1. Biến và hằng số trong Java

Biến trong Java

Mỗi biến trong Java bao gồm 3 phần sau: tên biến, kiểu dữ liệu và giá trị của biến đó.

- Tên biến là sự biểu diễn tượng trưng của vùng nhớ trong đó thông tin được lưu trữ.
- Kiểu dữ liệu dùng để xác định kích thước và loại giá trị có thể được lưu trữ.
- Giá trị là dữ liệu thực tế được lưu trữ trên biến và có thể thay đổi được.

Trong Java có 3 loại biến thường gặp đó là: Local variable (biến cục bộ), Instance variable (thuộc tính) và Static variable (biến tĩnh).

Biến hằng trong Java

Để khai báo hằng số ta sử dụng từ khóa static final đặt trước tên hằng số: [Phạm vi truy cập] static final [kiểu dữ liệu] [tên hằng số] = [giá trị];

2. Cấu trúc điều khiển

Cấu trúc điều khiển if - else trong Java

Cấu trúc điều khiển if - else đầy đủ

```
if (điều kiện) {
     hành động 1;
} else {
     hành động 2;
}
```

Cấu trúc điều khiển if - else khuyết else

```
if (điều kiện) {
     hành động;
}
```

Cấu trúc điều khiển if - else if - else

```
if (điều kiện 1) {
     hành động 1;
} else if (điều kiện 2) {
     hành động 2;
```

```
} …. else {
      hành động n;
}
```

Cấu trúc điều khiển if - else lồng nhau

```
if (điều kiện 1) {
      hành động 1
} else if (điều kiện 2) {
      hành động 2
} else if (điều kiện 3) {
      hành động 3
} ... else {
      hành động n
}
```

Cấu trúc toán tử điều kiện 3 ngôi

Toán tử điều kiện 3 ngôi là cấu trúc thay thế của biểu thức điều kiện if - else trong Java. [Biểu thức điều kiện] ? [Giá trị 1] : [Giá trị 2];

Cấu trúc rẽ nhánh switch - case trong Java

Cấu trúc rẽ nhánh switch - case cho phép lựa chọn một trong nhiều phương án có khả năng xảy ra, nó có thể dùng để thay thế cho cấu trúc điều khiển if - else if - else

```
switch (biểu thức) {
    case giá_trị 1:
        Lệnh 1;
        break;
    case giá_trị 2:
        Lệnh 2;
        break;
    ...
    case giá trị n:
        Lệnh n;
        break;
    [default: Lệnh 0;]
}
```

3. Vòng lặp

Vòng lặp while trong Java

Vòng lặp while là cấu trúc điều khiển lặp được dùng để thực hiện một lệnh hay một khối lệnh với số lần lặp chưa xác định trước. Với cấu trúc này điều kiện lặp được kiểm tra trước khi thực hiện thân của vòng lặp.

```
while (điều_kiện lặp) {
    // Các lệnh
}
```

Vòng lặp do - while trong Java

Vòng lặp do - while là cấu trúc điều khiển lặp được dùng để thực hiện một lệnh hay một khối lệnh với số lần lặp chưa xác định trước nhưng khác với while, cấu trúc do - while chỉ kiểm tra điều kiện lặp sau khi thân vòng lặp đã được thực hiện một lần.

```
do {
      // Các lệnh
} while (điều_kiện lặp);
```

Vòng lặp for trong Java

Vòng lặp for là cấu trúc điều khiển lặp được dùng để thực hiện một lệnh hay một khối lệnh với số lần lặp được xác định trước.

```
for ([biểu thức _1]; [biểu thức 2]; [biểu thức 3]) {
    // Các lệnh
}
```

Từ khóa break và continue trong Java

Break

Ngoài việc dùng trong cấu trúc switch, từ khóa break được dùng để thoát ra khỏi vòng lặp chứa nó ngay lập tức và chuyển sang câu lệnh tiếp theo bên ngoài vòng lặp vừa kết thúc (tức là chương trình sẽ dừng ngay mọi vòng lặp nếu bên trong vòng lặp đó có chứa từ khóa break). Thông thường, từ khóa break thường được dùng với một lệnh if bên trong vòng lặp để kiểm tra điều kiện dừng của vòng lặp.

Continue

Khi gặp từ khóa continue thì lần lặp kế tiếp sẽ được thực hiện (tức là bỏ qua không thực hiện các lệnh phía bên dưới từ khóa continue của vòng lặp và quay lên kiểm tra trở lại biểu thức điều kiện lặp). Tương tự như break, từ khóa continue cũng thường được dùng với một

lệnh if bên trong vòng lặp để kiểm tra khi nào thì cần bỏ qua những lệnh sau nó để tiếp tục thực hiện vòng lặp mới.

4. Chuỗi và Mảng

Chuỗi (String) trong Java

Khai báo chuỗi

```
String tenChuoi = "giá_trị_khởi_tạo";
```

hoặc

```
String tenChuoi = new String("giá_trị");
```

Các hàm xử lý chuỗi

Hàm xác định độ dài chuỗi

```
int length = tên_chuỗi.length();
```

Hàm nối 2 chuỗi

```
String string3 = string1.concat(String string2);
```

Hàm trả về 1 ký tự trong chuỗi

```
char character = chuoi.charAt(int index);
```

Hàm so sánh 2 chuỗi

```
int result = string1.compareTo(String string2);
```

Hàm trả về vị trí xuất hiện đầu tiên của một chuỗi trong chuỗi khác

```
int result = string1.indexOf(String string2);
```

Hàm trả về vị trí xuất hiện cuối cùng của một chuỗi trong chuỗi khác.

```
int result = string1.lastIndexOf(String string2);
```

Hàm thay thế một chuỗi con trong chuỗi ký tự bằng chuỗi khác

```
string1.replace(char oldChar, char newChar);
```

Hàm loại bỏ các khoảng trắng thừa ở đầu và cuối chuỗi

```
String string1 = string1.trim();
```

Hàm tạo một chuỗi con từ vị trí index trong chuỗi cha

```
String chuoiCon = chuoiCha.substring(int beginIndex);
String chuoiCon = chuoiCha.substring(int beginIndex, int
endIndex);
```

Mảng một chiều trong Java

Khai báo mảng

```
[Kiểu_dữ_liệu] tên_mảng[];
```

hoặc

```
[Kiểu_dữ_liệu][] tên_mảng;
```

Cấp phát bộ nhớ cho mảng

```
 [Kiểu_dữ_liệu] tên_mảng[] = new [Kiểu_dữ_liệu]
[Số_phần_tử_của_mảng];
```

hoặc

```
[Kiểu_dữ_liệu][] tên_mảng = new [Kiểu_dữ_liệu]
[Số_phần_tử_của_mảng];
```

Truy xuất các phần tử của mảng

```
Tên_mảng[Chỉ_số_phần_tử];
```

Mảng hai chiều trong Java

Khai báo mảng

```
[Kiểu_dữ_liệu] Tên_mảng[][];
```

hoặc

```
[Kiểu_dữ_liệu][][] Tên_mảng;
```

Cấp phát bộ nhớ cho mảng

```
[Kiểu_dữ_liệu] Tên_mảng[][] = new [Kiểu_dữ_liệu]
[Số_dòng][Số_cột];
```

hoặc

```
[Kiểu_dữ_liệu][][] Tên_mảng = new [Kiểu_dữ_liệu]
[Số_dòng][Số_cột];
```

5. Collections

Bao gồm interface (Set, List, Queue, Deque vv) và các lớp (ArrayList, Vector, LinkedList, PriorityQueue, HashSet, LinkedHashSet, TreeSet vv)

6. Exception

Mô hình sơ đồ phân cấp của Exception trong java:

- Class ở mức cao nhất là Throwable
- Hai class con trực tiếp là Error và Exception.

Error

Khi liên kết động thất bại, hoặc trong máy ảo xảy ra một vấn đề nghiêm trọng, nó sẽ ném ra một Error. Các chương trình Java điển hình không nên bắt lỗi (Error). Ngoài ra, nó không chắc rằng các chương trình Java điển hình sẽ bao giờ ném lỗi

Exceptions

Hầu hết các chương trình ném và bắt các đối tượng là con của class Exception. Trường hợp Exception cho thấy một vấn đề xảy ra nhưng vấn đề không phải là một vấn đề mang tính hệ thống nghiêm trọng.

Runtime Exceptions

Class RuntimeException đại diện cho trường hợp ngoại lệ xảy ra trong thời gian chạy chương trình.