

Bài 5

Câu lệnh điều kiện - 2

Môn học: PF-JAVA

Mục tiêu

- Trình bày cú pháp câu lệnh switch-case
- So sánh giữa if bậc thang và switch-case
- Giải thích cách sử dụng từ khoá break, default
- Sử dụng được câu lệnh điều kiện switch-case
- Sử dụng từ khoá break, default
- Trình bày được biểu thức điều kiện
- Sử dụng được biểu thức điều kiện
- Giải thích các bước để debug một ứng dụng Java
- Debug được ứng dụng Java

switch-case

switch-case

switch-case với kiểu dữ liệu String

switch-case với kiểu enumeration

Từ khoá break và default

Câu lệnh switch-case

- switch-case là một cấu trúc điều kiện cho phép lựa chọn thực thi các khối lệnh khác nhau dựa trên kết quả của việc so sánh
- switch-case so sánh giá trị của một biến với lần lượt từng giá trị một, nếu có giá trị phù hợp với biến thì khối lệnh tương ứng sẽ được thực thi
- switch-case không thể thay thế if-else trong tất cả các trường hợp

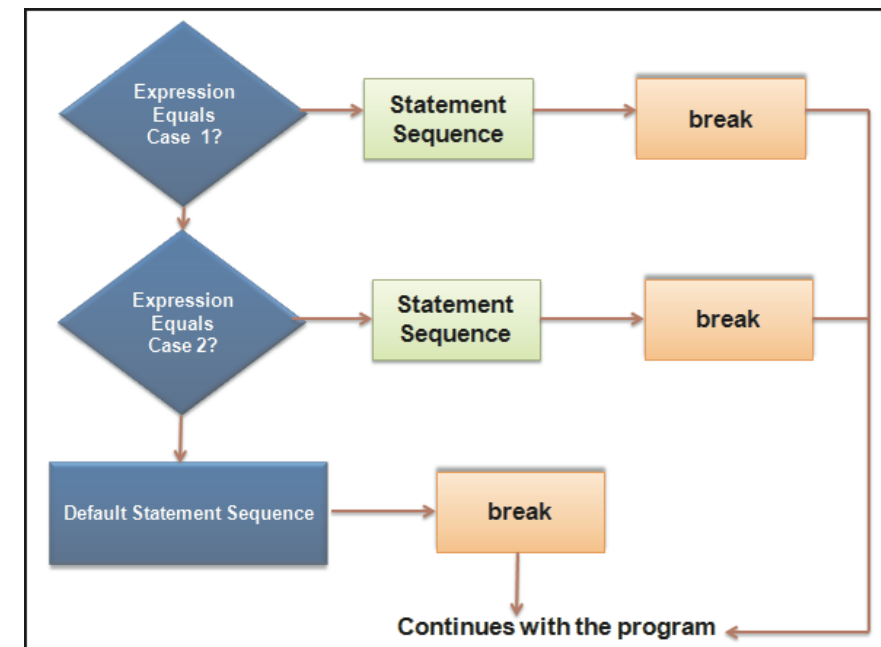
switch-case: Cú pháp

```
switch (switch-expression) {  
    case value1: statement(s)1;  
                break;  
    case value2: statement(s)2;  
                break;  
    ...  
    case valueN: statement(s)N;  
                break;  
    default:    statement(s)-for-default;  
}
```

- Trong đó:
 - *switch-expression*: là biểu thức trả về giá trị thuộc một trong các kiểu: char, byte, short, int hoặc String
 - *value1, ..., valueN* có cùng kiểu dữ liệu so với switch-expression
 - *break* là từ khoá để dừng thực thi các câu lệnh ở phía sau. *break* là không bắt buộc.
 - *default* là từ khoá để quy định khối lệnh sẽ được thực thi nếu không có trường hợp nào ở các *case* là đúng. *default* là không bắt buộc.

switch-case: Mô tả

- Giá trị của biểu thức sẽ được so sánh với từng trường hợp (case)
- Nếu có trường hợp bằng nhau thì khối lệnh tương ứng sẽ được thực thi
- Nếu gặp câu lệnh break thì sẽ kết thúc thực thi khối switch-case
- Nếu gặp trường hợp bằng nhau, nhưng sau đó không có câu lệnh break thì tất cả những khối lệnh phía sau cũng được thực thi
- Nếu không có trường hợp nào bằng nhau thì khối lệnh trong default (nếu có) sẽ được thực thi



switch-case: Ví dụ tính thuế thu nhập cá nhân

```
switch (level) {  
    case 0:  
        //Tính thuế cho thu nhập đến 5tr  
        break;  
    case 1:  
        //Tính thuế cho thu nhập từ 5tr đến 10tr  
        break;  
    case 2:  
        //Tính thuế cho thu nhập từ 10tr đến 18tr  
        break;  
    .....  
        //Tính thuế cho thu nhập từ 18tr đến 32tr  
        break;  
    default:  
        //Không hợp lệ  
        break;  
}
```

Gộp nhiều case

- Có thể lựa chọn thực thi một khối lệnh nếu giá trị bằng với một trong số nhiều case được gộp lại với nhau
- Ví dụ, phân loại ngày trong tuần với ngày cuối tuần:

```
switch (day) {  
    case 1:  
    case 2:  
    case 3:  
    case 4:  
    case 5:  
        System.out.println("Weekday");  
        break;  
    case 0:  
    case 6:  
        System.out.println("Weekend");  
}
```


switch-case với kiểu dữ liệu String

- Từ Java SE 7 thì switch-case hỗ trợ thêm kiểu dữ liệu String

- Ví dụ:

```
switch (greeting) {  
    case "hello":  
        System.out.println("You are speaking English"); break;  
    case "hola":  
        System.out.println("Usted está hablando español"); break;  
    case "bonjour":  
        System.out.println("Vous parlez français"); break;  
    case "hallo":  
        System.out.println("Du sprichst Deutsch"); break;  
}
```

Lưu ý: Nếu greeting có giá trị null thì sẽ phát sinh lỗi khi thực thi

Kiểu Enum

- Enum là kiểu dữ liệu đặc biệt, cho phép định nghĩa một danh sách các hằng cho trước
- Khi biến được khai báo thuộc một kiểu enum thì giá trị của nó phải là một trong các hằng của enum
- Ví dụ, định nghĩa các hằng để chỉ hướng đi (Đông, Tây, Nam, Bắc)

```
enum DIRECTION{  
    NORTH, SOUTH, EAST, WEST  
}
```

```
DIRECTION direction = DIRECTION.NORTH;
```

switch-case với kiểu enumeration

- Ví dụ, khai báo enum:

```
enum Cards {  
    Spade, Heart, Diamond, Club  
}
```

- Sử dụng trong switch-case:

```
switch (card) {  
    case Spade:  
        System.out.println("SPADE"); break;  
    case Heart:  
        System.out.println("HEART"); break;  
    case Diamond:  
        System.out.println("DIAMOND"); break;  
    case Club:  
        System.out.println("CLUB"); break;  
}
```

So sánh if và switch-case

if	switch-case
Có thể sử dụng để so sánh lớn hơn, nhỏ hơn...	Chỉ có thể sử dụng để so sánh bằng hoặc khác nhau
Mỗi câu lệnh if có một biểu thức điều kiện, với giá trị trả về là true hoặc false	Tất cả các trường hợp (case) đều so sánh với giá trị của biểu thức điều kiện duy nhất
Biểu thức điều kiện cần trả về giá trị kiểu boolean	Biểu thức điều kiện cần trả về giá trị là kiểu byte, short, char, int, hoặc String
Chỉ có một khối lệnh được thực thi nếu điều kiện đúng	Nếu điều kiện đúng mà không có câu lệnh break thì tất cả các khối lệnh ở phía sau cũng được thực thi

Lưu ý: Viết mã sạch (1)

- Trong ví dụ về phân loại ngày trong tuần, việc để các case là 0, 1, 2 ... gây khó hiểu cho người đọc mã nguồn
- Có thể sử dụng phương pháp “tách hằng” để giúp cho mã nguồn dễ hiểu hơn
- Ví dụ:

```
final int MONDAY = 1;
```

```
switch (day) {  
    case 1:  
    case 2:  
    case 3:  
    case 4:  
    case 5:  
        System.out.println("Weekday");  
        break;  
    case 0:  
    case 6:  
        System.out.println("Weekend");  
}
```

Lưu ý: Viết mã sạch (2)

- Định nghĩa các hằng số:

```
final int MONDAY = 1, TUESDAY = 2, WEDNESDAY = 3, THURSDAY = 4, FRIDAY = 5, SAT = 7, SUNDAY = 0;
```

- Viết lại switch-case:

```
switch (day) {  
    case MONDAY:  
    case TUESDAY:  
    case WEDNESDAY:  
    case THURSDAY:  
    case FRIDAY:  
        System.out.println("Weekday"); break;  
    case SAT:  
    case SUNDAY:  
        System.out.println("Weekend");  
}
```

Biểu thức điều kiện

Biểu thức điều kiện

- Biểu thức điều kiện đánh giá một biểu thức dựa vào một điều kiện cho trước
- Biểu thức điều kiện là một toán tử 3 ngôi
- Cú pháp:

condition ? expression_true_case : expression_false_case

Trong đó:

- *condition*: biểu thức điều kiện dùng để đánh giá
- *expression_true_case*: biểu thức sẽ được sử dụng trong trường hợp **true**
- *expression_false_case*: biểu thức sẽ được sử dụng trong trường hợp **false**

Biểu thức điều kiện: Ví dụ

- Tìm giá trị lớn nhất trong 2 số:

```
int maxValue = number1 > number2 ? number1 : number2;
```

- Tính số lượng dựa trên điều kiện khoảng cách:

```
int count = isHere ? getHereCount(index) : getAwayCount(index);
```

Debug ứng dụng Java

Debug

- Debug (dò lỗi) là quá trình tìm kiếm và sửa chữa các lỗi trong một chương trình
- Các loại lỗi thường gặp:
 - Lỗi cú pháp (syntax error) dễ phát hiện bởi vì compiler đưa ra thông báo
 - Lỗi thực thi (runtime error) cũng dễ phát hiện bởi vì interpreter cũng hiển thị lỗi trên màn hình
 - Lỗi logic thường khó phát hiện hơn
- Các lỗi logic còn được gọi là bug
- Thông thường, cần kết hợp nhiều cách để tìm được bug

Một số phương pháp debug

- Đọc mã nguồn (hand-trace)
- Chèn các lệnh in ra các giá trị trong từng đoạn của chương trình để kiểm tra các giá trị và việc thực thi các câu lệnh
- Sử dụng debugger: một chương trình cho phép quan sát quá trình thực thi của một ứng dụng
- Java cung cấp sẵn một debugger là **jdb** (cũng là một ứng dụng Java)
- Các IDE thông thường cũng tích hợp sẵn debugger

Các tính năng của debugger

- Thực thi riêng lẻ từng câu lệnh tại một thời điểm
- Theo dõi hoặc bỏ qua luồng thực thi vào sâu bên trong một phương thức
- Đặt các breakpoint: các điểm mà chương trình sẽ dừng lại để lập trình viên quan sát trạng thái hiện tại của chương trình
- Hiển thị giá trị của các biến tại từng thời điểm
- Hiển thị ngăn xếp các phương thức được gọi
- Thay đổi giá trị của biến tại thời điểm thực thi (chỉ một số debugger hỗ trợ tính năng này)

Stack trace

- Stack trace thể hiện lần lượt các phương thức được gọi
- Sử dụng stack trace hỗ trợ việc tìm ra loại lỗi và vị trí lỗi
- Ví dụ:
 - Lỗi chia cho 0
 - Vị trí lỗi: dòng số 7, bên trong phương thức *divide()*


```
public static int divide(int a, int b){  
    return 1 / (a - b);  
}
```

```
public static void main(String[] args) {  
    int a = 1;  
    int b = 1;  
    int c = divide(a, b);  
}
```

```
Exception in thread "main" java.lang.ArithmeticException: / by zero  
    at SystemTime.divide(SystemTime.java:7)  
    at SystemTime.main(SystemTime.java:14)
```

```
Process finished with exit code 1
```

Debug với IntelliJ IDEA

- Khởi chạy chế độ debug bằng phím (^D) hoặc nút 
- Đặt breakpoint
- Xem thông tin chi tiết tại điểm breakpoint
- Step into (quan sát sâu vào bên trong một phương thức được gọi)
- Step over (không quan sát bên trong một phương thức được gọi)

[Thực hành] Giải phương trình bậc nhất

[Thực hành] Tính số ngày của tháng

[Bài tập] Ứng dụng xử số

[Bài tập] Giải phương trình bậc hai

[Bài tập] Đọc số thành chữ

[Bài tập] Debug ứng dụng

Tổng kết

- switch-case là một cấu trúc điều kiện cho phép lựa chọn thực thi các khối lệnh khác nhau dựa trên kết quả của việc so sánh
- Cần lựa chọn sử dụng switch-case và if phù hợp với từng tình huống
- switch-case có thể sử dụng với kiểu dữ liệu String
- switch-case có thể sử dụng với kiểu dữ liệu enum
- Debug là quá trình tìm kiếm và sửa lỗi
- Java cung cấp sẵn debugger
- IntelliJ IDEA và các IDE khác thường tích hợp sẵn debugger