TẬP BÀI GIẢNG MÔN LẬP TRÌNH JAVA CƠ BẢN

BUỔI 2: CÚ PHÁP CƠ BẢN VÀ CẤU TRÚC ĐIỀU KHIỂN

I. Các kiểu dữ liệu nguyên thủy và tham chiếu

Trong Java, dữ liệu được chia thành hai loại chính: **kiểu dữ liệu nguyên thủy (primitive data types)** và **kiểu dữ liệu tham chiếu (reference data types)**. Dưới đây là chi tiết:

Trong Java, dữ liệu được chia thành hai loại chính: **kiểu dữ liệu nguyên thủy (primitive data types)** và **kiểu dữ liệu tham chiếu (reference data types)**. Dưới đây là chi tiết:

1. Kiểu dữ liệu nguyên thủy (Primitive Data Types)

Java có 8 kiểu dữ liệu nguyên thủy được định nghĩa sẵn, dùng để lưu trữ các giá trị đơn giản như số, ký tự, hoặc giá trị logic.

Kiểu dữ liệu	Kích thước (bit)	Giá trị mặc định	Phạm vi giá trị	Mô tả
`byte`	8	0	-128 đến 127	Số nguyên nhỏ gọn.
`short`	16	0	-32,768 đến 32,767	Số nguyên nhỏ.
`int`	32	0	-2^31 đến 2^31-1	Số nguyên chuẩn.
`long`	64	0L	-2^63 đến 2^63-1	Số nguyên lớn.
`float`	32	0.0f	1.4E-45 đến 3.4E+38	Số thực, dấu chấm động đơn.
`double`	64	0.0d	4.9E-324 đến 1.8E+308	Số thực, dấu chấm động kép.
`char`	16	'\u0000'	0 đến 65,535 (theo Unicode)	Ký tự Unicode.
`boolean`	1	`false`	`true` hoặc `false`	Giá trị logic.

2. Kiểu dữ liệu tham chiếu (Reference Data Types)

Kiểu tham chiếu lưu trữ địa chỉ của đối tượng (object) trong bộ nhớ. Kiểu này bao gồm:

Đối tượng (Objects):
Được tạo từ các lớp (classes). Ví dụ:

```
String str = "Hello";
```

```
Integer number = 100;
```

Mảng (Arrays):
Lưu trữ một danh sách các phần tử có cùng kiểu dữ liệu.
Ví dụ:

```
int[] numbers = {1, 2, 3, 4};
```

Interface:
Lưu trữ tham chiếu đến các lớp thực thi interface.

Kiểu generic:
Ví dụ như `List`, `Map` trong Java Collections Framework.

```
List<String> list = new ArrayList<>();
```

So sánh giữa kiểu nguyên thủy và tham chiếu

Đặc điểm	Kiểu nguyên thủy	Kiểu tham chiếu
Lưu trữ	Giá trị trực tiếp.	Địa chỉ của đối tượng trong bộ nhớ.
Tốc độ	Nhanh hơn, dùng ít bộ nhớ hơn.	Chậm hơn, do cần xử lý đối tượng.
Giá trị mặc định	0 hoặc `false` (tuỳ loại).	`null`.
Khả năng mở rộng	Không thể mở rộng.	Có thể mở rộng với các lớp tự định nghĩa.

II. Biến, hằng số

Biến là một vùng bộ nhớ trong chương trình dùng để lưu trữ dữ liệu và dữ liệu có thể thay đổi trong quá trình thực thi.

Cú pháp khai báo:

<kiểu_dữ_liệu> <tên_biến> = <giá_trị_khởi_tạo>;

Ví dụ:

int age = 25; // Biến kiểu số nguyên

String name = "John"; // Biến kiểu chuỗi

double pi = 3.14; // Biến kiểu số thực

boolean isJavaFun = true; // Biến kiểu boolean

Các loại biến:

1. Biến cục bộ (Local Variable):

- Được khai báo bên trong một phương thức, constructor hoặc block.
- Chỉ sử dụng được trong phạm vi nơi nó được khai báo.
- Không có giá trị mặc định, cần phải khởi tạo trước khi sử dụng.

```
public void printNumber() {
    int number = 10; // Biến cục bộ
    System.out.println(number);
}
```

2. Biến thành viên (Instance Variable):

- Được khai báo bên trong một lớp nhưng bên ngoài bất kỳ phương thức, constructor, hoặc block nào.
- Mỗi đối tượng của lớp sẽ có bản sao riêng của biến này.
- Có giá trị mặc định (0, null, false, ...).

```
class Person {
    String name; // Biến thành viên
}
```

3. Biến tĩnh (Static Variable):

- Được khai báo với từ khóa `static`.
- Thuộc về lớp, không phải đối tượng.
- Chia sẻ chung cho tất cả các đối tượng.

```
class MathUtils {
    static double pi = 3.14159; // Biến tĩnh
}
```

Hằng số

Hằng số là biến có giá trị cố định và không thể thay đổi sau khi được khởi tạo.

Cú pháp khai báo:

final <kiểu_dữ_liệu> <tên_hằng_số> = <giá_trị_khởi_tạo>;

Ví dụ:

```
final int MAX_AGE = 100;          // Hằng số kiểu số nguyên
final double PI = 3.14159;        // Hằng số kiểu số thực
final String GREETING = "Hello"; // Hằng số kiểu chuỗi
```

Đặc điểm:

- Sử dụng từ khóa final để khai báo.
- Tên hằng số thường viết hoa, cách nhau bởi dấu gạch dưới _ để dễ nhận biết.

Hằng số tĩnh:

Khi kết hợp từ khóa `static`, hằng số có thể được truy cập trực tiếp thông qua tên lớp, không cần tạo đối tượng.

```
class Constants {
    public static final double GRAVITY = 9.8;
}


System.out.println(Constants.GRAVITY);
```

So sánh Biến và Hằng số

Đặc điểm	Biến	Hằng số
Thay đổi giá trị	Có thể thay đổi	Không thể thay đổi sau khi khởi tạo
Từ khóa liên quan	Không có (trừ khi là `static` hoặc kiểu cụ thể)	`final` (và có thể kết hợp với `static`)
Phạm vi sử dụng	Phụ thuộc vào loại biến	Phạm vi phụ thuộc vào nơi khai báo

III. Các hàm xuất-nhập chuẩn

Trong Java, các hàm nhập xuất chuẩn (standard input/output) thường được sử dụng để làm việc với dữ liệu từ bàn phím hoặc xuất dữ liệu ra màn hình. Dưới đây là danh sách các hàm và lớp phổ biến liên quan đến xuất nhập trong Java:

1. Xuất dữ liệu

Sử dụng `System.out.print` **và** `System.out.println`

Hai phương thức này thường dùng để in dữ liệu ra màn hình:

- `System.out.print`: In mà không xuống dòng.
- `System.out.println`: In và xuống dòng.

```
public class OutputExample {
```

```
    public static void main(String[] args) {
        System.out.print("Hello, ");
        System.out.println("World!");
    }
}
```

Sử dụng `System.out.printf`

Hàm này cho phép định dạng chuỗi trước khi in.

```
public class PrintfExample {
    public static void main(String[] args) {
        int age = 25;
        double score = 85.75;
        System.out.printf("Tuổi: %d, Điểm: %.2f%n", age, score);
    }
}
```

2. Nhập dữ liệu

Sử dụng Scanner

Lớp Scanner trong gói java.util thường được sử dụng để đọc dữ liệu đầu vào từ bàn phím.

```
import java.util.Scanner;
public class InputExample {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Nhập một số nguyên: ");
        int number = scanner.nextInt();

        System.out.print("Nhập một chuỗi: ");
        scanner.nextLine(); // Loại bỏ dòng thừa
        String str = scanner.nextLine();
```

```
        System.out.println("Số bạn nhập: " + number);
        System.out.println("Chuỗi bạn nhập: " + str);

        scanner.close();
    }
}
```

IV. Toán tử

Toán tử trong Java là các ký hiệu hoặc từ khóa được sử dụng để thực hiện các phép toán trên các toán hạng (operand). Dưới đây là danh sách các nhóm toán tử phổ biến trong Java:

1. Toán tử số học

Dùng để thực hiện các phép toán cơ bản như cộng, trừ, nhân, chia.

Toán tử	Ý nghĩa	Ví dụ
`+`	Cộng	`a + b`
`-`	Trừ	`a - b`
`*`	Nhân	`a * b`
`/`	Chia	`a / b`
`%`	Chia lấy dư	`a % b`

2. Toán tử quan hệ (so sánh)

Dùng để so sánh hai giá trị.

Toán tử	Ý nghĩa	Ví dụ
`==`	Bằng nhau	`a == b`
`!=`	Không bằng	`a != b`
`>`	Lớn hơn	`a > b`

<	Nhỏ hơn	a < b
>=	Lớn hơn hoặc bằng	a >= b
<=	Nhỏ hơn hoặc bằng	a <= b

3. Toán tử logic

Dùng để thực hiện các phép toán logic.

Toán tử	Ý nghĩa	Ví dụ
&&	AND (và)	a > 0 && b > 0
||	OR (hoặc)	a > 0 || b > 0
!	NOT (phủ định)	!(a > 0)

4. Toán tử gán

Dùng để gán giá trị cho biến.

Toán tử	Ý nghĩa	Ví dụ
=	Gán giá trị	a = 5
+=	Cộng và gán	a += 5 (tương đương a = a + 5)
-=	Trừ và gán	a -= 5 (tương đương a = a - 5)
*=	Nhân và gán	a *= 5 (tương đương a = a * 5)
/=	Chia và gán	a /= 5 (tương đương a = a / 5)
%=	Chia lấy dư và gán	a %= 5 (tương đương a = a % 5)

5. Toán tử tăng/giảm

Dùng để tăng hoặc giảm giá trị của biến.

Toán tử	Ý nghĩa	Ví dụ
`++`	Tăng 1 đơn vị	`a++` hoặc `++a (tương đương a = a+1)`
`--`	Giảm 1 đơn vị	`a--` hoặc `–a (tương đương a = a – 1)`

6. Toán tử điều kiện (ternary)

Dùng để rút gọn các biểu thức điều kiện.

Toán tử	Ý nghĩa	Ví dụ
`?:`	Toán tử ba ngôi	`a = (b > 0) ? 1 : -1;`

7. Toán tử bitwise

Dùng để thao tác trên cấp độ bit.

Toán tử	Ý nghĩa	Ví dụ
`&`	AND	`a & b`
`` ` ``	`` ` ``	OR
`^`	XOR	`a ^ b`
`~`	NOT	`~a`
`<<`	Dịch trái	`a << 2`
`>>`	Dịch phải	`a >> 2`
`>>>`	Dịch phải (không dấu)	`a >>> 2`

8. Toán tử instanceof

Dùng để kiểm tra một đối tượng có thuộc kiểu dữ liệu cụ thể hay không.

```
if (obj instanceof String) {
```

```
    System.out.println("obj là chuỗi.");
}
```

BUỔI 3 : CẤU TRÚC ĐIỀU KHIỂN

I. Câu lệnh if-else, switch-case

Câu lệnh if-else

Dùng để kiểm tra một hoặc nhiều điều kiện và thực hiện các khối lệnh khác nhau dựa trên điều kiện đó.

Cú pháp:

```
if (điều_kiện) {
    // Khối lệnh thực hiện nếu điều kiện đúng
} else if (điều_kiện_khác) {
    // Khối lệnh thực hiện nếu điều kiện khác đúng
} else {
    // Khối lệnh thực hiện nếu tất cả điều kiện trên đều sai
}
```

Ví dụ:

```
int x = 10;
if (x > 0) {
    System.out.println("x là số dương");
} else if (x < 0) {
    System.out.println("x là số âm");
} else {
    System.out.println("x bằng 0");
}
```

Câu lệnh `switch-case`

Dùng để kiểm tra giá trị của một biểu thức và thực hiện các khối lệnh tương ứng với giá trị đó.

Cú pháp:

```
switch (biểu_thức) {
```

```
    case giá_trị_1:
        // Khối lệnh thực hiện nếu biểu_thức == giá_trị_1
        break;
    case giá_trị_2:
        // Khối lệnh thực hiện nếu biểu_thức == giá_trị_2
        break;
    ...
    default:
        // Khối lệnh thực hiện nếu không có giá trị nào khớp
        break;
}
```

Ví dụ:

```
int day = 3;
switch (day) {
    case 1:
        System.out.println("Chủ Nhật");
        break;
    case 2:
        System.out.println("Thứ Hai");
        break;
    case 3:
        System.out.println("Thứ Ba");
        break;
    default:
        System.out.println("Ngày không hợp lệ");
        break;
}
```

So sánh giữa `if-else` và `switch-case`:

Trong Java, các câu lệnh `if-else` và `switch-case` được sử dụng để kiểm soát luồng chương trình dựa trên các điều kiện. Dưới đây là chi tiết cách sử dụng chúng:

So sánh giữa `if-else` và `switch-case`:

Tiêu chí	if-else	switch-case
Sử dụng	Dùng khi có điều kiện phức tạp hoặc nhiều điều kiện.	Dùng khi kiểm tra giá trị của một biểu thức cụ thể.
Kiểu dữ liệu hỗ trợ	Hỗ trợ tất cả các kiểu dữ liệu.	Chỉ hỗ trợ `int`, `char`, `String`, `enum`, v.v.
Tính dễ đọc	Dễ bị rối khi có nhiều điều kiện.	Dễ đọc hơn khi có nhiều giá trị cần kiểm tra.
Hiệu suất	Chậm hơn khi số lượng điều kiện lớn.	Hiệu quả hơn với nhiều giá trị cố định.

II. Vòng lặp

Trong Java, các câu lệnh vòng lặp được sử dụng để lặp lại một đoạn mã nhiều lần cho đến khi một điều kiện cụ thể được đáp ứng. Có ba loại vòng lặp chính trong Java:

1. Vòng lặp `for`

Được sử dụng khi biết trước số lần lặp.

Cú pháp:

```
for (khởi_tạo; điều_kiện; bước_nhảy) {
    // Khối lệnh được thực hiện
}
```

Ví dụ:

```
for (int i = 0; i < 5; i++) {
    System.out.println("Giá trị của i là: " + i);
}
```

2. Vòng lặp `while`

Thực hiện khi điều kiện đúng, và dừng lại khi điều kiện sai.

Cú pháp:

```
while (điều_kiện) {
    // Khối lệnh được thực hiện
}
```

Ví dụ:

```
int i = 0;
while (i < 5) {
    System.out.println("Giá trị của i là: " + i);
    i++;
}
```

3. Vòng lặp `do-while`

Thực hiện ít nhất một lần, sau đó kiểm tra điều kiện.

Cú pháp:

```
do {
    // Khối lệnh được thực hiện
} while (điều_kiện);
```

Ví dụ:

```
int i = 0;
do {
    System.out.println("Giá trị của i là: " + i);
    i++;
} while (i < 5);
```

4. Vòng lặp `for-each` (sử dụng cho mảng hoặc danh sách)

Thích hợp để lặp qua các phần tử trong một mảng hoặc `Collection`.

Cú pháp:

```
for (kiểu_dữ_liệu biến : tập_hợp) {
```

```
    // Khối lệnh được thực hiện
}
```

Ví dụ:

```
int[] numbers = {1, 2, 3, 4, 5};
for (int number : numbers) {
    System.out.println("Số: " + number);
}
```

Lưu ý:

- **`break`**: Thoát khỏi vòng lặp ngay lập tức.
- **`continue`**: Bỏ qua phần còn lại của vòng lặp và chuyển sang lần lặp tiếp theo.

Ví dụ với `break` và `continue`:

```
for (int i = 0; i < 10; i++) {
    if (i == 5) {
        break; // Thoát khỏi vòng lặp khi i = 5
    }
    if (i % 2 == 0) {
        continue; // Bỏ qua nếu i là số chẵn
    }
    System.out.println(i);
}
```

Bài tập

1. Viết chương trình đọc vào 2 số nguyên, in ra kết quả của phép cộng (+), phép trừ (-), phép nhân (*), phép chia (/) của 2 số nguyên đó. Nhận xét kết quả phép chia hai số nguyên.
2. Viết chương trình:

    1. Tạo một biến `x = 10`.
    2. Tăng giá trị của `x` thêm 1 bằng cách sử dụng `++`.
    3. Giảm giá trị của `x` xuống 1 bằng cách sử dụng `--`.

4. Tăng giá trị của x lên 5 bằng cách sử dụng toán tử +=.
5. In kết quả sau mỗi bước.

3. Viết chương trình nhập vào năm sinh của một người và tính tuổi của người đó.
4. Viết chương trình nhập vào bán kính hình cầu, tính và in ra diện tích, thể tích của hình cầu đó. Công thức tính diện tích hình cầu $S = 4\pi R^2$ và công thức tính thể tích hình cầu $V = (4/3)\pi R^3$.
5. Viết chương trình nhập vào một số nguyên a bất kỳ và in ra giá trị bình phương (a2), lập phương (a3) của a.
6. Viết chương trình nhập từ bàn phím 3 số nguyên biểu diễn ngày, tháng, năm và xuất ra màn hình dưới dạng "ngay/thang/nam" (chỉ lấy 2 số cuối của năm).
7. Viết chương trình nhập vào số giây từ 0 đến 86399, đổi số giây nhập vào thành dạng "gio:phut:giay", mỗi thành phần là một số nguyên có hai chữ số.
8. Viết chương trình nhập vào điểm thi và hệ số 3 môn Toán, Lý, Hóa của một học sinh. Tính điểm trung bình của học sinh đó.
9. Viết chương trình nhập vào số xe (gồm 4 chữ số). Cho biết số xe đó được mấy nút?

10. Viết chương trình yêu cầu người dùng nhập một số nguyên.

- Nếu số đó chia hết cho 2, in ra "Số chẵn".
- Nếu không, in ra "Số lẻ".

11. Viết chương trình yêu cầu người dùng nhập ba số nguyên:

a) Kiểm tra xem số nào là lớn nhất.
b) Kiểm tra xem cả ba số có phải đều là số dương không.
c) Kiểm tra nếu ít nhất một trong ba số là số âm.

12. Viết chương trình kiểm tra xem một năm có phải là năm nhuận không.

Năm nhuận là năm chia hết cho 4 nhưng không chia hết cho 100, hoặc chia hết cho 400.

13. Viết chương trình nhập vào điểm thi của một học sinh (0-100). Sử dụng if-else để xếp loại học sinh theo các tiêu chí:
- **Giỏi**: Điểm từ 85 trở lên.
- **Khá**: Điểm từ 70 đến dưới 85.
- **Trung bình**: Điểm từ 50 đến dưới 70.
- **Yếu**: Điểm dưới 50.

14. Viết chương trình tính tiền taxi từ số km đã đi được nhập vào, biết:
    - 1 km đầu giá 5000đ.
    - Từ km thứ 2 – thứ 5: 4500 đ/km
    - Từ km thứ 6 trở đi: 3500 đ/km, đi hơn 120 km sẽ được giảm 10% trên tổng số tiền theo quy định.

15. Viết chương trình nhập vào một số nguyên từ 1 đến 7 và sử dụng switch-case để in ra ngày trong tuần tương ứng. Ví dụ:

    - 1 -> Chủ Nhật
    - 2 -> Thứ Hai
    - ...

16. Viết chương trình nhập vào hai số và một phép toán (+, -, *, /). Sử dụng switch-case để thực hiện phép tính tương ứng và in kết quả.
17. Viết chương trình in ra các số từ 1 đến 100 bằng các vòng lặp for, while, và do-while.
18. Viết chương trình nhập vào một số nguyên dương n từ bàn phím và tính tổng:
    Tổng=1+2+3+...+n
19. Viết chương trình nhập vào một số nguyên n, kiểm tra xem n có phải là số nguyên tố hay không. (Số nguyên tố là số lớn hơn 1 và chỉ chia hết cho 1 và chính nó.)
20. Viết chương trình in ra bảng cửu chương từ 1 đến 9.
21. Viết chương trình in ra dãy Fibonacci có n số đầu tiên.
    (Dãy Fibonacci: 0, 1, 1, 2, 3, 5, 8, ... trong đó mỗi số là tổng của 2 số trước đó.)
22. Viết chương trình nhập vào một số nguyên n. Kiểm tra xem số đó có phải là số đối xứng không.
    (Ví dụ: 121, 12321 là số đối xứng.)
23. Viết chương trình sử dụng vòng lặp for, in ra hình tam giác số như sau (với n = 5):

```
1
12
123
1234
12345
```

24. Viết chương trình sử dụng vòng lặp, in ra hình tam giác sao ngược (với n = 5):

```
*****
****
***
**
*
```

25. Viết chương trình nhập vào một số nguyên n. Viết chương trình in ra số đảo ngược của n. (Ví dụ: Nhập 1234 thì kết quả là 4321.)
26. Viết chương trình nhập vào một số nguyên n. Tìm tất cả các ước số của n.