

## LESSON 02

### Cấu trúc dữ liệu

### Tổng qua về kiểu dữ liệu, biến trong Java



# Agenda

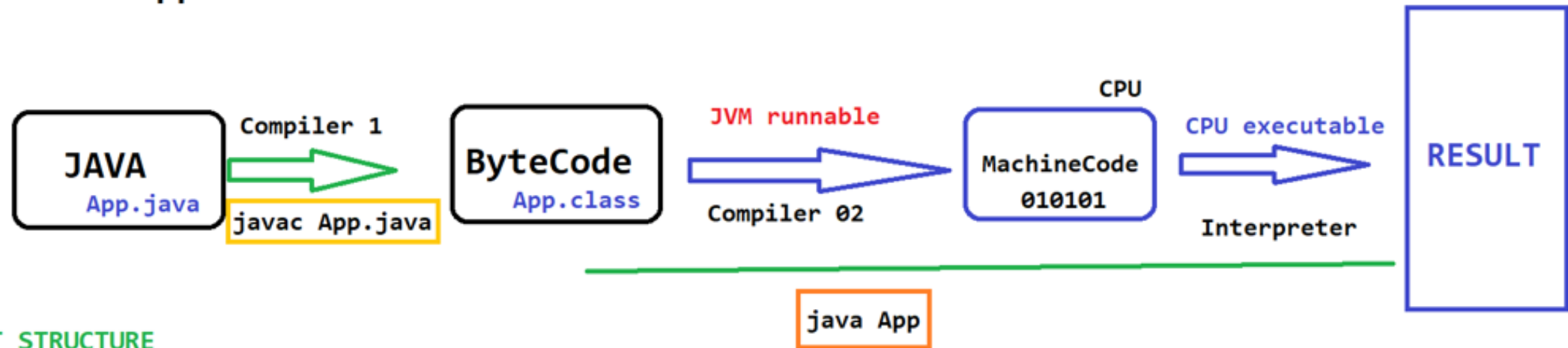
---

- 1. Quá trình thực thi
- 2. Chương trình main
- 3. Các kiểu dữ liệu cơ sở
- 4. Biến
- 5. Giới thiệu về các toán tử

# Quá trình thực thi chương trình

JDK >> OS (windows)  
+ JRE  
+ JVM: Java Virtual Machine(windows)  
>> run JAVA application

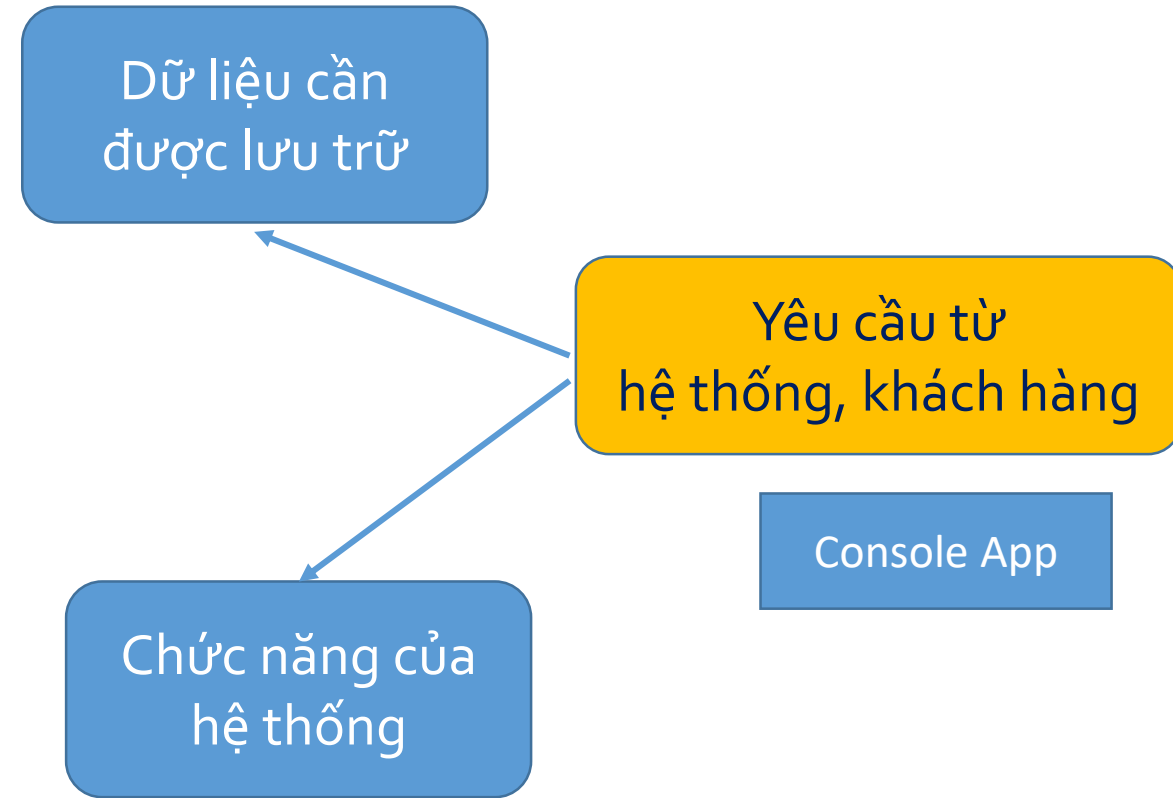
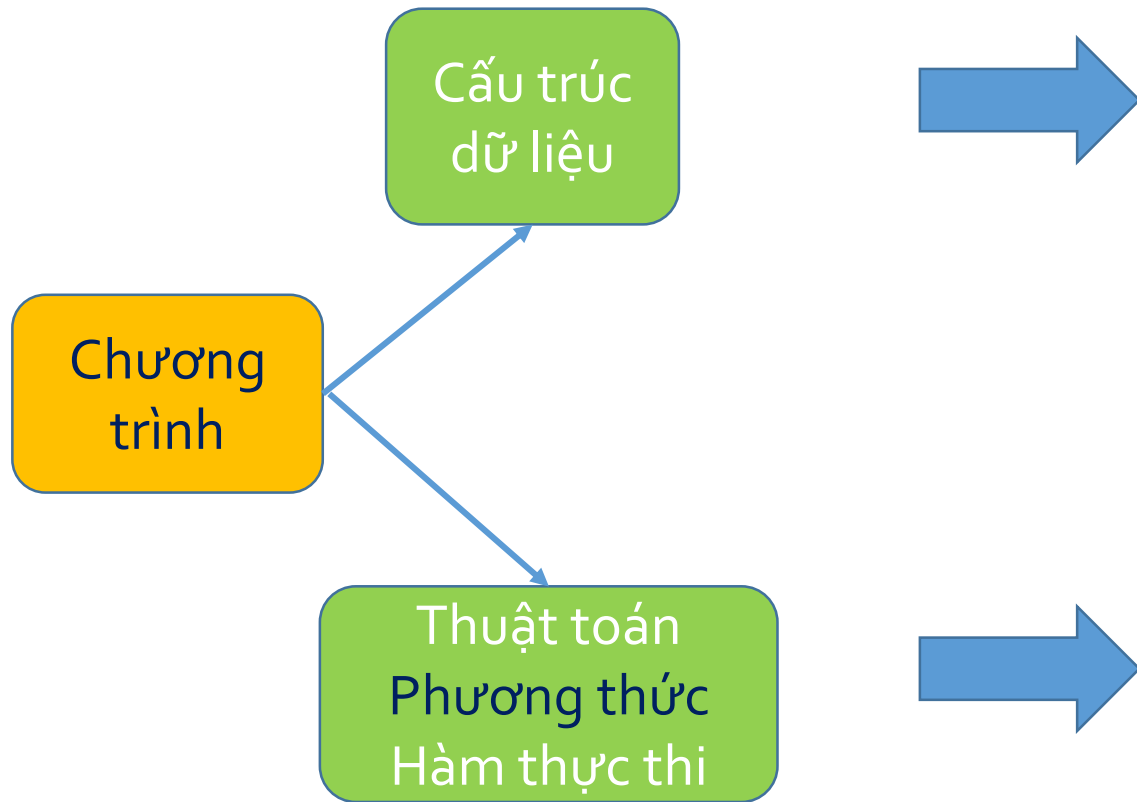
Compiler & Interpreter



JAVA PROJECT STRUCTURE

WORKSPACE >> PROJECTS >> [PACKAGES] >> FILES .JAVA

# ➤ Cấu trúc của một chương trình





# Cấu trúc của một chương trình

Khởi tạo, lưu trữ  
dữ liệu

Chương  
trình

Cấu trúc  
dữ liệu

Thuộc tính  
Biến

Thuật toán  
Phương thức  
Hàm thực thi

Hàm  
Phương thức

```
private static List<Item> getItems() {  
    List<Item> items = new ArrayList<>();  
  
    Item item1 = new Item(1, 10, "A10", 100);  
    Item item2 = new Item(2, 20, "A20", 200);  
    Item item3 = new Item(3, 30, "A30", 300);  
    Item item4 = new Item(1, 12, "A12", 120);  
  
    items.add(item1);  
    items.add(item2);  
    items.add(item3);  
    items.add(item4);  
  
    return items;  
}
```

Khai báo cấu  
trúc dữ liệu

```
public class Item {  
    private int storeId;  
    private int itemId;  
    private String name;  
    private double price;  
}
```

Tìm kiếm những mặt hàng có giá trên 100K

```
private static List<Item> filter(List<Item> items) {  
    List<Item> result = new ArrayList<>();  
    for (Item item: items) {  
        if (item.getPrice() > 100) {  
            result.add(item);  
        }  
    }  
    return result;  
}
```

Console App



# JAVA console – main method

DEMO

Java  
executable class

Phương thức main

Được gọi khi thực thi  
Exo1TestMainMethod

java Exo1TestMainMethod

```
public class Exo1TestMainMethod {  
    public static void main(String[] args) {  
        // comment: this code will not be executed by VM  
        System.out.println("Welcome to JAVA - Spring course");  
        anotherMainMethod();  
    }  
  
    public static void anotherMainMethod() {  
        System.out.println("Fake main method");  
    }  
}
```

Another main method

Sẽ được thực thi khi hàm  
official main của Java gọi đến

Access Modifier: public, private, protected

Loading time of JVM: static, non-static

Return Type: void, List<Item> - depends  
on business and reusable of method



# Kiểu dữ liệu

Kiểu dữ liệu

Biến – Tham Chiếu

Giá trị

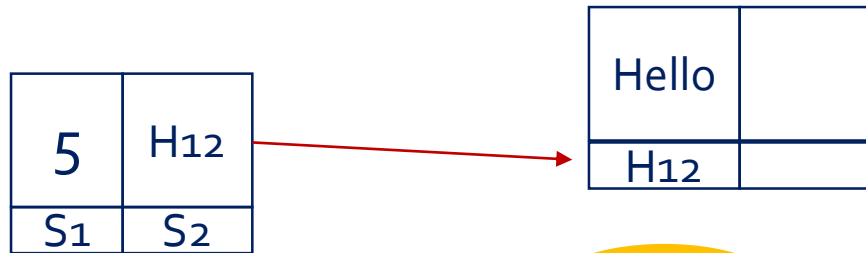
➤ Kiểu dữ liệu: Xác định dữ liệu cần lưu trữ chứa thông tin gì

- Chữ số - chuỗi ký tự - danh sách – đối tượng
- Int, float, char, Boolean, String, List<T>

```
int myNum = 5;           // Integer (whole number)
float myFloatNum = 5.99f; // Floating point number
char myLetter = 'D';     // Character
boolean myBool = true;   // Boolean
String myText = "Hello"; // String
```

➤ Biến: Tên biến, alias trỏ đến ô nhớ đang lưu trữ giá trị trong vùng nhớ

- Sẽ trả về giá trị của ô nhớ mà biến đang trỏ đến khi mình gọi biến đó
- Lưu trữ ở vùng nhớ STACK
- Khi biến có kiểu dữ liệu gì thì sẽ lưu trữ giá trị trên phạm vi của kiểu dữ liệu đó
  - Int: Số nguyên
  - Float: Số thực
  - String: Chuỗi ký tự ...
- Khi biến là kiểu dữ liệu nguyên thủy(primitive type): Giá trị trong ô nhớ(stack) chính là giá trị của biến
- Khi biến là kiểu dữ liệu đối tượng(object type): Giá trị trong ô nhớ(stack) là địa chỉ của ô nhớ(heap) mà biến đang trỏ đến



myNum myText

HEAP

STACK

static data type

dynamic data type



# Kiểu dữ liệu

## ➤ Tồn tại 2 kiểu dữ liệu trong Java

- KDL nguyên thủy – primitive type
- KDL đối tượng – object type

## ➤ Nguyên thủy

- Sử dụng để lưu trữ dữ liệu
- Giá trị được lưu trữ ở vùng nhớ STACK
- Giá trị mặc định phụ thuộc vào KDL
- Lưu trữ duy nhất một giá trị

```
public static boolean isDigit(char ch) {  
    return isDigit((int)ch);  
}
```

## ➤ Đối tượng

- Sử dụng để lưu trữ dữ liệu
- Giá trị được lưu trữ ở vùng nhớ HEAP
- Giá trị mặc định là null – chưa được khởi tạo ở HEAP
- KDL đối tượng có thể là một class, interface, abstract class ...
- Có thể tự tạo ra các KDL đối tượng như Item, Employee, CustomList
- Là wrapper, lưu trữ một hoặc nhiều thông tin, phù hợp yêu cầu thực tế
- Hỗ trợ các phương thức, giúp xử lý yêu cầu bài toán

Primitive	Object
boolean	Boolean
char	Character
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double

```
public class Item {  
    private int storeId;  
    private int itemId;  
    private String name;  
    private double price;  
}
```

```
private static List<Item> filter(List<Item> items) {  
    List<Item> result = new ArrayList<>();  
    for (Item item: items) {  
        if (item.getPrice() > 100) {  
            result.add(item);  
        }  
    }  
    return result;  
}
```

## ➤ Biến – tham chiếu được lưu trữ ở STACK





# Kiểu dữ liệu – Phạm Vi – Giá trị mặc định

Data Type	Size	Description
byte	1 byte	Stores whole numbers from -128 to 127
short	2 bytes	Stores whole numbers from -32,768 to 32,767
int	4 bytes	Stores whole numbers from -2,147,483,648 to 2,147,483,647
long	8 bytes	Stores whole numbers from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	4 bytes	Stores fractional numbers. Sufficient for storing 6 to 7 decimal digits
double	8 bytes	Stores fractional numbers. Sufficient for storing 15 decimal digits
boolean	1 bit	Stores true or false values
char	2 bytes	Stores a single character/letter or ASCII values

Data Type	Default Value (for fields)
byte	0
short	0
int	0
long	0L
float	0.0f
double	0.0d
char	'\u0000'
String (or any object)	null
boolean	false



# Khai báo biến và sử dụng

DEMO

➤ Cú pháp: [access modifier] [static] [final] datatype variablename

## ➤ Biến

- Biến cục bộ
- Biến toàn cục

## ➤ Biến cục bộ - Local variable

- Khai báo và sử dụng trong các phương thức, cấu trúc điều khiển
- Phạm vi sử dụng: BLOCK CODE {}
- Cú pháp khai báo: datatype variablename

## ➤ Biến toàn cục(thuộc tính) – Global variable(attribute)

- Khai báo và sử dụng trong các class, interface ...
- Phạm vi sử dụng: Ở bất kỳ đâu(phương thức) trong class, được dùng bên ngoài class hiện tại(phụ thuộc vào access modifier)
- Cú pháp khai báo: [access modifier] [static] [final] datatype variablename

```
public class Item {
    // attribute
    public int id;
    public String name;
    public double price;

    // constructor
    public Item(final int id, final String name, final double price) {
        this.id = id;
        this.name = name;
        this.price = price;
    }

    @Override
    public String toString() {
        // automatically called by JVM while printing Item object
        return id + ", " + name + ", " + price;
    }
}
```

```
public class Ex03 {
    public static void main(String[] args) {
        Item phone = new Item(1, "Iphone X", 2000);
        Item usb = new Item(2, "USB type C x20", 10);
        System.out.println(phone);
        System.out.println(usb);
    }
}
```

```
public class Ex02 {
    // global variable: attribute
    private static int year = 2021;

    // function: method
    public static void main(String[] args) {
        // local variable
        int year = 1998;
        System.out.println("year: " + year);

        // local variable
        int age = 20;
        int salary = 20;
        age = 21;
        salary = 22;

        // assign salary value to age
        age = salary;
    }
}
```



# Quy tắc đặt tên biến

- Gồm các ký tự chữ, số, dấu gạch dưới \_ và \$
- Bắt đầu bằng ký tự chữ, phân biệt hoa thường
- Không được trùng với từ khóa của Java: int, static, final, protected ...
- Quy tắc đặt tên: Camel case naming convention
  - Một chữ cái: i, j
  - Một từ: student, list, array, employee, salary
  - Nhiều từ: studentId, salesPrice, totalOfItems
  - Viết thường chữ cái đầu tiên để phân biệt với tên class
  - **Tên class:** tương tự với tên biến – chữ cái đầu tiên viết hoa
- Đặt tên biến theo chức năng, giá trị mà biến đó lưu trữ
- Hạn chế đặt tên biến vô nghĩa



# Biến là hằng số

- Lưu trữ những giá trị không đổi
- Thêm từ khóa **final** để được hằng số
- Biến là hằng số với
  - KDL nguyên thủy
  - KDL đối tượng
- KDL nguyên thủy
  - Giá trị lưu trữ ở STACK – hằng số thì sẽ không thể cập nhật giá trị
- KDL đối tượng
  - Giá trị lưu ở HEAP – hằng số vẫn có thể cập nhật giá trị của đối tượng
  - Địa chỉ sẽ được lưu trữ ở STACK – không thể cập nhật địa chỉ mà biến đang trỏ đến
- BIẾN LÀ HẰNG SỐ - KHÔNG THỂ THAY ĐỔI GIÁ TRỊ Ở STACK

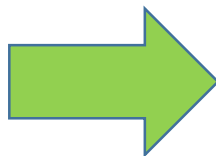


# Cấu trúc dữ liệu – Thực hành

DEMO

- Exo4: KDL nguyên thủy - Giải thích các lưu trữ và hoạt động
  - Cập nhật giá trị trong cùng 1 phương thức (main)
  - Cập nhật giá trị thông qua tham số trong 1 phương thức khác

```
public class Ex04PExecution {  
|.... public static void main(String[] args) {  
|....     int value = 14;  
|....     int anotherValue = 20;  
|....     System.out.println("Value1: " + value);  
  
|....     // modify in same method  
|....     value = 20;  
|....     System.out.println("Value2: " + value);  
  
|....     // modify via another method  
|....     modify(value);  
|....     System.out.println("Value2: " + value);  
|.... }  
  
|.... private static void modify(int input) {  
|....     input = 999;  
|.... }  
}
```



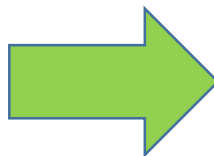


# Cấu trúc dữ liệu – Thực hành

DEMO

- Exo5: KDL đối tượng - Giải thích các lưu trữ và hoạt động
  - Cập nhật giá trị trong cùng 1 phương thức (main)
  - Cập nhật giá trị thông qua tham số trong 1 phương thức khác
  - Giới thiệu: System.identityHashCode

```
public class Ex050Execution {  
    .....public static void main(String[] args) {  
        .....String text = "start";  
        .....String anotherText = "end";  
        .....System.out.println("Value1: " + text);  
  
        .....// modify in same method  
        .....text = "in-progress";  
        .....System.out.println("Value2: " + text);  
  
        .....// modify via another method  
        .....modify(text);  
        .....System.out.println("Value2: " + text);  
    .....}  
  
    .....private static void modify(String input) {  
        .....input = "write something";  
    .....}  
}
```



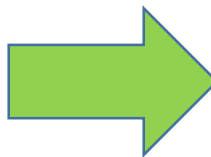


# Cấu trúc dữ liệu – Thực hành

DEMO

- Exo6: Giải thích cơ chế lưu trữ của các KDL có sẵn trong JAVA
  - Integer, String, Double, Float, Long
  - Khái niệm constant pool

```
public class Ex06 {  
    public static void main(String[] args) {  
        // constant pool  
        Integer age = 22;  
        Integer yearOfBirth = 2000;  
        PrintUtils.hash("age", age);  
        PrintUtils.hash("yearOfBirth", yearOfBirth);  
  
        // normal heap - never used  
        Integer salary = new Integer(30);  
        Integer exp = new Integer(3);  
        PrintUtils.hash("salary", salary);  
        PrintUtils.hash("exp", exp);  
    }  
}
```



```
// normal heap - never used  
Integer salary = new Integer(30);  
Integer exp = new Integer(3);  
PrintUtils.hash("salary", salary);  
PrintUtils.hash("exp", exp);
```

Unnecessary boxing 'new Integer(30)'

Remove boxing Alt+Shift+Enter More actions... Alt+Enter



# Cấu trúc dữ liệu – Thực hành

---

- Exo7: Giải thích cơ chế của final
  - Primitive Type
  - Object Type

DEMO





# Cấu trúc dữ liệu – Thực hành

- Exo8: Viết chương trình hoán đổi giá trị - swap 2 phần tử số nguyên
  - int
  - Integer
  - CustomInteger

DEMO



# Cấu trúc dữ liệu – Thực hành

DEMO

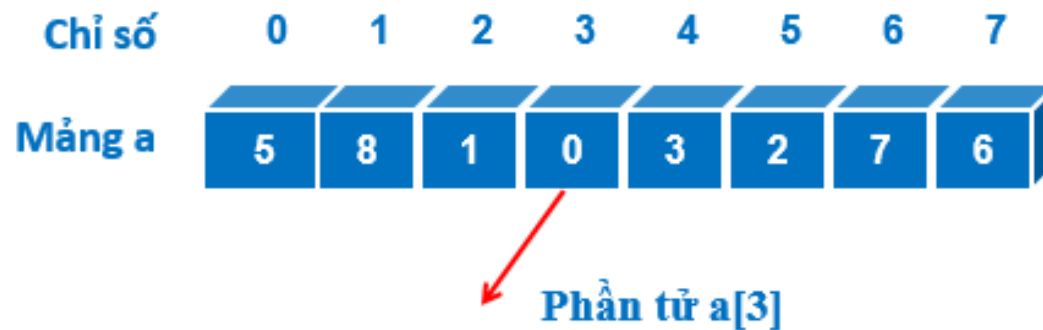
- Exog: Khi nào sử dụng KDL nguyên thủy, đối tượng
  - Phân biệt
  - Cho ví dụ
  - Kiểm tra 1 ký tự, chuỗi nhập vào có phải là số hay không



# Kiểu dữ liệu – Mảng

## Khái niệm

- Mảng là một **cấu trúc dữ liệu**, lưu trữ một **tập các phần tử cùng kiểu dữ liệu** và có kích thước cố định: **fixed-size sequential**.
- Thay vì khai báo individual variables var0, var1, ..., var99. Ta có thể khai báo mảng có với các phần tử var[0], var[1], ..., var[99].
- Mỗi phần tử trong mảng được truy xuất thông qua **chỉ số, giá trị**





# Kiểu dữ liệu – Mảng

## Khai báo

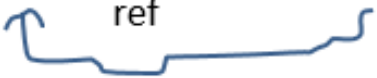
### ❖ Khai báo: Array Variables

```
data_type arr[];    // works but not preferred way.  
data_type[] arr;    // preferred way.
```

```
double elements[];    // works but not preferred way.  
double[] elements;    // preferred way  
String[] elements;    // preferred way.  
Employee[] employees; // preferred way.
```

### ❖ Khởi tạo: Array Variables

```
dataType[] arrayRefVar = new dataType[arraySize];
```



#### Cú pháp:

```
dataType[] arrayRefVar = new dataType[arraySize];  
dataType[] arrayRefVar = {value0, value1, ..., valuek};
```

#### **Example:**

```
String[] elements = new String[4];  
String[] elements = {"a", "b", "c"};  
String[] elements = new String[] {"a", "b"};
```



# Kiểu dữ liệu – Mảng – Thực hành

DEMO

- Ex09: Khai báo và khởi tạo mảng số nguyên int
  - Mảng rỗng
  - Mảng có giá trị
  - Truy cập thông qua chỉ số 0 – length-1
  - Duyệt theo 2 cách
    - ForIndex
    - ForEach
- Ex10: Viết hàm tìm các phần tử lẻ trong mảng
- Ex11: Khai báo và khởi tạo mảng đối tượng Item
  - Tương tự



## Kiểu dữ liệu - Enumeration

---

- Là KDL đối tượng đặc biệt sử dụng để định nghĩa một tập các các giá trị không đổi.
- Enum có thể chứa các thuộc tính, phương thức và hàm khởi tạo (same as class)
- Hàm khởi tạo của enum mặc định là private và không được phép là public
- Có thể được khai báo bên trong, ngoài một lớp



# Kiểu dữ liệu - Enumeration

- Ex11: Viết chương trình tìm vị trí tương đối của một điểm A so với đường tròn tâm O bán kính R

DEMO

- A(5, 7)
- O(0, 0) R = 6
- Tạo enum lưu trữ 3 vị trí tương đối của 1 điểm so với đường tròn
  - INSIDE
  - ONSIDE
  - OUTSIDE
- Ví dụ khác ...

```
public enum CircleUnit {  
    INSIDE("Trong đường tròn"),  
    ONSIDE("Trên đường tròn"),  
    OUTSIDE("Ngoài đường tròn");  
  
    String value;  
  
    private CircleUnit(String value) {  
        this.value = value;  
    }  
  
    public String getValue() {  
        return value;  
    }  
}
```

# Toán tử trong lập trình





# Toán tử - Operator

DEMO

Ký hiệu	Mô tả	Ví dụ
=	Gán toán tử hạng hai cho toán tử hạng nhất	a = 1
+=	Cộng hoặc nối chuỗi toán hạng sau vào toán hạng đầu và gán kết quả cho toán hạng đầu	a += 1 a = a + 1
-=	Trừ toán hạng sau khỏi toán hạng đầu và gán kết quả cho toán hạng đầu.	a -= 1
*=	Nhân toán hạng sau vào toán hạng đầu và gán kết quả cho toán hạng đầu	a *= 2
/=	Chia toán hạng sau cho toán hạng đầu và gán kết quả cho toán hạng đầu	a /= 2

```
package operator;

public class Ex01 {
    public static void main(String[] args) {
        int a = 5;
        a += 5; // a = a + 5
        System.out.println("Value: " + a);
    }
}
```

Ex01 ×  
"C:\Program Files\Java\jdk1.8.0\_152\bin\java.exe" ...  
Value: 10



# Toán tử - Operator

DEMO

## ➤ Toán tử số học

Toán tử	Mô tả
+	Cộng
-	Trừ
*	Nhân
/	Chia lấy phần nguyên
%	Chia lấy phần dư

```
public class Ex02 {  
    public static void main(String[] args) {  
        int div1 = 7 / 2;  
        int div2 = 7 % 2;  
        System.out.println("div1: " + div1);  
        System.out.println("div2: " + div2);  
    }  
}
```

Ex02 x

```
"C:\Program Files\Java\jdk1.8.0_152\bin\java.exe"  
div1: 3  
div2: 1
```

## ➤ Toán tử một ngôi

Toán tử	Mô tả
++	Tăng giá trị lên 1
--	Giảm giá trị đi 1
!	Phép toán phủ định

```
int n = 10;  
for (int i = 1; i <= n; i++) {  
    // isEven(i) <==> isEven(i) == true  
    if (isEven(i)) {  
        System.out.println(i + " is even number");  
    }  
  
    // !isEven(i) <==> isEven(i) == false  
    if (!isEven(i)) {  
        System.out.println(i + " is odd number");  
    }  
}
```



# Toán tử - Operator

DEMO

## ➤ Toán tử so sánh – trả về true false

Toán tử	Mô tả
==	So sánh bằng
!=	So sánh không bằng
>	So sánh lớn hơn
>=	So sánh lớn hơn hoặc bằng
<	So sánh nhỏ hơn
<=	So sánh nhỏ hơn hoặc bằng

## ➤ Toán tử kết hợp

Toán tử	Mô tả
==	So sánh bằng
!=	So sánh không bằng

## ➤ Toán tử ba ngôi

? :	Expression ? A : B
-----	--------------------

## Toán tử - Operator – Thực hành

DEMO

➤ Exo4: Thực hiện test nhanh trên giấy. Đoán kết quả của x,y,z sau khi thực

Hiện phép toán sau đây

Input: int x = 5, y = 2, z = 3

Phép toán

$y += y + x++ + z++ + ++z + ++x - y-- + z$

Yêu cầu

x = ?, y = ?, z = ?

Quy tắc

Trong một biểu thức (TRÁI → PHẢI)

Nếu có TRÁI = PHẢI. Thực hiện vế PHẢI trước rồi gán cho vế TRÁI

LAST IN FIRST OUT

## Toán tử - Operator – Thực hành

DEMO

```
int i = 2 ;
if(++i > 2 && i++ > 2){
    i++;
}

if(i++ > 4 || ++i > 5){
    System.out.println(i);
}

if(i++ < 4 || ++i > 5) {
    System.out.println(i);
}
```

```
int x = 0;
int y = 0;
for (int z = 0; z < 5; z++) {
    if ((++x > 2) && (++y > 2)) {
        x++;
    }
}
System.out.println(x + " " + y);
```



# Các lệnh nhập xuất

---

- Là thao tác người dùng nhập liệu đưa input vào cho chương trình xử lý
- Đầu vào có thể là
  - Nhập liệu từ console – bàn phím – console app
  - Nhập liệu từ giao diện ứng dụng
  - Nhập liệu từ giao diện web



# Các lệnh nhập xuất

DEMO

- Nhập xuất từ bàn phím – console app trong Java
- Sử dụng thư viện class Scanner xuất hiện từ JDK 1.5
- Có thể phân biệt ký tự nhập vào thuộc kiểu dữ liệu gì và convert tương ứng

```
String text = ip.nextLine();  
int number = ip.nextInt();  
double price = ip.nextDouble();
```



# Các lệnh nhập xuất

DEMO

- Hiện tượng trôi lệnh xảy ra khi
  - Trước: Nhập dữ liệu không phải chuỗi
  - Ký tự enter sẽ lưu trong bộ đệm và hiểu cho lần enter tiếp theo của String
  - Sau: Nhập dữ liệu chuỗi

```
public class Ex02 {  
    public static void main(String[] args) {  
        Scanner ip = new Scanner(System.in);  
  
        System.out.print("Enter your first name: ");  
        String firstName = ip.nextLine();  
  
        System.out.print("Enter your last name: ");  
        String lastName = ip.nextLine();  
  
        System.out.print("Enter your age: ");  
        int age = ip.nextInt();  
  
        // issue, shuffle with age order  
        System.out.println("Enter your email: ");  
        String email = ip.nextLine();  
  
        System.out.println("=====");  
  
        System.out.println("firstName: " + firstName);  
        System.out.println("lastName: " + lastName);  
        System.out.println("age: " + age);  
        System.out.println("email: " + email);  
    }  
}
```

```
Enter your first name: Ade  
Enter your last name: Lane  
Enter your age: 28  
Enter your email:  
=====  
firstName: Ade  
lastName: Lane  
age: 28  
email:
```





# Random

---

- Tạo ra giá trị ngẫu nhiên thay thế cho nhập liệu
- Tên gọi khác fake input, mock data
- Thường dùng cho dữ liệu test, kiểm thử
- Random
  - Số nguyên ngẫu nhiên
  - Random lấy ngẫu nhiên các phần tử trong danh sách



# Random – Thực hành

---

DEMO

- Exo3: Tạo số ngẫu nhiên
- Exo4: Tạo số ngẫu nhiên trong phạm vi từ a đến b
- Exo5: Random sinh viên ngẫu nhiên trong danh sách để nộp bài tập

# Cấu trúc điều kiện – vòng lặp



# Cấu trúc điều kiện

---

- **Wikipedia:** [https://vi.wikipedia.org/wiki/%C4%90i%E1%BB%81u\\_ki%E1%BB%87n\\_\(%E1%BA%ADp\\_tr%C3%ACnh\\_m%C3%A1y\\_t%C3%ADnh\)](https://vi.wikipedia.org/wiki/%C4%90i%E1%BB%81u_ki%E1%BB%87n_(%E1%BA%ADp_tr%C3%ACnh_m%C3%A1y_t%C3%ADnh))
- Sử dụng để xác định, kiểm tra các điều kiện thỏa mãn yêu cầu bài toán
- Ví dụ: Cho danh sách các cựu sinh viên tại khoa CNTT
- Tìm các sinh viên hiện **đang làm việc tại Đà Nẵng**
- Tìm các sinh viên đang là **quản lý** tại các công ty trên **địa bàn Đà Nẵng**
- Điều kiện sẽ là một biểu thức có kết quả đúng, sai – true, false



# Cấu trúc điều kiện

Khởi tạo, lưu trữ  
dữ liệu

Chương  
trình

Cấu trúc  
dữ liệu

Thuộc tính  
Biến

Thuật toán  
Phương thức  
Hàm thực thi

Hàm  
Phương thức

```
private static List<Item> getItems() {  
    List<Item> items = new ArrayList<>();  
  
    Item item1 = new Item(1, 10, "A10", 100);  
    Item item2 = new Item(2, 20, "A20", 200);  
    Item item3 = new Item(3, 30, "A30", 300);  
    Item item4 = new Item(1, 12, "A12", 120);  
  
    items.add(item1);  
    items.add(item2);  
    items.add(item3);  
    items.add(item4);  
  
    return items;  
}
```

Khai báo cấu  
trúc dữ liệu

```
public class Item {  
    private int storeId;  
    private int itemId;  
    private String name;  
    private double price;  
}
```

Tìm kiếm những mặt hàng có giá trên 100K

```
private static List<Item> filter(List<Item> items) {  
    List<Item> result = new ArrayList<>();  
    for (Item item: items) {  
        if (item.getPrice() > 100) {  
            result.add(item);  
        }  
    }  
    return result;  
}
```

Console App



# Cấu trúc điều kiện

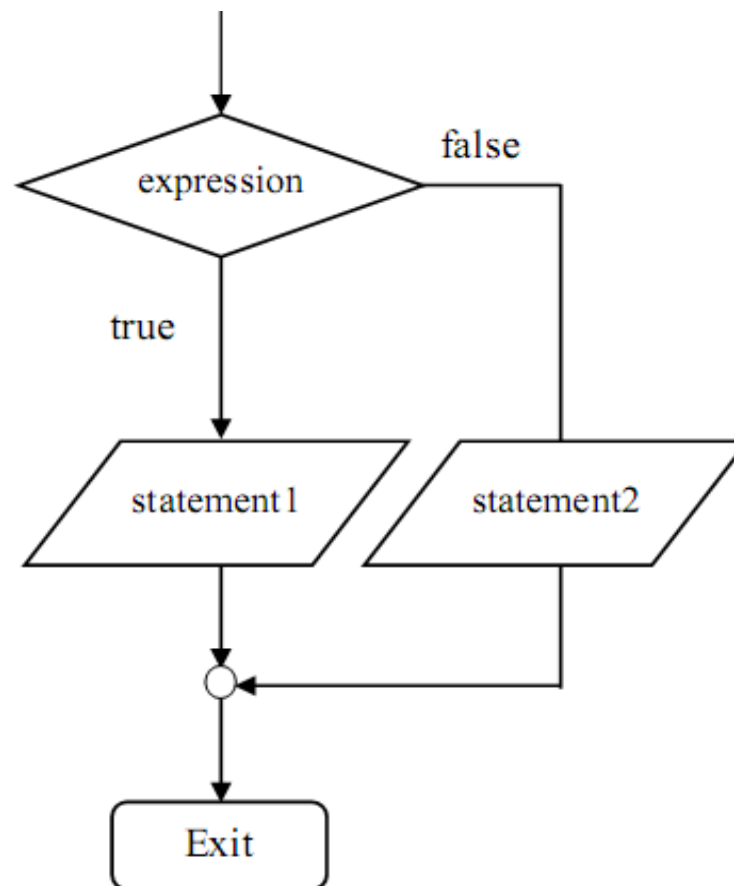
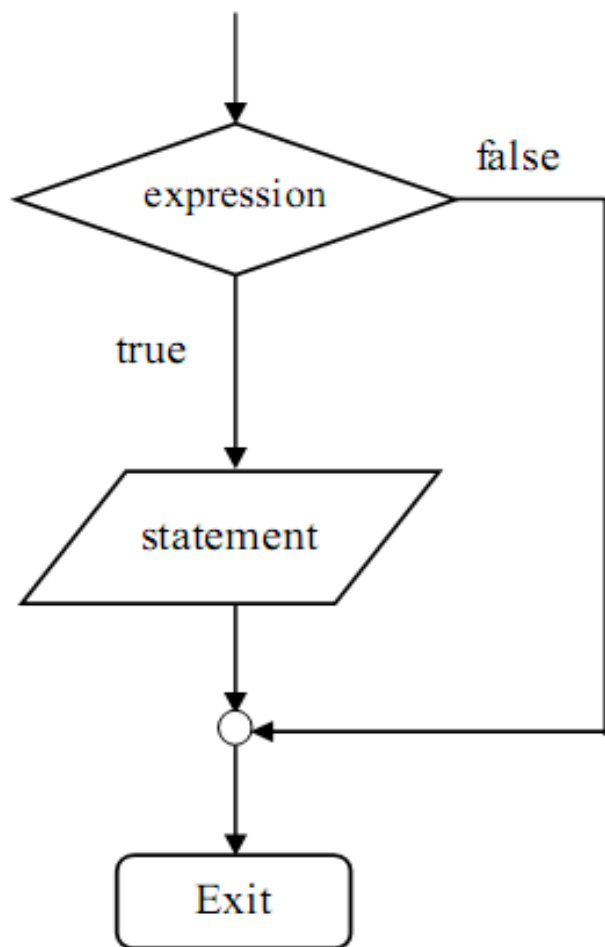
---

- Cú pháp cấu trúc điều kiện trong lập trình
  - IF ELSE
  - SWITCH CASE
  - Toán tử 3 ngôi



# Cấu trúc if else

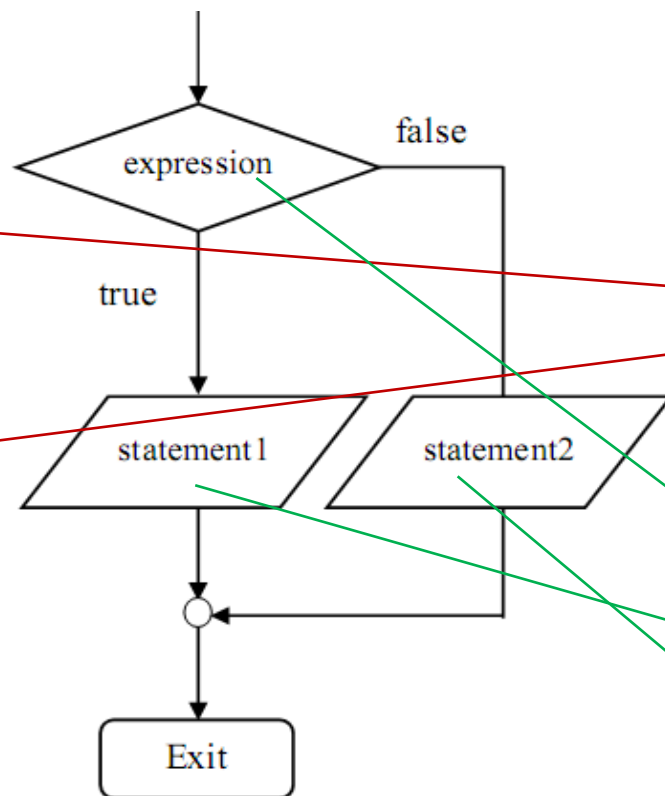
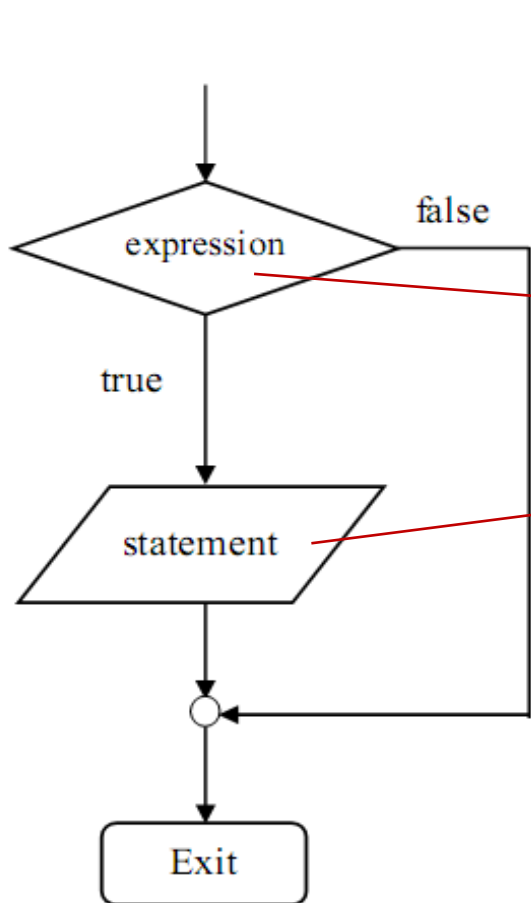
## ➤ Lưu đồ hoạt động





# Cấu trúc if else

## ➤ Lưu đồ hoạt động



```
public static void main(String[] args) {  
    int number = 22;  
    String text = "admin123";  
    // Kiểm tra 1 số có phải là bội của 3 không.  
    // Nếu phải thì in ra thông tin  
    int p3 = number % 3;  
    boolean isPower = (number % 3 == 0);  
    if (isPower) {  
        System.out.println(number + " is power of 3");  
    }  
}
```

```
// Kiểm tra 1 chuỗi có nhiều hơn 5 ký tự không  
// Nếu thỏa mãn in ra thông tin  
// Nếu không thỏa mãn báo lỗi không hợp lệ  
if (text.length() > 5) {  
    System.out.println(text + " passed the validation");  
} else {  
    System.out.println(text + " is invalid - (Length at least 5 letters)");  
}
```





## Cấu trúc if else – Thực hành

---

- Exo2: Cho ba số nguyên a, b, c. Viết chương trình tìm số lớn nhất, nhỏ nhất và số còn lại
- Exo3: Sử dụng toán tử ba ngôi. Kiểm tra một số nhập vào là chẵn hay lẻ



## Cấu trúc switch case

---

- Sử dụng khi bài toán có quá nhiều điều kiện so sánh **bằng nhau**
- Giúp code đẹp hơn so với if else else ... if ... else



# Cấu trúc switch case

---

## ➤ Lưu đồ hoạt động

```
switch (<biến cần kiểm tra>) {  
    case <giá trị 1>:  
        <công việc 1>;  
        break;  
    case <giá trị 2>:  
        <công việc 2>;  
        break;  
    default:  
        Thực hiện khi không thỏa mãn các điều kiện trên  
        break;  
}
```



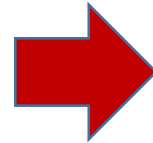
# Cấu trúc switch case

```
int month = 2;

if (month == 1) {
    System.out.println("Tháng " + month + " có 31 ngày");
} else if (month == 2) {
    // Giả sử không phải năm nhuận
    System.out.println("Tháng " + month + " có 28 ngày");
} else if (month == 3) {
    System.out.println("Tháng " + month + " có 31 ngày");
} else if (month == 4) {
    System.out.println("Tháng " + month + " có 30 ngày");
}

// .....
// .....

else {
    System.out.println("Dữ liệu không hợp lệ");
}
```



```
public static void main(String[] args) {
    int month = 2;
    switch (month) {
        case 1:
            System.out.println("Tháng " + month + " có 31 ngày");
            break;
        case 2:
            System.out.println("Tháng " + month + " có 28 ngày");
            break;
        case 3:
            System.out.println("Tháng " + month + " có 31 ngày");
            break;
        case 4:
            System.out.println("Tháng " + month + " có 30 ngày");
            break;
        default:
            System.out.println("Dữ liệu không hợp lệ");
    }
}
```



# Cấu trúc vòng lặp

---

- Wikipedia: [https://en.wikipedia.org/wiki/For\\_loop](https://en.wikipedia.org/wiki/For_loop)
- Sử dụng để xác duyệt qua danh sách các phần tử trong hệ thống
- Ví dụ: Cho danh sách các số nguyên. Tìm những số là bội của 5
- Bước 1: **Duyệt** từng phần tử trong danh sách
- Bước 2: Kiểm tra nếu số đó chia hết cho 5 thì là bội của 5



# Cấu trúc vòng lặp

---

- Cú pháp duyệt trong lập trình
- Cách 1: for index - duyệt theo chỉ số
- Cách 1: for each – duyệt theo giá trị
- Cách 2: while
- Cách 3: do while
- Cách 4: iterate



# Cấu trúc for - index

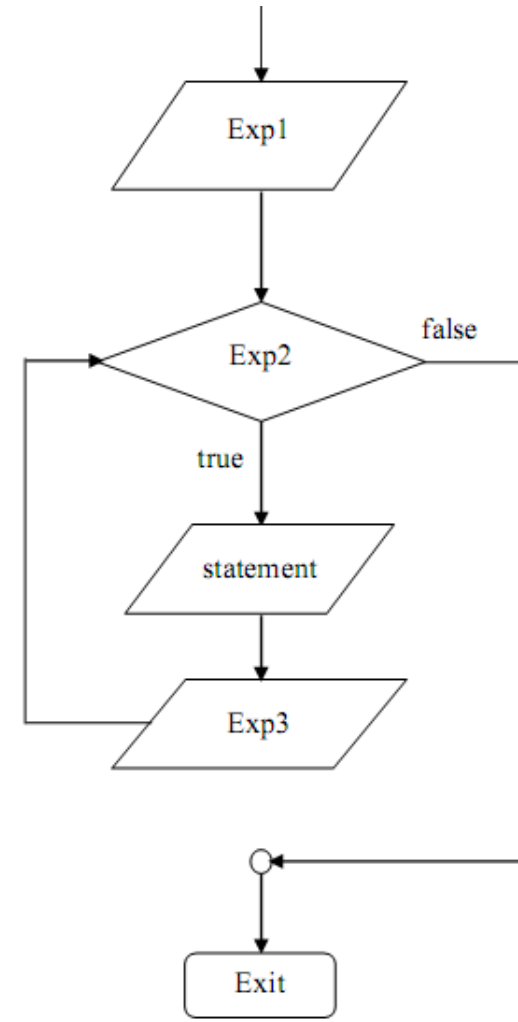
## ➤ Cú pháp:

```
for (Exp1; Exp2; Exp3) {  
    // statement;  
}
```

## ➤ Giải thích

- Exp1: là biểu thức khởi tạo
- Exp2: là biểu thức điều kiện
- Exp3: là biểu thức điều khiển lặp

```
public static void main(String[] args) {  
    for (int i = 0; i < 10; i++) {  
        // todo: add your code here  
        System.out.println(i);  
    }  
}
```





# Cấu trúc while

## ➤ Cú pháp:

```
while(expression) {  
    statement;  
}
```

## ➤ Giải thích

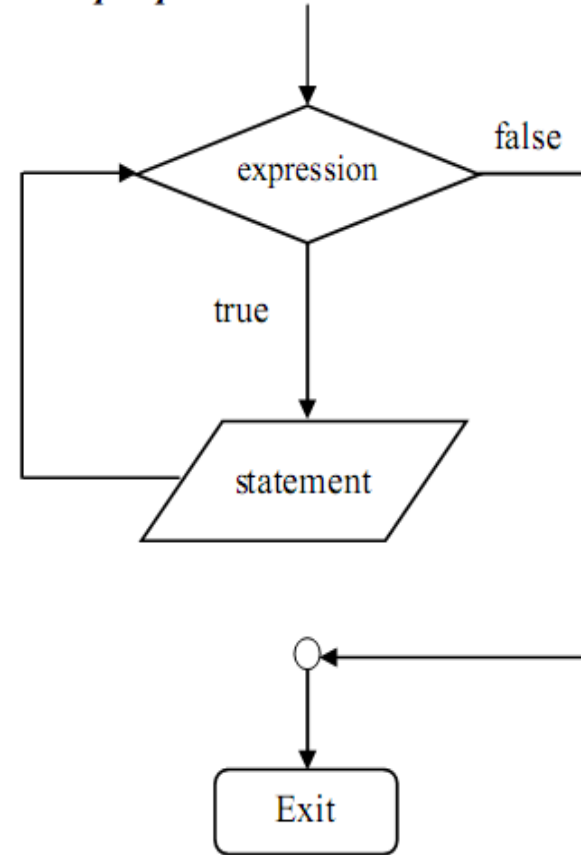
B1: Thực hiện kiểm tra expression

B2: Nếu kết quả là **true** thì statement sẽ được thực thi và quay lại B1

-----  
Nếu kết quả là **false** thì thoát khỏi vòng lặp while.

```
public static void main(String[] args) {  
    ... // Exp01  
    int i = 0;  
    while (i < 10) // Exp02  
    {  
        ... // todo: add your code here  
        System.out.println(i);  
        i++; // Exp03  
    }  
}
```

*Lưu đồ cú pháp:*







# Cấu trúc do while

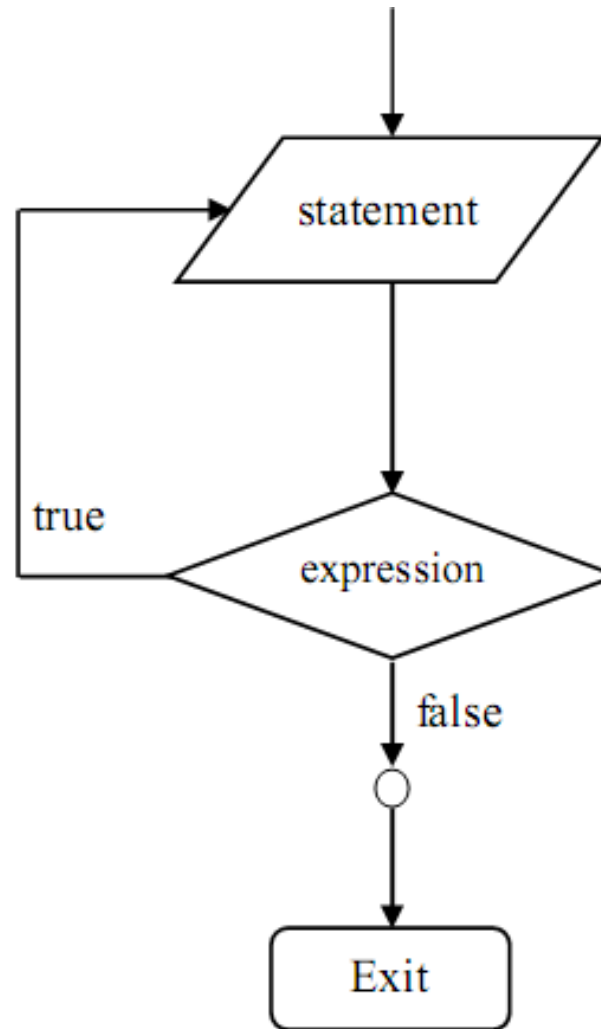
## ➤ Cú pháp:

```
do {  
    statement;  
} while(expression);
```

## ➤ Giải thích:

- B1: Thực hiện |statement|
- B2: Kiểm tra expression.
- Nếu expression là true thì quay lại B1
- Nếu expression là false thì thoát khỏi vòng lặp.

```
// Exp01  
int i = 0;  
do {  
    // todo: add your code here  
    System.out.println(i);  
    i++; // Exp03  
} while (i < 10); // Exp02
```





# Cấu trúc break continue

- Thoát khỏi vòng lặp: break
- Kết thúc sớm vòng lặp hiện tại: continue

```
public static void main(String[] args) {  
    for (int i = 1; i <= 10; i++) {  
        if (i == 4) {  
            continue;  
        }  
        if (i == 8) {  
            break;  
        }  
        System.out.println(i);  
    }  
}
```



```
"C:\Program Files\Java\jdk1.8.0_152\bin\java.exe"  
1 2 3 5 6 7  
Process finished with exit code 0
```



# Thực hành

How to attack an account with a basic password ?

-----

E.g:

+ username: admin

+ password: 259

-----

Random string value until get the correct password ?

```
019 586 115 047 196 489 468 784 613 442 429 183 621
972 154 658 958 050 659 955 855 749 704 787 680 925
133 543 737 393 251 818 549 570 734 735 518 136 726
128 575 235 063 980 047 582 731 902 154 388 552 654
802 276 953 186 162 323 770 326 623 987 563 801 184
987 689 296 478 042 849 741 832 709 289 047 928 052
669 558 791 177 910 469 157 764 230 973 889 541 923
120 890 663 419 515 951 658 326 795 162 726 995 718
036 683 916 808 897 228 862 816 097 651 890 634 473
447 230 125 646 343 018 165 041 949 562 889 516 312
628 858 828 794 754 077 339 514 161 333 120 422 196
381 290 698 294 111 796 885 471 968 093 243 031 989
625 871 284 871 508 154 810 239 086 222 666 436 262
893 251 103 541 793 189 622 659 158 297 561 789 893
033 193 424 503 065 371 546 622 146 322 621 294 967
033 310 829 845 971 190 707 522 698 581 901 186 900
437 662 799 744 094 828 330 277 730 869 959 595 630
470 630 079 667 785 168 698 626 143 093 807 785 746
565 682 741 170 346 815 144 259
=== Sign In successfully ===
Username: admin
Password: 259
```