

NGÔN NGỮ LẬP TRÌNH JAVA

BUỔI 6 CÁC THÀNH PHẦN TIỆN ÍCH (TT)



GVGD: ThS. Lê Thanh Trọng

NỘI DUNG

- 1. Enum**
- 2. System**
- 3. Date, Time**
- 4. Regular Expression**
- 5. Lập trình Generic**

NỘI DUNG

- 1. Enum**
- 2. System**
- 3. Date, Time**
- 4. Regular Expression**
- 5. Lập trình Generic**

Giới thiệu về enum

- ❖ Có từ JDK 1.5
- ❖ Dùng định nghĩa tập hợp các hằng số
- ❖ Là một kiểu đặc biệt của lớp trong Java
- ❖ Một enum có thể chứa các trường dữ liệu, phương thức và constructor
- ❖ Có thể được định nghĩa bên trong hoặc bên ngoài một lớp

```
public class EnumTest {  
    enum Size {S, M, L, XL}  
    public static void main(String[] args) {  
        Size size = Size.XL;  
        System.out.println(size);    //XL  
    }  
}
```

```
enum Size {S, M, L, XL}  
public class EnumTest {  
    public static void main(String[] args) {  
        Size size = Size.XL;  
        System.out.println(size);    //XL  
    }  
}
```

Duyệt các phần tử trong enum

- ❖ values(): mảng chứa tất cả các giá trị của enum

```
enum Size {S, M, L, XL}
public class EnumTest {
    public static void main(String[] args) {
        for(Size size : Size.values())
            System.out.print(size + " ");
    }
}
```

Trường dữ liệu enum

```
enum Size {  
    S(1), M(2), L(3), XL(4);  
    public int value;  
    private Size(int i){  
        value = i;  
    }  
}  
  
public class EnumTest {  
    public static void main(String[] args) {  
        for(Size size : Size.values())  
            System.out.print(size.value);  
    }  
}
```

Enum trong câu lệnh switch

```
public class EnumTest {  
    public static void main(String[] args) {  
        Size size = Size.S;  
        switch(size)  
        {  
            case S:  
                System.out.println("Size S");  
                break;  
            case M:  
                System.out.println("Size S");  
                break;  
            case L:  
                System.out.println("Size S");  
                break;  
            case XL:  
                System.out.println("Size S");  
                break;  
        }  
    }  
}
```

So sánh các phần tử enum

- ❖ Sử dụng phương thức **equals()** hoặc toán tử **==** (so sánh bằng)

```
public class EnumTest {  
    public static void main(String[] args) {  
        Size size1= Size.S;  
        Size size2= Size.L;  
        Size size3= Size.S;  
        System.out.println(size1.equals(size2));    //false  
        System.out.println(size1==size3);           //true  
    }  
}
```


Chuyển enum sang chuỗi

❖ Sử dụng phương thức **toString()**

```
public class EnumTest {  
    public static void main(String[] args) {  
        String str = Size.L.toString();  
        System.out.println(str);    //"L"  
    }  
}
```

NỘI DUNG

1. Enum
2. **System**
3. Date, Time
4. Regular Expression
5. Lập trình Generic

Giới thiệu về lớp System

- ❖ java.lang.System
 - *public final class System extends Object*
- ❖ Chứa một số trường dữ liệu và phương thức hữu ích
- ❖ Không cần khởi tạo
- ❖ Cung cấp
 - standard output, standard input
 - error output streams
 - Cách thức sao chép các phần tử của mảng
 - Nạp các file dữ liệu và thư viện

Các trường dữ liệu trong System

- ❖ **static PrintStream err:** standard error output stream
- ❖ **static InputStream in:** standard input stream
- ❖ **static PrintStream out:** standard output stream

Các phương thức thông dụng trong System

- ❖ *static void **arraycopy**(Object src, int srcPos, Object dest, int destPos, int length):* sao chép **length** phần tử từ mảng **src**, bắt đầu từ vị trí **srcPos**, đến vị trí **destPos** của mảng **dest**

```
public static void main(String[] args) {  
    int arr1[] = {0, 1, 2, 3};  
    int arr2[] = {4, 5, 6};  
    System.arraycopy(arr1, 1, arr2, 1, 2); //arr2 = {4, 1, 2}  
}
```

- ❖ *public static long **currentTimeMillis**():* trả về thời gian hiện tại tính bằng mili giây

```
System.out.println(System.currentTimeMillis());
```

Các phương thức thông dụng trong System

❖ *static void* **exit**(int status): Dừng máy ảo Java đang chạy

```
public static void main(String[] args) {  
    for(int i =0; i< 100; i++)  
    {  
        System.out.println(i);  
        if(i>50)  
        {  
            System.out.println("exit...");  
            System.exit(0);  
        }  
    }  
}
```

Các phương thức thông dụng trong System

- ❖ *public static void **load**(String filename):* nạp các file code với tên gọi và đường dẫn (dynamic library)

```
public static void main(String[] args) {  
    System.load("E:\\SystemLoadTest.ext");  
}
```

NỘI DUNG

1. Enum
2. System
- 3. Date, Time**
4. Regular Expression
5. Lập trình Generic

Các lớp liên quan (JDK 8)

- ❖ **LocalDate:** Biểu diễn date (ngày/tháng/năm) theo tiêu chuẩn ISO 8601, không Time, không Time zone
- ❖ **LocalTime:** Biểu diễn time (giờ, phút, giây) theo tiêu chuẩn ISO 8601, không Date, không Time zone
- ❖ **LocalDateTime:** bao gồm cả 2 API trên, tạo ra instance chứa cả Date, Time và không có Time zone
- ❖ **ZonedDateTime:** bao gồm API LocalDateTime có Time zone

LocalDate

❖ java.time.LocalDate;

```
//Lấy ngày hiện tại
LocalDate currentDate = LocalDate.now();
System.out.println("Thời gian hiện tại = " + currentDate);
//Tạo đối tượng LocalDate là '1/1'2021'
LocalDate firstDay2021 = LocalDate.of(2021, Month.JANUARY, 1);
System.out.println("Ngày đầu tiên của năm 2021 = " + firstDay2021);
//Lấy thời gian hiện tại tại Alaska, Mỹ
LocalDate todayAlaska = LocalDate.now(ZoneId.of("US/Alaska"));
System.out.println("Thời gian tại Alaska (IST) = " + todayAlaska);
//Trả về ngày sau n ngày tính từ ngày đầu tiên của năm bất kỳ
LocalDate _365Day2021 = LocalDate.ofYearDay(2021, 365);
System.out.println("Ngày thứ 365 của năm 2014 = " + _365Day2021); //'31/12/2021
```

LocalTime

❖ java.time.LocalTime;

```
LocalTime CurrentTime = LocalTime.now();
System.out.println("Current Time=" + CurrentTime);
//LocalTime.of(int hour, int minute, int second, int nanoOfSecond)
LocalTime CreatedTime = LocalTime.of(12, 20, 25, 40);
System.out.println("Specific Time of Day = " + CreatedTime);
//Lấy thời gian hiện tại Hồ Chí Minh (IST)
LocalTime HCMTime = LocalTime.now(ZoneId.of("Asia/Ho_Chi_Minh"));
System.out.println("Thời gian tại HCM (IST) = " + HCMTime);
//Lấy thời gian cách n giây sau ngày '01/01/1970'
LocalTime NextTime = LocalTime.ofSecondOfDay(10000);
System.out.println("Thời gian sau 10000 giây = " + NextTime);
```

LocalDateTime

❖ java.time.LocalDateTime;

```
//Lấy thời gian hiện tại
LocalDateTime Today = LocalDateTime.now();
System.out.println("Thời gian hiện tại = " + Today);

//LocalDateTime.of(int year, Month month, int dayOfMonth, int hour, int minute, int second)
LocalDateTime CreatedDate = LocalDateTime.of(2021, Month.JANUARY, 1, 15, 10, 30);
System.out.println("Specific Date = " + CreatedDate);

//Thời gian hiện tại của TP. Hồ Chí Minh
LocalDateTime HCMDateTime = LocalDateTime.now(ZoneId.of("Asia/Ho_Chi_Minh"));
System.out.println("Current Date in IST = " + HCMDateTime);
```

ZonedDateTime

❖ java.time.ZonedDateTime

```
//Lấy danh sách gồm các zone
Set<String> AllZoneIds = ZoneId.getAvailableZoneIds();
for(String s:AllZoneIds)
    System.out.println(s);

// Creating LocalDateTime by providing input arguments
LocalDateTime Today = LocalDateTime.now();
System.out.println("LocalDateTime = " + Today);

//Tạo một zone tại TP.Hồ Chí Minh
ZoneId HCMZone = ZoneId.of("Asia/Ho_Chi_Minh");

//Creating ZonedDateTime by providing input arguments
ZonedDateTime HCMDateTime = ZonedDateTime.of(Today, HCMZone);
System.out.println("ZonedDateTime = " + HCMDateTime);

//Tạo offsets
ZoneOffset Offset = ZoneOffset.of("+05:00");
System.out.println("Offset = " + Offset);

OffsetDateTime TodayPlusFive = OffsetDateTime.of(Today, Offset);
System.out.println("Hiện tại +5 = " + TodayPlusFive);
```

NỘI DUNG

1. Enum
2. System
3. Date, Time
- 4. Regular Expression**
5. Lập trình Generic

Giới thiệu Regular Expression

- ❖ Là biểu thức chính quy được dùng để xử lý chuỗi nâng cao thông qua biểu thức
- ❖ Còn gọi là Regex
- ❖ Nguyên tắc hoạt động của biểu thức Regex là so khớp dựa vào khuôn mẫu (pattern)
- ❖ Regex thường được sử dụng để xử lý chuỗi, xử lý văn bản như: tìm và thay thế chuỗi, kiểm tra tính hợp lệ của dữ liệu, trích xuất chuỗi con từ một chuỗi,...

Các ký tự thường dùng trong Regex

BT chính quy	Mô tả
.	Bất kỳ ký tự nào
^regex	Khớp tại điểm bắt đầu
regex\$	Khớp tại điểm kết thúc
[abc]	Khớp với bất kỳ ký tự nào trong cặp dấu [], có thể khớp với a hoặc b hoặc c
[abc][xy]	Khớp với a hoặc b hoặc c và theo sau là x hay y
[^abc]	Khớp với bất kỳ ký tự nào ngoại trừ a hoặc b hoặc c.
[a-d1-7]	Chuỗi giữa a – d và các số từ 1 đến 7
a b	Tìm a hoặc b
\d	Số bất kỳ, viết ngắn gọn cho [0-9]
\D	Ký tự không phải là số, viết ngắn gọn cho [^0-9]
\w	Ký tự chữ, viết ngắn gọn cho [a-z, A-Z, 0-9]
\W	Ký tự không phải chữ, viết ngắn gọn cho [^\w]
\b	Ký tự thuộc a-z hoặc A-Z hoặc 0-9 hoặc _, viết ngắn gọn cho [a-zA-Z0-9_]

Các ký tự thường dùng trong Regex

BT chính quy	Mô tả
\s	Ký tự khoảng trắng, viết ngắn gọn cho [\t\n\r\f]
\S	Ký tự không phải khoảng trắng, viết ngắn gọn cho [^\s]
{X}	Xuất hiện X lần
{X, }	Xuất hiện X lần trở lên
{X,Y}	Xuất hiện trong khoảng X tới Y lần
*	Xuất hiện 0 hoặc nhiều lần, A*B khớp với B, AB, AAB
+	Xuất hiện 1 hoặc nhiều lần, A+B khớp với AB, AAB
?	Xuất hiện 0 hoặc 1 lần, A?B sẽ khớp với B hay AB
X Z	Tìm X hoặc Z
XZ	Tìm X và theo sau là Z

Sử dụng Regex

❖ *public boolean **matches**(String regex)*

```
String str1 = "Toi la Java";  
//bắt đầu là 'T', theo sau là bất kì ký tự nào (true)  
System.out.println(str1.matches("^T.+"));  
//Kết thúc là 'a'  
System.out.println(str1.matches("."+a$"));  
//Kiểm tra phải là số phone hay không  
//Số phone: 10 số, bắt đầu là 0  
String phoneNumber = "0912345678";  
System.out.println(phoneNumber.matches("^0+[1-9]{1}+[0-9]{8}"));  
//Kiểm tra thời gian dd/mm/yyyy  
String myDate = "31-12-2021";  
System.out.println(myDate.matches("([0-2][1-9]|3[01])-(0[1-9]|1[0-2])-([12][0-9]{3})"));
```

Sử dụng Regex

❖ *public String[] **split**(String regex)*

```
String str = "Chao    mung  ban      den voi \t Java";  
String[] arr = str.split("\\s+");  
System.out.println(Arrays.toString(arr)); // [Chao, mung, ban, den, voi, Java]  
  
//Thay thế tất cả các khoảng trắng với ký tự '_'.  
String newString = str.replaceAll("\\s+", "_");  
System.out.println(newString); //Chao_mung_ban_den_voi_Java
```

NỘI DUNG

1. Enum
2. System
3. Date, Time
4. Regular Expression
5. Lập trình Generic

Giới thiệu Generic

- ❖ Có từ JDK 5
- ❖ Cho phép chỉ định kiểu dữ liệu làm việc với một class, một interface hay một phương thức tại thời điểm biên dịch
- ❖ Ưu điểm
 - Kiểm tra dữ liệu chặt chẽ ở Compile-time, giúp an toàn kiểu dữ liệu (không cho phép lưu trữ các loại đối tượng khác kiểu được chỉ định)
 - Các thuật toán được sử dụng nhiều (reusable), dễ dàng thay đổi, an toàn dữ liệu và dễ đọc, sử dụng ở nhiều tình huống khác nhau
- ❖ Ví dụ:
 - //Chỉ được thêm vào các phần tử Integer
 - *List<Integer> list = new ArrayList<Integer>();* //Trước Java 7
 - *List<Integer> list = new ArrayList<>();* //Từ Java 7

Đặt tên với Generic

- ❖ Đặt theo quy ước chung để dễ đọc, dễ bảo trì
- ❖ Gợi ý đặt tên
 - **E**: Element (phần tử – trong Collection Framework)
 - **K**: Key (khóa)
 - **V**: Value (giá trị)
 - **N**: Number (kiểu số: Integer, Double, Float, ...)
 - **T**: Type (Kiểu Wrapper class: String, Integer, Long, Float, ...)
 - **S, U, V, ...**: được sử dụng để đại diện cho các kiểu dữ liệu (Type) thứ 2, 3, 4, ...

Ví dụ

```
class KeyAndValue<K, V> {  
    private K key;  
    private V value;  
  
    public KeyAndValue(K key, V value) {  
        this.key = key;  
        this.value = value;  
    }  
    public K getKey() {  
        return key;  
    }  
    public void setKey(K key) {  
        this.key = key;  
    }  
    public V getValue() {  
        return value;  
    }  
    public void setValue(V value) {  
        this.value = value;  
    }  
}
```

```
public static void main(String[] args) {  
    KeyAndValue<String, Integer> entry =  
        new KeyAndValue<>("Anna", 9123);  
    String name = entry.getKey();  
    Integer id = entry.getValue();  
    System.out.println("Name = " + name + ", ID = " + id);  
}
```

Kế thừa với Generic

```
class MoreWithKeyAndValue<K,V,M> extends KeyAndValue<K, V> {  
    M more;  
    public MoreWithKeyAndValue(K key, V value, M more) {  
        super(key, value);  
        this.more = more;  
    }  
    public M getMore() {  
        return more;  
    }  
    public void SetMore(M more) {  
        this.more = more;  
    }  
}
```

```
MoreWithKeyAndValue<String, Integer, Long> entry =  
    new MoreWithKeyAndValue<>("Anna", 9123, 180000000L);  
String name = entry.getKey();  
Integer id = entry.getValue();  
Long salary = entry.getMore();  
System.out.println("Name = " + name + ", ID = " + id + ", Salary = " + salary);
```


Generic với Interface

```
interface GenericInterface<T> {  
    void Add(T obj);  
    void Delete(T obj);  
}  
  
class ImplClass<T> implements GenericInterface<T> {  
    @Override  
    public void Add(T obj) {  
        //code hiện thực  
    }  
    @Override  
    public void Delete(T obj) {  
        //code hiện thực  
    }  
}
```

Generic với Interface

```
class Employee
{
    String name;
    Long salary;
    Employee(String name, Long salary)
    {
        this.name = name;
        this.salary = salary;
    }
}
```

```
ImplClass<Employee> st = new ImplClass<>();
Employee employee = new Employee("Jack", 90000000L);
st.Add(employee);
```

Generic với phương thức

```
class Employee extends Object
{
    String name;
    Long salary;
    Employee(String name, Long salary)
    {
        this.name = name;
        this.salary = salary;
    }
    @Override
    public boolean equals(Object obj)
    {
        Employee temp = (Employee)obj;
        return (salary.equals(temp.salary));
    }
}
```

```
class Utility
{
    public static <T> boolean compare(T a, T b)
    {
        return a.equals(b);
    }
}
```

```
Employee employee1 = new Employee("Jack", 90000000L);
Employee employee2 = new Employee("Nam", 90000000L);
System.out.println(Utility.compare(employee1, employee2)); //true
```

Tóm tắt bài học

- ❖ Enum dùng định nghĩa tập hợp các hằng số, là một kiểu đặc biệt của lớp trong Java, có thể chứa các trường dữ liệu, phương thức và constructor, có thể được định nghĩa bên trong hoặc bên ngoài một lớp
- ❖ Lớp System chứa một số trường dữ liệu và phương thức hữu ích, không cần khởi tạo khi sử dụng, cung cấp các thao tác trên luồng nhập, xuất, sao chép các phần tử của mảng, nạp các file dữ liệu và thư viện,...
- ❖ Từ JDK 8, các lớp tiện ích liên quan đến quản lý ngày, thời gian và zone gồm có LocalDate, LocalTime, LocalDateTime, ZonedDateTime thuộc package **java.time.***

Tóm tắt bài học

- ❖ Regular Expression là biểu thức chính quy được sử dụng để xử lý chuỗi, xử lý văn bản như: tìm và thay thế chuỗi, kiểm tra tính hợp lệ của dữ liệu, trích xuất chuỗi con từ một chuỗi,...
- ❖ Từ JDK 5, Generic cho phép chỉ định kiểu dữ liệu làm việc với một class, một interface hay một phương thức tại thời điểm biên dịch giúp kiểm tra dữ liệu chặt chẽ ở compile-time, giúp an toàn kiểu dữ liệu và tái sử dụng.

Bài tập

1. Cho enum Qua{ NHO(10), TÁO(24), XOÀI(11), CHANH(99)}

Xây dựng lớp chứa ID (int) nhằm quản lý một quả và override phương thức toString() trả về tên quả.

Viết chương trình cho phép người dùng nhập vào N quả (N do người dùng nhập), và tính số tiền phải mua biết:

- Nho: 1000đ/quả.
- Táo: 10000đ/quả.
- Xoài: 5000đ/quả.
- Chanh: 500đ/quả.

2. Viết một chương trình có enum là 12 tháng trong năm. Khởi tạo giá trị của các phần tử trong enum như: JANUARY(1), FEBRUARY(2), ... , DECEMBER(12). Tạo một số nguyên ngẫu nhiên từ 1 - 12 (random), xuất số ngày trong tháng ứng với phần tử trong enum có giá trị bằng giá trị số nguyên trên (tháng 2 có 28 ngày).

Bài tập

3. Viết phần mềm quản lý thông tin đăng nhập của người dùng. Người dùng sử dụng tài khoản bao gồm tên đăng nhập và mật khẩu, tên đăng nhập không bắt đầu bằng số, không bao gồm kí tự đặc biệt; mật khẩu gồm ít nhất 3 kí tự.

Ví dụ:

- ❖ Tên đăng nhập: "java123", mật khẩu: "oracle", là hợp lệ
- ❖ Tên đăng nhập: "123java", mật khẩu: "oracle", là không hợp lệ
- ❖ Tên đăng nhập: "java#\$", mật khẩu: "oracle", là không hợp lệ
- ❖ Tên đăng nhập: "java123", mật khẩu: "or", là không hợp lệ

Viết chương trình có các tính năng sau:

- a. Nhập vào danh sách các Tài khoản, các Tài khoản phải thỏa mã yêu cầu tên đăng nhập và mật khẩu, tên đăng nhập không được trùng lặp; nếu không thì nhập lại. In ra thông tin các tài khoản đó.
- b. Để người dùng dễ đặt tên đăng nhập, trong thao tác nhập tên đăng nhập, nếu tên đăng nhập không hợp lệ do có kí tự đặc biệt thì in ra gợi ý cho người dùng tên đã bỏ các kí tự đó. (Lưu ý sử dụng StringTokenizer) ví dụ: Java#Oracle không hợp lệ, in ra gợi ý đặt tên là JavaOracle.

Bài tập

4. Viết chương trình thực thi các yêu cầu sau:
 - a. Kiểm tra năm nay có phải năm nhuận hay không, ngày đầu tiên của năm vào thứ mấy trong tuần?
 - b. In ngày hiện tại theo các định dạng khác nhau.
 - c. Nhập vào 1 ngày bất kỳ kiểm tra xem ngày đó là thứ mấy trong tuần.
 - d. Lấy ngày đầu tiên và ngày cuối cùng của một tháng từ 1 ngày người dùng nhập vào.
 5. Viết chương trình cho phép nhập vào các trường username, email, password với các yêu cầu
 - 1/ không cho phép 1 trong 3 trường trên trống
 - 2/ kiểm tra username số ký tự $15 \leq \text{username} \leq 30$ (chỉ gồm chữ cái, khoảng trắng, và ký tự đầu mỗi từ phải là chữ hoa)
 - 3/ kiểm tra email theo định dạng `aaa@bbb.ccc` với:
 - email không được chứa chữ hoa, ký tự đặc biệt.
 - aaa: có thể là chữ thường hoặc số (trừ ký tự đầu là chữ (a->z)), bbb và ccc là chữ thường.
 - 4/ kiểm tra password ít nhất 8 ký tự, ký tự đầu là chữ hoa (A->Z), password chỉ gồm chữ hoa, chữ thường và số (không được chứa ký tự đặc biệt)
- ❖ Chương trình xuất ra 3 trường trên nếu đúng tất cả các định dạng, ngược lại thông báo lỗi kèm theo chi tiết lỗi định dạng mắc phải (dạng 1, 2, 3) nếu sai định dạng.