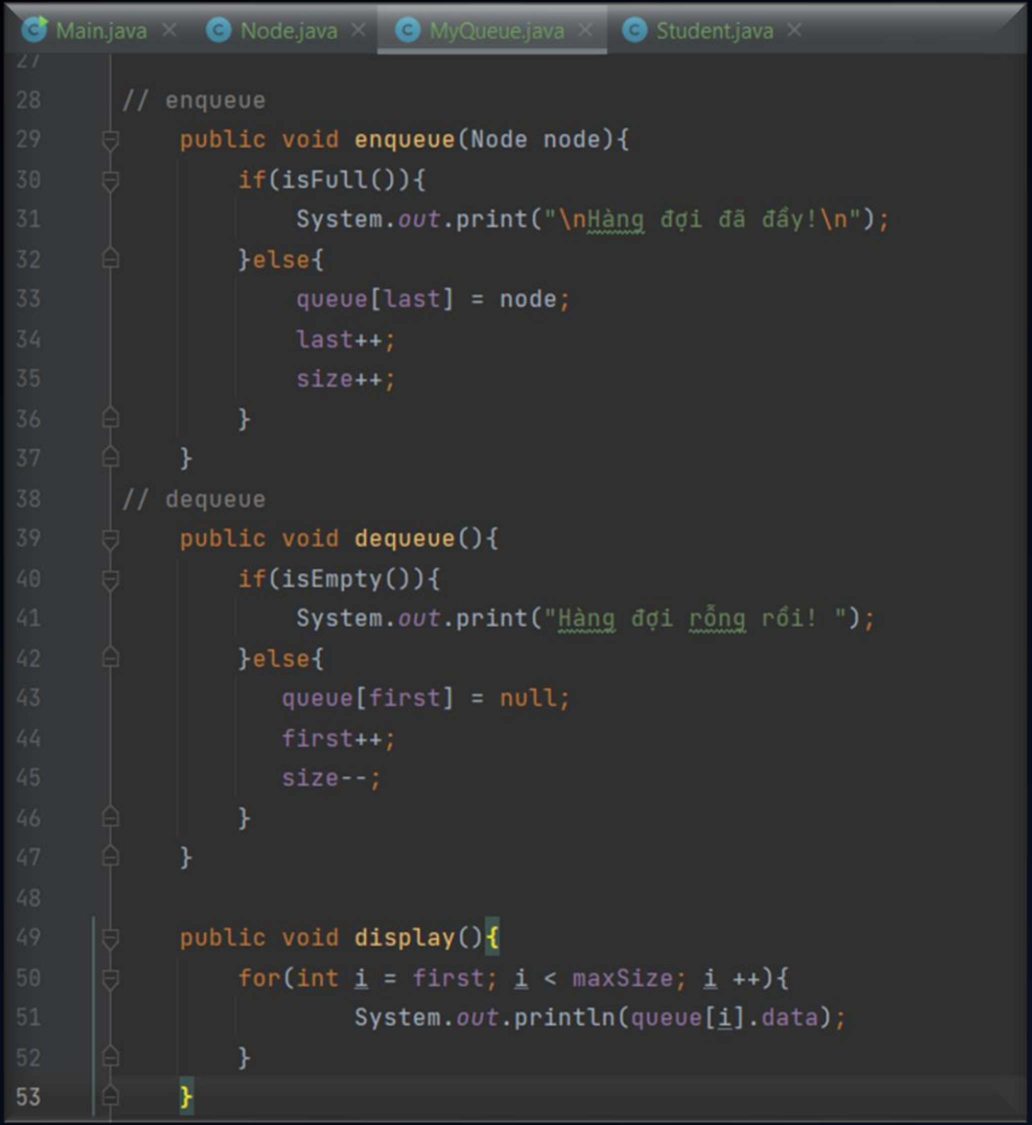
Main.java Node.java MyQueue.java Student.java

```java
package com.company;

public class MyQueue {
    int first;
    int last;
    int size;
    int maxSize;
    Node [] queue;

// Constructor
    public MyQueue(int maxSize) {
        first = last = 0;
        this.size = 0;
        this.maxSize = maxSize;
        this.queue = new Node[maxSize];
    }
// isEmpty
    boolean isEmpty() { return size == 0; }
// isFull
    boolean isFull() { return size == maxSize; }
```

**Class MyQueue**

- Attributes
- Constructor
- Check Empty
- Check Full

## M1- What is Queue Data Type



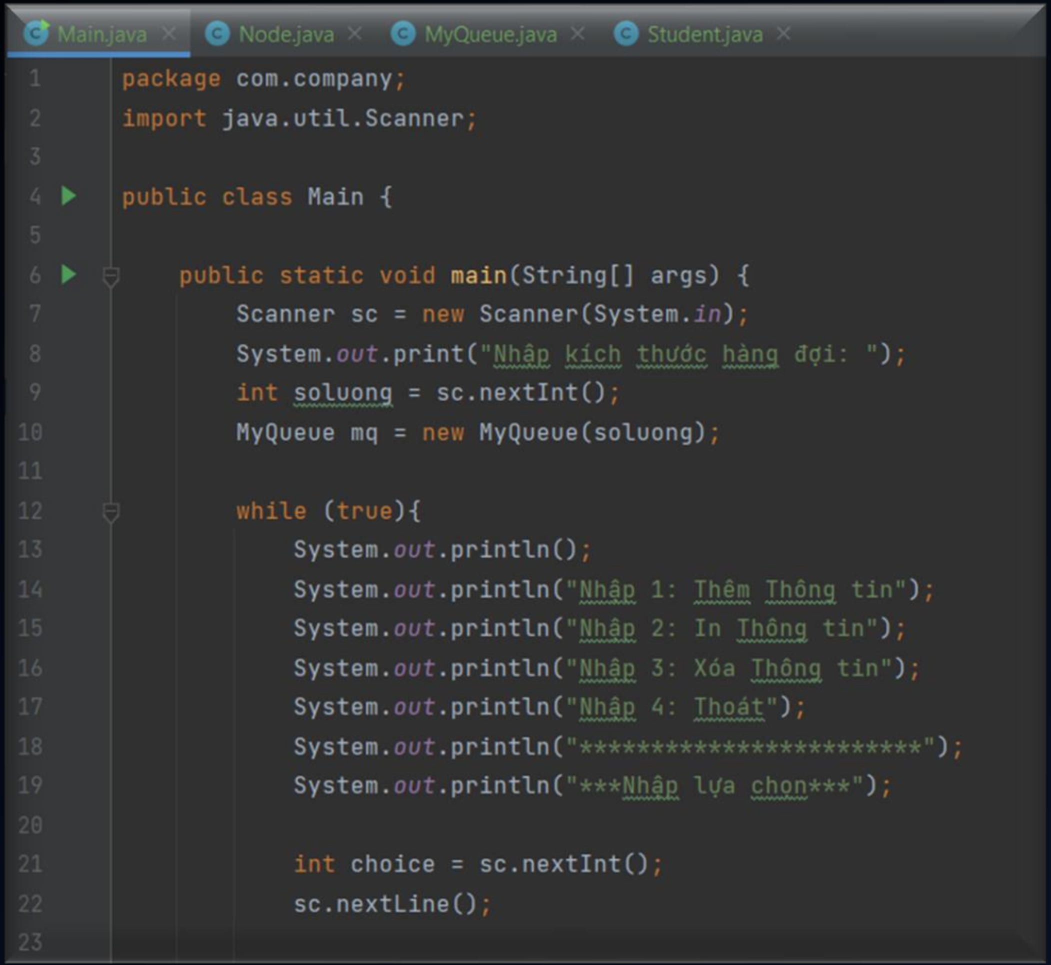
```java
    // enqueue
        public void enqueue(Node node){
            if(isFull()){
                System.out.print("\nHàng đợi đã đầy!\n");
            }else{
                queue[last] = node;
                last++;
                size++;
            }
        }
    // dequeue
        public void dequeue(){
            if(isEmpty()){
                System.out.print("Hàng đợi rỗng rồi! ");
            }else{
              queue[first] = null;
              first++;
              size--;
            }
        }

        public void display(){
            for(int i = first; i < maxSize; i ++){
                System.out.println(queue[i].data);
            }
        }
```

**Class MyQueue**

- Enqueue
- Dequeue
- Display

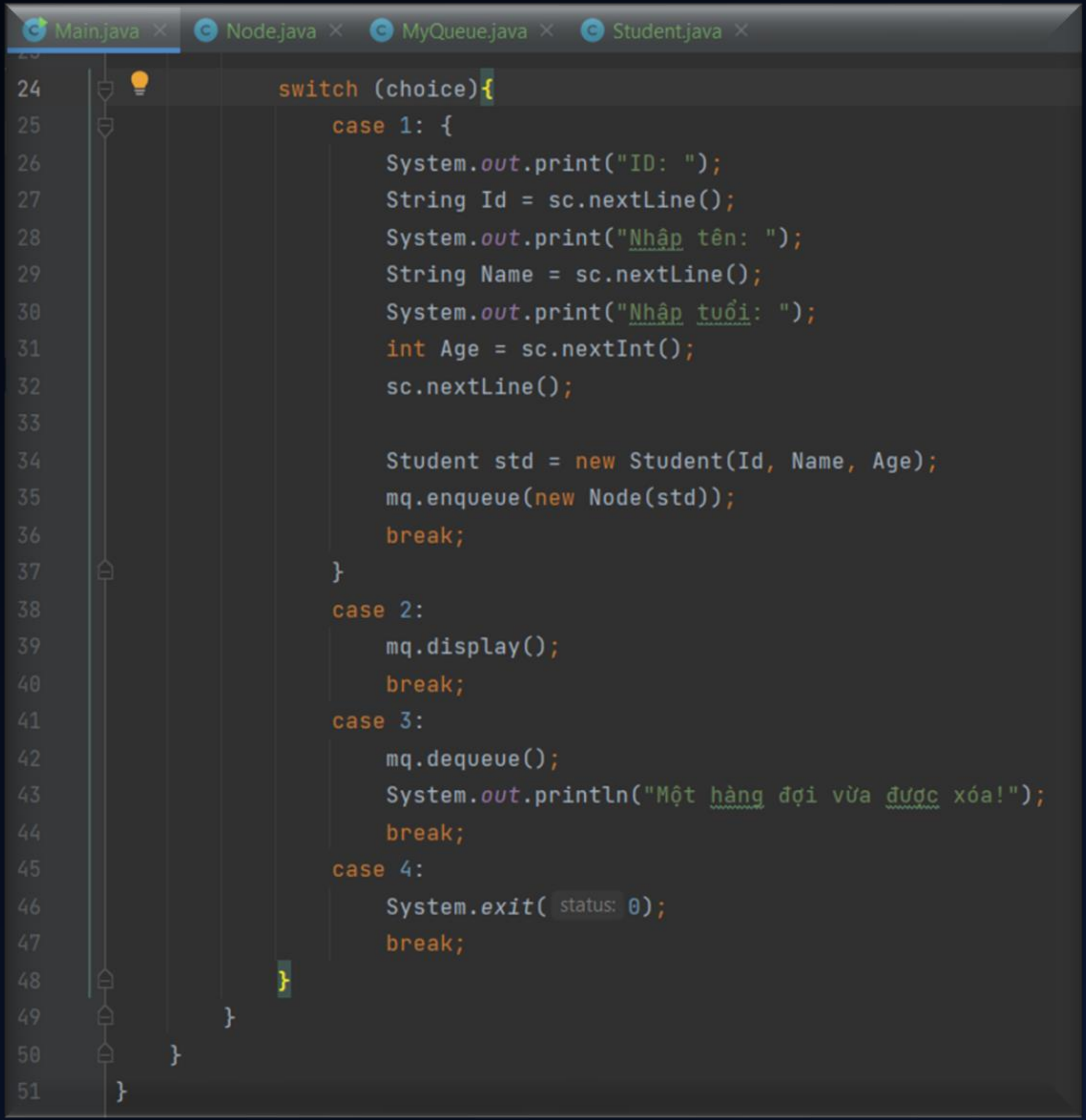
# M1- What is Queue Data Type

## Class Main



Main.java × Node.java × MyQueue.java × Student.java ×

```java
package com.company;
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Nhập kích thước hàng đợi: ");
        int soluong = sc.nextInt();
        MyQueue mq = new MyQueue(soluong);

        while (true){
            System.out.println();
            System.out.println("Nhập 1: Thêm Thông tin");
            System.out.println("Nhập 2: In Thông tin");
            System.out.println("Nhập 3: Xóa Thông tin");
            System.out.println("Nhập 4: Thoát");
            System.out.println("**********************");
            System.out.println("***Nhập lựa chọn***");

            int choice = sc.nextInt();
            sc.nextLine();

```

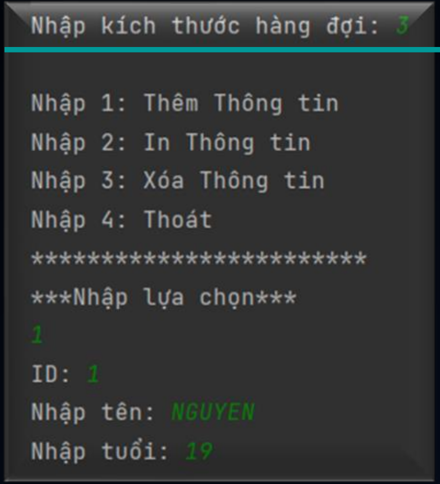


Main.java × Node.java × MyQueue.java × Student.java ×

```java
            switch (choice){
                case 1: {
                    System.out.print("ID: ");
                    String Id = sc.nextLine();
                    System.out.print("Nhập tên: ");
                    String Name = sc.nextLine();
                    System.out.print("Nhập tuổi: ");
                    int Age = sc.nextInt();
                    sc.nextLine();

                    Student std = new Student(Id, Name, Age);
                    mq.enqueue(new Node(std));
                    break;
                }
                case 2:
                    mq.display();
                    break;
                case 3:
                    mq.dequeue();
                    System.out.println("Một hàng đợi vừa được xóa!");
                    break;
                case 4:
                    System.exit( status: 0);
                    break;
            }
        }
    }
}
```

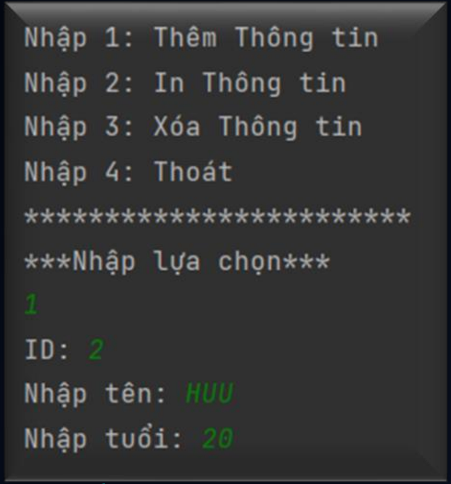
## M1- What is Queue Data Type

**Insert info**

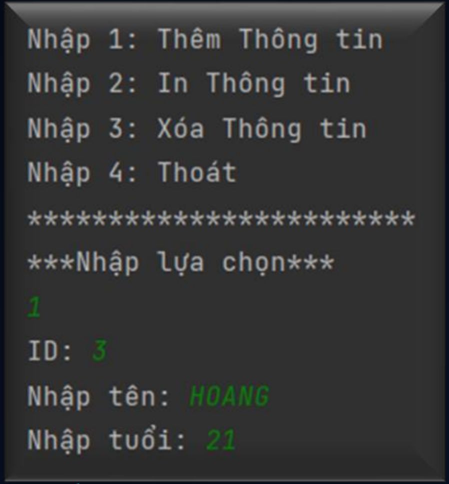


```
Nhập kích thước hàng đợi: 3

Nhập 1: Thêm Thông tin
Nhập 2: In Thông tin
Nhập 3: Xóa Thông tin
Nhập 4: Thoát
***********************
***Nhập lựa chọn***
1
ID: 1
Nhập tên: NGUYEN
Nhập tuổi: 19
```

maxSize = 3



```
Nhập 1: Thêm Thông tin
Nhập 2: In Thông tin
Nhập 3: Xóa Thông tin
Nhập 4: Thoát
***********************
***Nhập lựa chọn***
1
ID: 2
Nhập tên: HUU
Nhập tuổi: 20
```



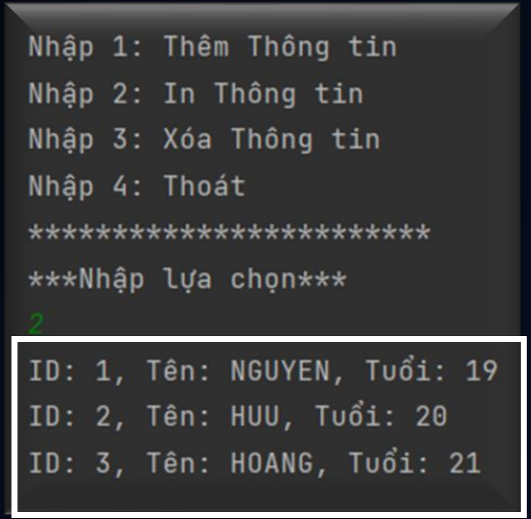
```
Nhập 1: Thêm Thông tin
Nhập 2: In Thông tin
Nhập 3: Xóa Thông tin
Nhập 4: Thoát
***********************
***Nhập lựa chọn***
1
ID: 3
Nhập tên: HOANG
Nhập tuổi: 21
```

ID: 1
Name: NGUYEN
Age: 19

ID: 2
Name: HUU
Age: 20

ID: 3
Name: HOANG
Age: 21

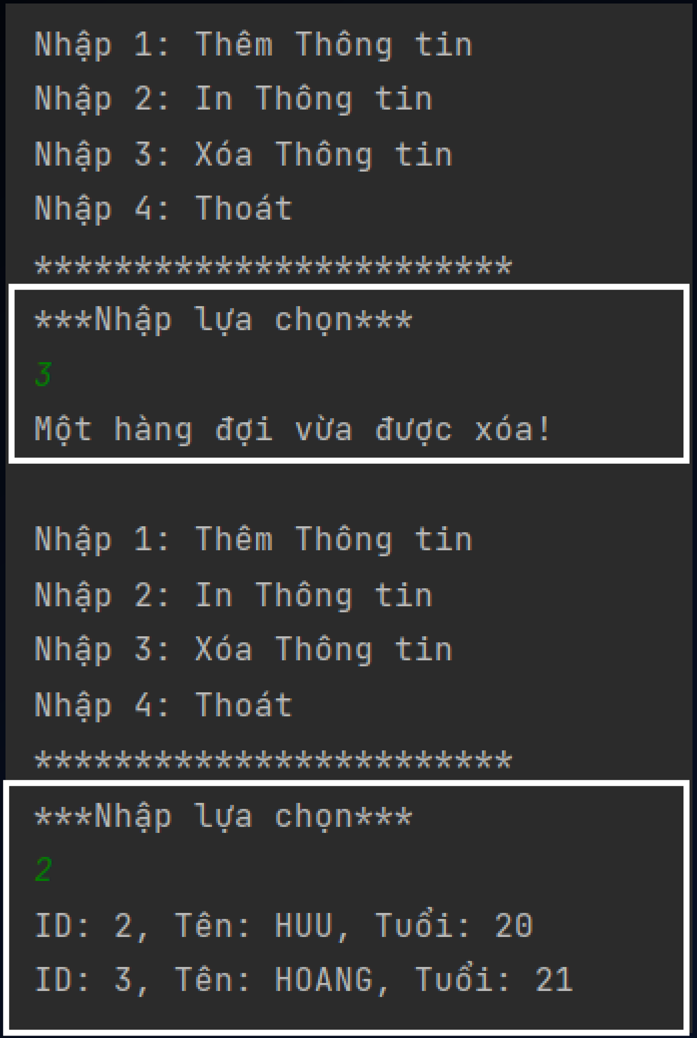
**Display**



```
Nhập 1: Thêm Thông tin
Nhập 2: In Thông tin
Nhập 3: Xóa Thông tin
Nhập 4: Thoát
***********************
***Nhập lựa chọn***
2
ID: 1, Tên: NGUYEN, Tuổi: 19
ID: 2, Tên: HUU, Tuổi: 20
ID: 3, Tên: HOANG, Tuổi: 21
```

## M1- What is Queue Data Type

**Remove**



```
Nhập 1: Thêm Thông tin
Nhập 2: In Thông tin
Nhập 3: Xóa Thông tin
Nhập 4: Thoát
************************
***Nhập lựa chọn***
3
Một hàng đợi vừa được xóa!

Nhập 1: Thêm Thông tin
Nhập 2: In Thông tin
Nhập 3: Xóa Thông tin
Nhập 4: Thoát
************************
***Nhập lựa chọn***
2
ID: 2, Tên: HUU, Tuổi: 20
ID: 3, Tên: HOANG, Tuổi: 21
```

**After removing,**

ID: 1, Name: NGUYEN and Age: 19 disappeared

## M2 - Compare two sorting algorithms?

### SORTING ALGORITHM

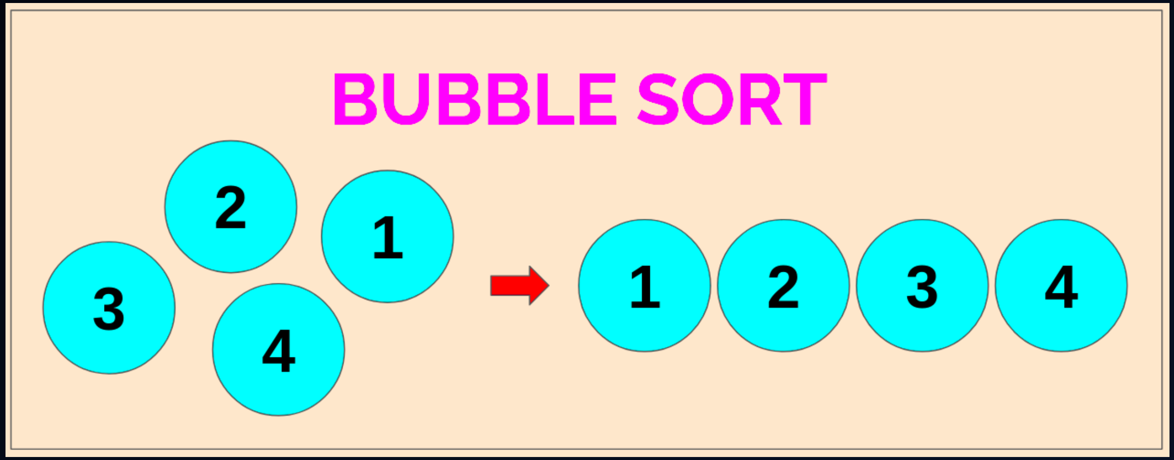
Sort is to arrange data in a specific format such as ascending/descending alphabet, ascending/descending numeric order. In computer science, sorting algorithms determine how to arrange data in a certain order. Sorting here is sorting in numerical or alphabetical order as in a dictionary.

The importance of data sorting lies in the fact that the search for data can be optimized if the data is sorted in a certain order (increasing or decreasing). Sorting is also used to represent data in a more readable format.

M2 - Compare two sorting algorithms?

## BUBBLE SORT IN JAVA?

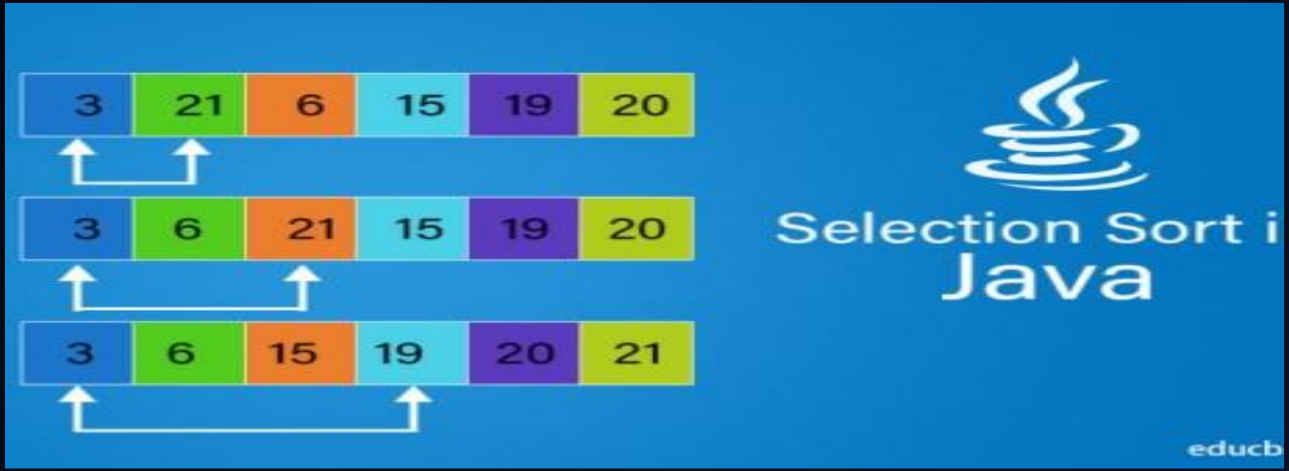
Bubble Sort is a simple sorting algorithm. This sorting algorithm is performed based on comparing pairs of adjacent elements and swapping the order if they are out of order.

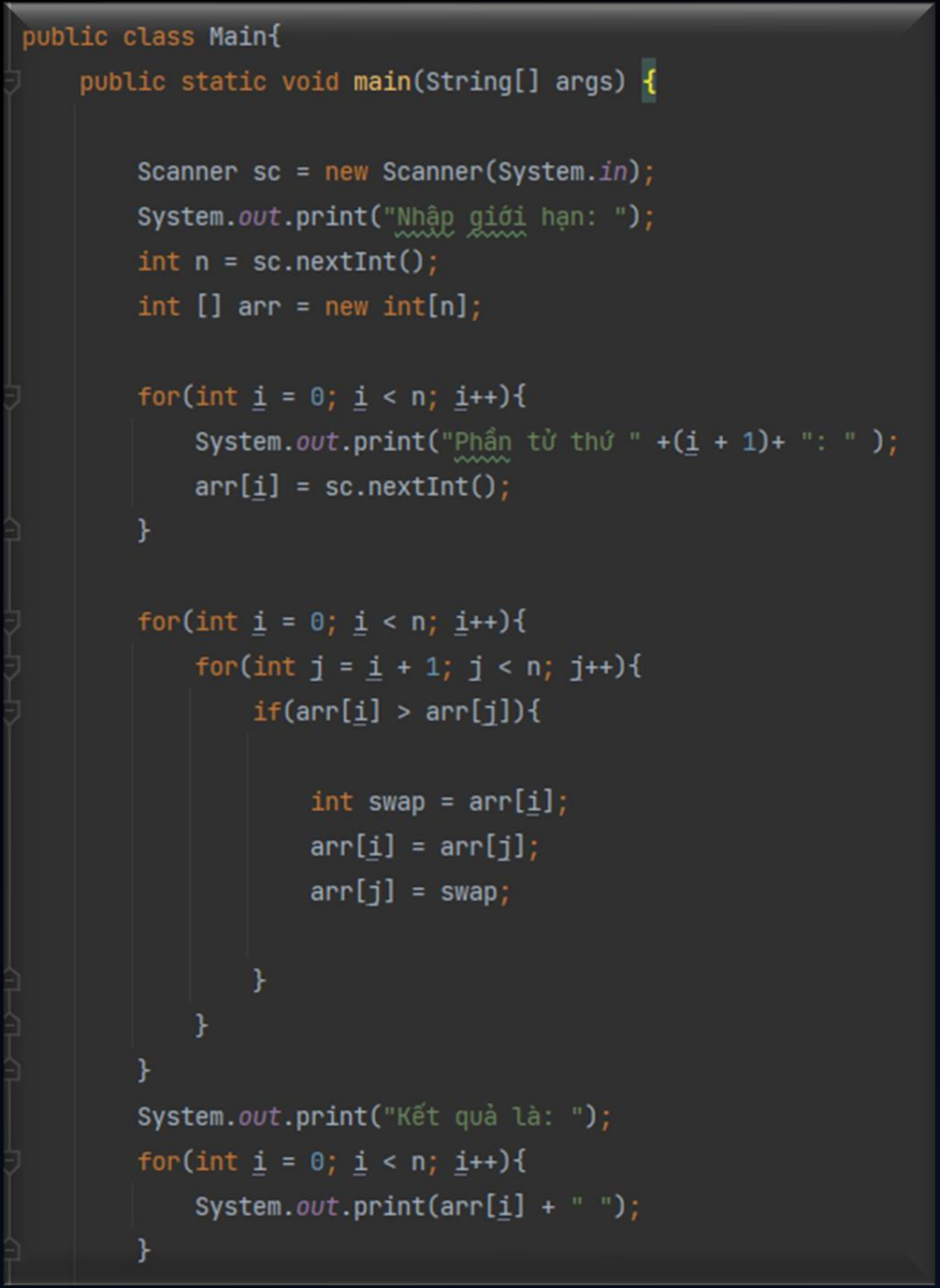


## M2 - Compare two sorting algorithms?

### SELECTION SORT IN JAVA

Selection sort is a sorting algorithm that selects the smallest element from an unsorted list in each iteration and places that element at the beginning of the unsorted list.



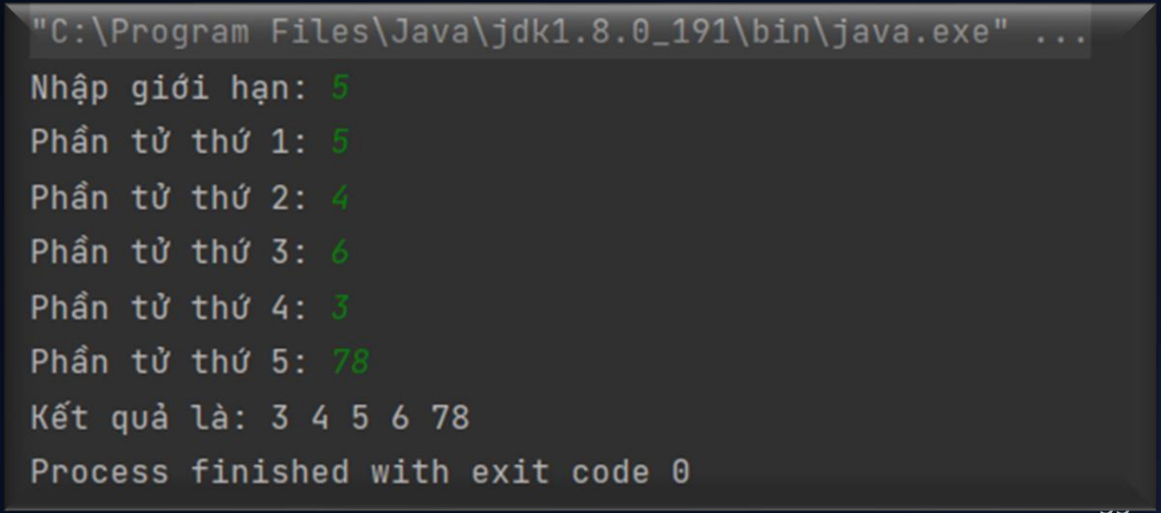


```java
public class Main{
    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
        System.out.print("Nhập giới hạn: ");
        int n = sc.nextInt();
        int [] arr = new int[n];

        for(int i = 0; i < n; i++){
            System.out.print("Phần tử thứ " +(i + 1)+ ": " );
            arr[i] = sc.nextInt();
        }

        for(int i = 0; i < n; i++){
            for(int j = i + 1; j < n; j++){
                if(arr[i] > arr[j]){

                    int swap = arr[i];
                    arr[i] = arr[j];
                    arr[j] = swap;

                }
            }
        }
        System.out.print("Kết quả là: ");
        for(int i = 0; i < n; i++){
            System.out.print(arr[i] + " ");
        }
```

## BUBBLE SORT - IDEA

1. Starting from the first index, compare the first and the second elements.
2. If the first element is greater than the second element, they are swapped.
3. Now, compare the second and the third elements. Swap them if they are not in order.
4. The above process goes on until the last element.

## RUN & RESULT



```
"C:\Program Files\Java\jdk1.8.0_191\bin\java.exe" ...
Nhập giới hạn: 5
Phần tử thứ 1: 5
Phần tử thứ 2: 4
Phần tử thứ 3: 6
Phần tử thứ 4: 3
Phần tử thứ 5: 78
Kết quả là: 3 4 5 6 78
Process finished with exit code 0
```

## M2 - Compare two sorting algorithms?

**Complication:**

- The main advantage of Bubble Sort is the simplicity of the algorithm.
- The space complexity for Bubble Sort is O(1) , since only a single additional memory space is needed, i.e. for the variable "swap".
- Also, the best case time complexity would be O(n), which is when the list is already sorted.

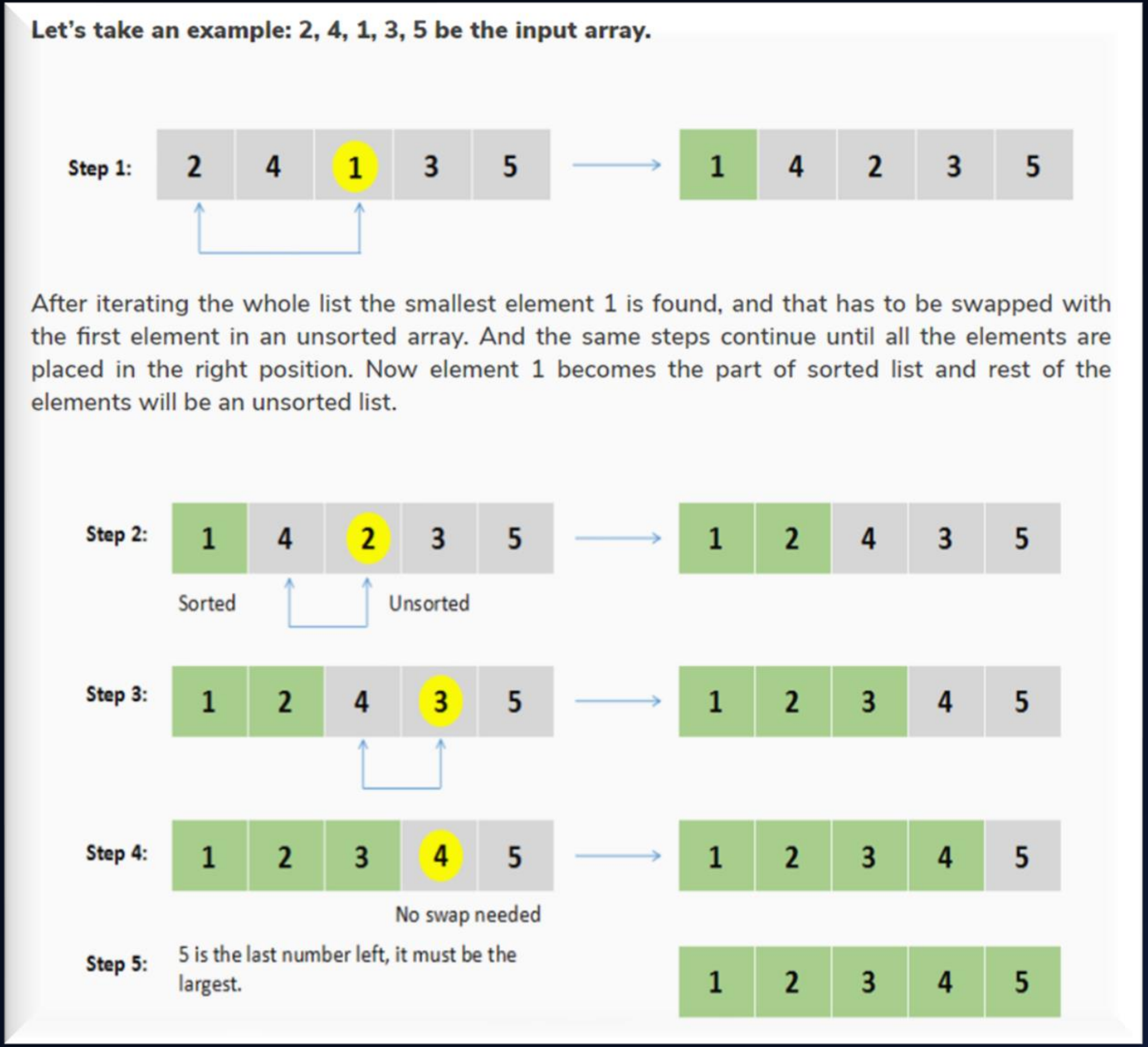
- Following is the Time and Space complexity for the Bubble Sort algorithm.

- Worst case Time complexity: O(n 2 )
- Best case time complexity]: O(n)
- Average time complexity: O(n 2 )
- Space Complexity: O(1)

M2 - Compare two sorting algorithms?

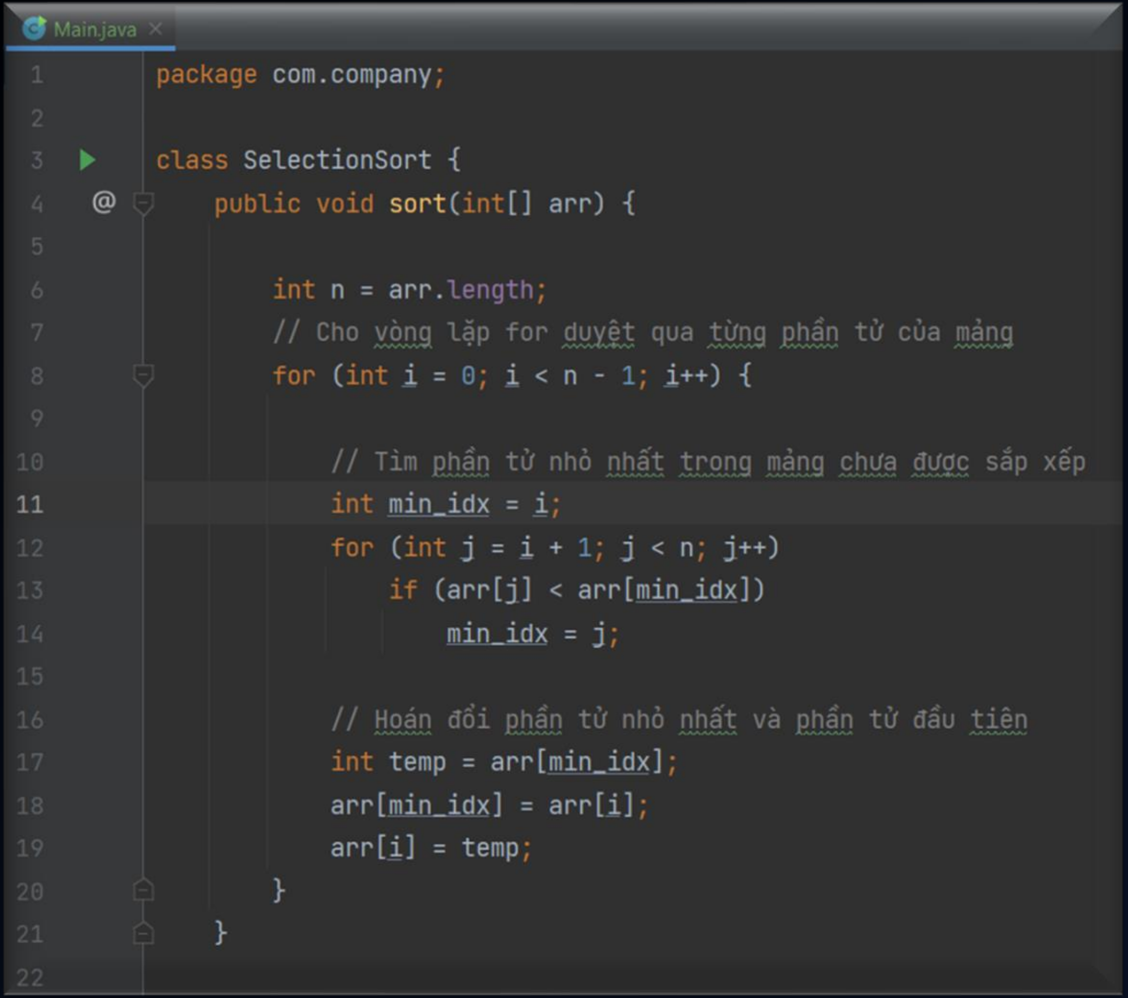
## SELECT SORT - IDEA

- Find the smallest element and put it at position 1
- Find the smallest next element put into position 2
- Find small element next put in position 3


Let's take an example: 2, 4, 1, 3, 5 be the input array.

After iterating the whole list the smallest element 1 is found, and that has to be swapped with the first element in an unsorted array. And the same steps continue until all the elements are placed in the right position. Now element 1 becomes the part of sorted list and rest of the elements will be an unsorted list.

## M2 - Compare two sorting algorithms?

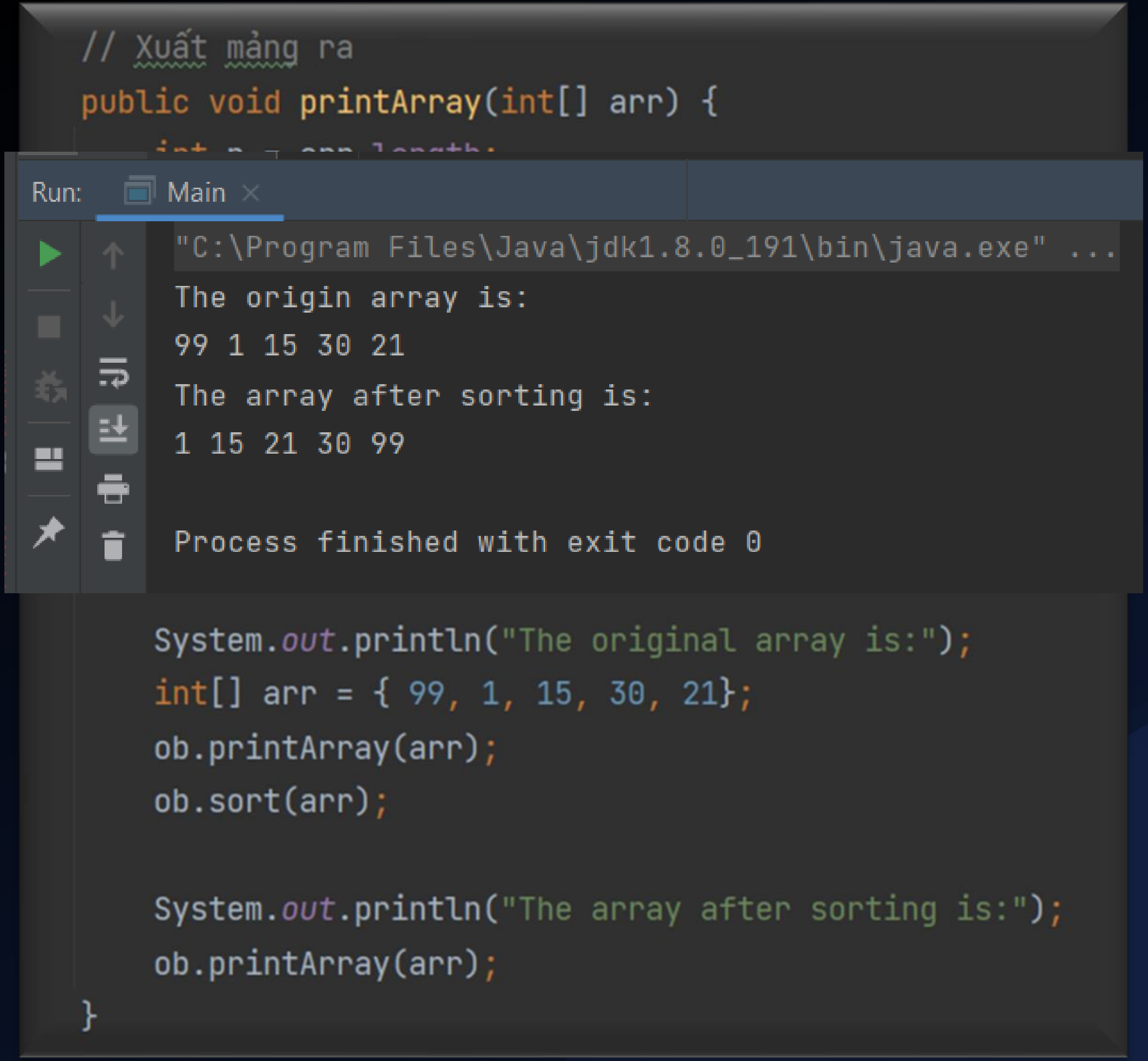


Main.java

```java
package com.company;

class SelectionSort {
    public void sort(int[] arr) {

        int n = arr.length;
        // Cho vòng lặp for duyệt qua từng phần tử của mảng
        for (int i = 0; i < n - 1; i++) {

            // Tìm phần tử nhỏ nhất trong mảng chưa được sắp xếp
            int min_idx = i;
            for (int j = i + 1; j < n; j++)
                if (arr[j] < arr[min_idx])
                    min_idx = j;

            // Hoán đổi phần tử nhỏ nhất và phần tử đầu tiên
            int temp = arr[min_idx];
            arr[min_idx] = arr[i];
            arr[i] = temp;
        }
    }

```

- Let the for loop iterate over each element of the array
- Then find the smallest element in the unsorted array
- Next we will swap the smallest and first element

## M2 - Compare two sorting algorithms?



```
// Xuất mảng ra
public void printArray(int[] arr) {
```

```
"C:\Program Files\Java\jdk1.8.0_191\bin\java.exe" ...
The origin array is:
99 1 15 30 21
The array after sorting is:
1 15 21 30 99

Process finished with exit code 0
```

```
    System.out.println("The original array is:");
    int[] arr = { 99, 1, 15, 30, 21};
    ob.printArray(arr);
    ob.sort(arr);

    System.out.println("The array after sorting is:");
    ob.printArray(arr);
}
```

- ❑ After swapping, use the print function to print the array to the screen
- ❑ In the main function, there will be 2 functions that are to print the original array and print the results after being sorted
- ❑ The previously given parts are 99, 1, 15, 30, 21 after printing it is 1, 15, 21, 30, 99

## M2 - Compare two sorting algorithms?

- ❑ Best case: 0 swap (n-1 as in the code), n_2n2/2 comparisons.
- ❑ Worst case: n – 1 swap and n_2n2/2 comparisons.
- ❑ Average case: O(n) swap and n_2n2/2 comparisons.

### Complication

Selection sort takes O(n^2)O( n2) time and O(1)O( 1 ) space.

The main time overhead comes from scanning through the array to find the next smallest item. We do pick up thing n times. The first time we will look at n elements, next time it will be n - 1 elements and so on, until we are left with only one element.

Add all that up and we have

n + (n - 1) + (n - 2) + ... +2+1
That triangle series is O(n^2).

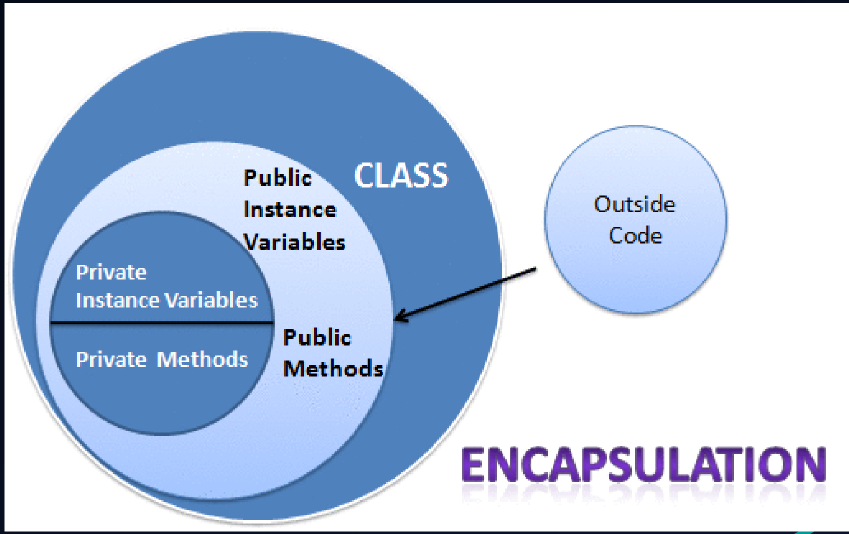
Even if the input is sorted, selection sort still involves continuously scanning all unsorted elements to find the next smallest element. So unlike insertion sort, it will still be O(n^2), even in the best case.

M3 - Explain the advantages of using encapsulation and hiding in ADT

## WHAT IS HIDE INFORMATION?

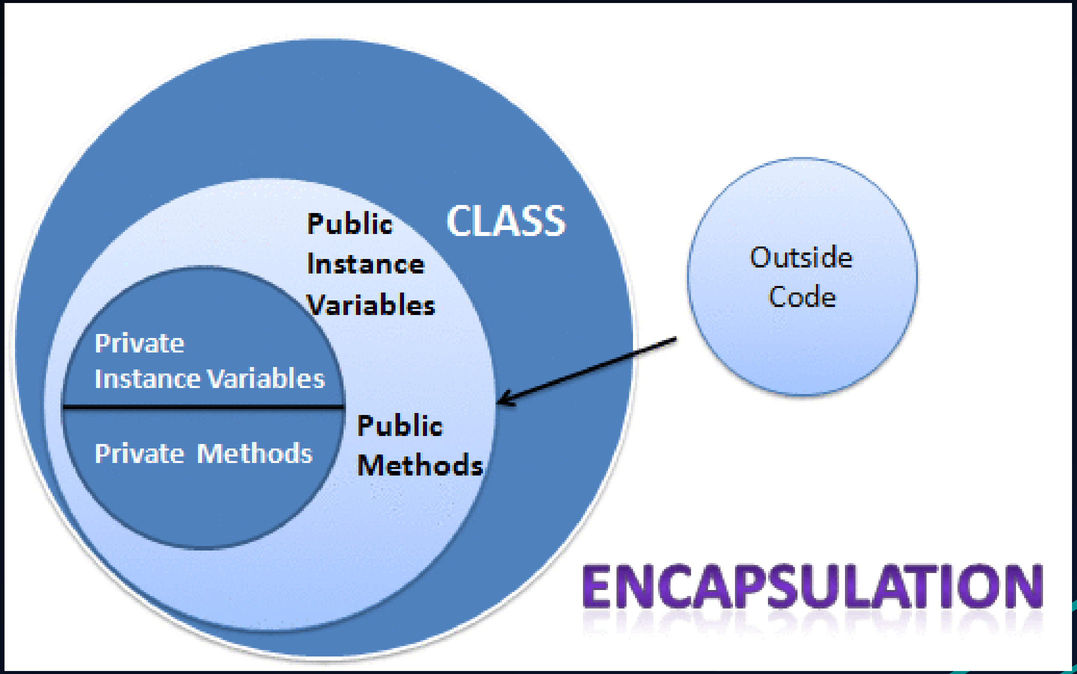
"Its interface or definition has been chosen to reveal as little as possible about its inner workings.“

"Confusion can occur when people don't distinguish between hiding information and a technique (e.g., abstraction) is used to help determine what information should be hidden."

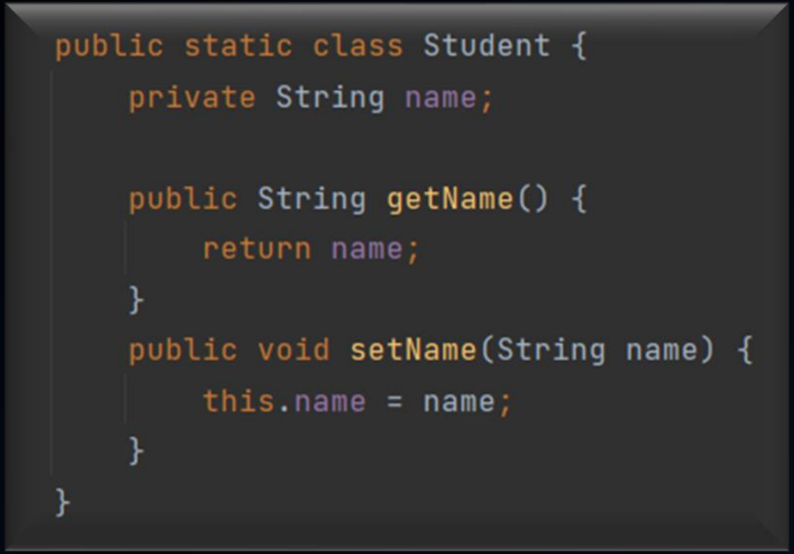


## WHAT ARE ENCAPSULATION?

**Encapsulation** in java is a technique to hide irrelevant information and display it as irrelevant. The main purpose of encapsulation in java is to reduce software development complexity.

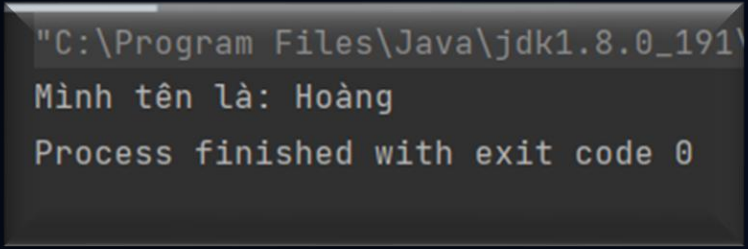


## M3 - Explain the advantages of using encapsulation and hiding in ADT

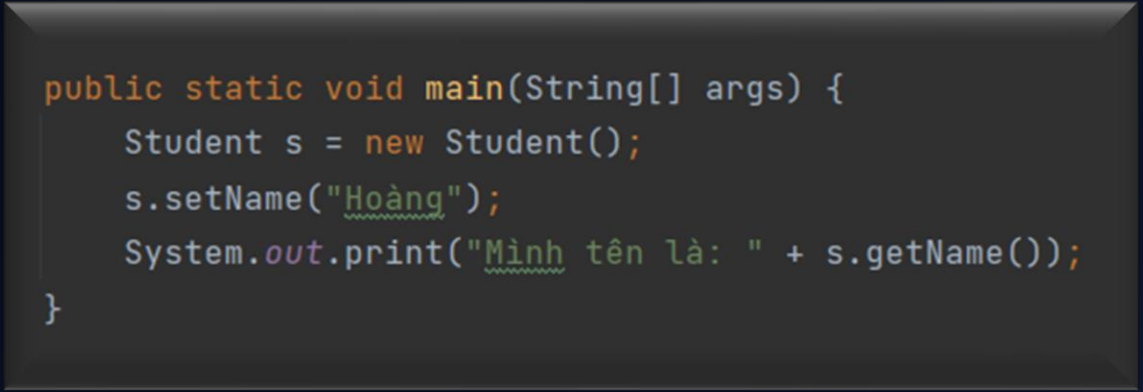


```
public static class Student {
    private String name;

    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
}
```

- I have a Student class with data type String name, scope is private
- To get the information from the Student class, we need the setter and getter methods
- In the main function, if you want to assign a value to it, setName, if you want to get that value, getName



```
"C:\Program Files\Java\jdk1.8.0_191\
Mình tên là: Hoàng
Process finished with exit code 0
```



```
public static void main(String[] args) {
    Student s = new Student();
    s.setName("Hoàng");
    System.out.print("Mình tên là: " + s.getName());
}
```

M3 - Explain the advantages of using encapsulation and hiding in ADT

## ADVANTAGES OF USING ENCAPSULATION IN ADT

- ❑ **Abstraction** and **encapsulation** are complementary concepts: abstraction focuses on the observable behavior of an object... encapsulation focuses on the implementation that leads to this behavior... encapsulation usually achieved through information hiding, which is the process of hiding all the secrets of an object that does not contribute to its essential characteristics.

- ❑ It improves the maintainability of an application.
- ❑ Provides flexibility for users to use the system very easily
- ❑ Help developers organize code better
- ❑ Makes the overall coding process easier, since you only care about what the other class does, not how it does it
- ❑ This method helps developers to be more objective and result oriented.
- ❑ The encapsulation code is quite flexible and easy to change with new code the requirements.
- ❑ Encapsulation makes unit testing easy

M3 - Explain the advantages of using encapsulation and hiding in ADT

## SUMMARY

**Abstraction**, **information hiding**, and **encapsulation** are very different, but the concepts are highly related. One could argue that abstraction is a technique that helps us determine what particular information should be. displayed and what information should be hidden. to encapsulate information in a way that hides what should be hidden and shows what is intended to be displayed.

Link: https://qastack.vn/programming/24626/abstraction-vs-information-hiding-vs-encapsulation

# REFERENCES

**[1]**
Examples Java Code Geeks. 2021. *ADT Java Tutorial*. [online] Available at: <https://examples.javacodegeeks.com/adt-java-tutorial/> [Accessed 1 July 2021].

**[2]**
Tutorialspoint.com. 2021. *Data Structure and Algorithms - Stack - Tutorialspoint*. [online] Available at: <https://www.tutorialspoint.com/data_structures_algorithms/stack_algorithm.htm> [Accessed 1 July 2021].

**[3]**
Vi.wikipedia.org. 2021. *Hàng đợi – Wikipedia tiếng Việt*. [online] Available at: <https://vi.wikipedia.org/wiki/H%C3%A0ng_%C4%91%E1%BB%A3i> [Accessed 1 July 2021].

**[4]**
VietTuts. 2021. *Bài tập Java - Các thuật toán sắp xếp trong Java - VietTuts*. [online] Available at: <https://viettuts.vn/bai-tap-java/cac-thuat-toan-sap-xep-trong-java> [Accessed 1 July 2021].

**[5]**
Programiz.com. 2021. *Selection Sort (With Code)*. [online] Available at: <https://www.programiz.com/dsa/selection-sort> [Accessed 1 July 2021].

**[6]**
2021. [online] Available at: <https://codingpearls.com/ky-thuat-lap-trinh/tim-hieu-giai-thuat-sap-xep-bubble-sort.html> [Accessed 1 July 2021].

**[7]**
Source.vn. 2021. *Thuật toán sắp xếp nào là nhanh nhất? - Source.vn*. [online] Available at: <https://source.vn/thuat-toan-sap-xep/> [Accessed 1 July 2021].

THANK YOU