UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

NGÔN NGỮ LẬP TRÌNH JAVA

Chương 3
CÁC LỚP TIỆN ÍCH TRONG JAVA (P2)

GVGD: ThS. Lê Thanh Trọng

NỘI DUNG

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

NỘI DUNG

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

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

Giới thiệu về enum

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

- 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

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

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

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

- 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

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

- 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

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

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

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

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 gon 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

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

BT chính quy	Mô tả
\s	Ký tự khoảng trắng, viết ngắn gọn cho [\t\n\x0b\r\f]
\S	Ký tự không phải khoản 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

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

- *public boolean matches(String regex)*

```
String str1 = "Toi la Java";
//bắt đầu là 'T', theo sao 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

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

- *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

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

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

Giới thiệu Generic

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

- 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 Interger
  - *List<Integer> list = new ArrayList<Integer>();* //Trước Java 7
  - *List<Integer> list = new ArrayList<>();* //Từ Java 7

Đặt tên với Generic

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

- 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

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

```
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, 18000000L);
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

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

```
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", 9000000L);
st.Add(employee);
```

Generic với phương thức

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

```
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", 9000000L);
Employee employee2 = new Employee("Nam", 9000000L);
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.