# Bài 3
# Biến, Kiểu Dữ liệu và Toán tử

Môn học: PF-JAVA

# Mục tiêu

- Trình bày được khái niệm biến
- Trình bày được cú pháp khai báo biến
- Trình bày được khái niệm kiểu dữ liệu
- Trình bày được các toán tử thông dụng
- Khai báo và sử dụng được biến
- Sử dụng được các kiểu dữ liệu
- Sử dụng được các toán tử cơ bản

# Biến và hằng

Định danh

Biến

Hằng

Quy ước đặt tên

# Định danh (Identifier)

- Định danh là tên gọi để chỉ đến các thành phần như lớp (class), phương thức (method), biến…
- Chẳng hạn *AppleProgram*, *main*, *args*, *totalApples*, *numberOfBaskets*, *applePerBasket* là các định danh:

```
public class AppleProgram {
    public static void main(String[] args) {
        int totalApples;
        int numberOfBaskets = 5;
        int applePerBasket = 10;

        totalApples = numberOfBaskets * applePerBasket;
        System.out.println("Number of apples is " + totalApples);
    }
}
```

# Quy tắc của định danh

- Chỉ được phép bao gồm: chữ cái, chữ số, dấu gạch dưới (_) và dấu dollar ($)
- Phải bắt đầu bằng chữ cái, dấu gạch dưới (_) hoặc dấu dollar ($)
- Không được sử dụng từ khoá (key word/reserved word), chẳng hạn như *int, float, public, class…*
- Không được đặt định danh *true, false, null*
- Không hạn chế độ dài của một định danh
- Java phân biệt định danh chữ hoa (uppercase) và chữ thường (lowercase)

# Định danh: Ví dụ

- Các định danh sau đây là hợp lệ:

  **$2**

  **_length**
  **ComputeArea**
  **area**
  **radius**
  **print**

- Các định danh sau đây là không hợp lệ:

  **2A**

  **d+4**

  **my color**

# Biến (variable)

- **Biến** là một tên gọi được gắn cho một vùng nhớ chứa dữ liệu
- Dữ liệu được lưu trữ trong vùng nhớ của biến được gọi là **giá trị** (value)
- Có thể truy nhập, gán hay thay đổi giá trị của biến
- Khi gán một giá trị mới thì giá trị cũ sẽ bị ghi đè lên
- Java yêu cầu phải khai báo biến trước khi sử dụng
- Chẳng hạn:

```
int x;     // khai báo
x = 10;    // sử dụng
```

# Khai báo biến

- Cú pháp:

*datatype variableName;*

Trong đó:

- datatype là kiểu dữ liệu của biến
- variableName là định danh (tên) của biến

- Có thể khai báo nhiều biến cùng kiểu giá trị trong một câu lệnh:

*datatype variable1, variable2, ..., variablen;*

- Ví dụ:

```
int i, j, k; //Khai báo các biến i, j, k là kiểu số nguyên
```

# Gán giá trị cho biến

- Có thể gán giá trị cho biến ngay tại thời điểm khai báo
- Ví dụ:

```
int count = 1;
```

- Ví dụ trên tương đương với:

```
int count;
count = 1;
```

- Có thể gán giá trị cho nhiều biến tại thời điểm khai báo
- Ví dụ:

```
int i = 1, j = 2;
```

# Lưu ý về biến

- Một biến cần được khai báo trước khi được gán giá trị
- Một biến khai báo trong phương thức thì cần được gán giá trị trước khi được sử dụng
- Nếu có thể thì hãy thực hiện việc khai báo và gán giá trị trong cùng một câu lệnh, thay vì tách rời chúng

# Hằng (constant)

- Hằng là một tên gọi đại diện cho một giá trị cố định
- Giá trị của hằng không thể thay đổi
- Giá trị của hằng cần phải được gán tại thời điểm khai báo
- Ví dụ, sử dụng hằng PI thay cho giá trị 3.14159:

```
double area = radius * radius * 3.14159;
```

Được thay bằng:

```
final double PI = 3.14159;
double area = radius * radius * PI;
```

# Khai báo hằng

- Cú pháp khai báo hằng:

```
final datatype CONSTANTNAME = value;
```

Trong đó:

- *final* là từ khoá bắt buộc để khai báo hằng
- *datatype* là kiểu dữ liệu của hằng
- *CONSTANTNAME* là tên của hằng
- *value* là giá trị của hằng

# Quy ước khi đặt tên (namming convention)

- Quy ước là những luật lệ không bắt buộc nhưng nên tuân thủ khi lập trình
- Tuân thủ các quy ước sẽ giúp cho mã nguồn dễ đọc hơn
- Một số quy ước đặt tên thông dụng:
  - Sử dụng tên gọi có ý nghĩa
  - Sử dụng tên viết thường cho biến và phương thức, ví dụ: **radius, area, print**
  - Nếu tên biến hoặc tên phương thức có nhiều từ thì viết hoa các chữ cái đầu tiên của các từ ở sau, ví dụ: **fristName, calculateArea, printMyName**
  - Viết hoa chữ cái đầu tiên của tên lớp, ví dụ: ComputeArea, System
  - Viết hoa toàn bộ tên gọi của hằng, ví dụ: **PI, MAX_VALUE, COUNTRY_CODE**

# Kiểu dữ liệu

Các kiểu dữ liệu nguyên thuỷ

Chuỗi

Giá trị mặc định

# Kiểu dữ liệu

- Các kiểu dữ liệu là cách để phân loại các dữ liệu ở trong chương trình
- Cần chỉ rõ kiểu dữ liệu khi khai báo biến hoặc hằng
- Kiểu dữ liệu quy định dung lượng bộ nhớ sẽ cấp cho biến
- Kiểu dữ liệu quy định các thao tác có thể thực hiện với biến
- Ví dụ:

```
int gear = 1;
```

Biến *gear* có kiểu dữ liệu là số nguyên (integer), chỉ được phép gán các giá trị số nguyên cho biến gear.

# 8 kiểu dữ liệu nguyên thuỷ (primitive datatype)

| Kiểu dữ liệu | Mô tả |
|---|---|
| **byte** | Kiểu số nguyên có kích thước 1 byte. Các giá trị nằm trong khoảng -128 đến 127. |
| **short** | Kiểu số nguyên có kích thước 2 byte. Các giá trị nằm trong khoảng -32768 đến 32767. |
| **int** | Kiểu số nguyên có kích thước 4 byte.Các giá trị nằm trong khoảng $-2^{31}$ đến $2^{31}$ -1. |
| **long** | Kiểu số nguyên có kích thước 8 byte. Các giá trị nằm trong khoảng $-2^{63}$ đến $2^{63}$ - 1. |
| **float** | Kiểu số thực có kích thước 4 byte. |
| **double** | Kiểu số thực có kích thước 8 byte. |
| **boolean** | Bao gồm 2 giá trị là true và false. |
| **char** | Kiểu ký tự Unicode có kích thước 2 byte. Có giá trị nhỏ nhất là '\u0000' (tương đương với 0) và giá trị lớn nhất là '\uffff' (tương đương với 65535) |

# Kiểu dữ liệu chuỗi (String)

- Chuỗi ký tự (gọi ngắn gọn là Chuỗi) bao gồm các ký tự liên tiếp nhau
- Java cung cấp lớp String để thao tác với chuỗi
- Chuỗi là các giá trị cố định (immutable), không thể thay đổi sau khi khởi tạo
- Ví dụ khai báo chuỗi:

```
String name = "John";

String countryName = new String("Vietnam");
```

# Giá trị mặc định

- Khi khai báo một biến của đối tượng (không phải biến địa phương) mà không gán giá trị cho nó thì nó sẽ có giá trị mặc định

| Data Type | Default Value (for fields) |
|---|---|
| byte | 0 |
| short | 0 |
| int | 0 |
| long | 0L |
| float | 0.0f |
| double | 0.0d |
| char | '\u0000' |
| String (or any object) | null |
| boolean | false |

# Kiểu dữ liệu: Ví dụ

```
byte age = 100;
short code = 1001;
int numberOfFriends = 400000;
long numberOfCellulars = 908766861;
float salary = 24345454.562f;
double exactLength =
1824378123.28634431424;
boolean isValid = true;
char command = 'G';
String message = "Success";
```

# Chuyển đổi kiểu (conversion)

- Trong các biểu thức mà có sự tham gia của các giá trị thuộc các kiểu dữ liệu khác nhau thì các kiểu dữ liệu kích thước nhỏ thì sẽ được tự động chuyển sang kiểu dữ liệu có kích thước lớn hơn
- Việc chuyển đổi này được thực hiện ngầm định (implicit)
- Chẳng hạn, kiểu **short** được tự động chuyển sang kiểu **int**, kiểu **int** tự động chuyển sang kiểu **double**…

```
short a = 15;
int b = a + 1;
double c = b + 1.5;
```

# Ép kiểu

- Nếu muốn chuyển giá trị thuộc kiểu dữ liệu kích thước lớn sang kiểu dữ liệu kích thước nhỏ hơn thì cần ép kiểu tường mình (explicit)
- Ví dụ, câu lệnh sau có lỗi do kiểu **int** lớn hơn kiểu **short**:

```
int a = 10;
short b = a + 1;
```

- Sử dụng ép kiểu từ **float** xuống **int**:

```
float a = 10;
int b = (int) a + 1;
```

# Lưu ý: phép toán số học

- Các phép toán số học (ví dụ như “+”) sẽ cho kết quả là một kiểu dữ liệu **int** hoặc **long**
- Do đó, đoạn mã sau có lỗi:

```
short a = 5;
short b = 10;
short c = a + b;
```

*//Bởi vì biểu thức **a + b** sẽ trả về là một giá trị kiểu **int**. Do đó không thể gán cho biến **c** có kiểu dữ liệu là **short** (nhỏ hơn **int**)*

- Có thể sửa lại như sau:

```
short c = (short)(a + b);
```

# Các toán tử

Toán tử gán

Toán tử số học

Toán tử so sánh

Toán tử logic

# Phép gán

- Phép gán (assignment) được sử dụng để gắn giá trị cho một biến
- Cú pháp:

**variable = expression;**

Trong đó:

- *variable* là tên của biến
- *expression* là biểu thức bao gồm giá trị, biến và các phép toán

- Ví dụ:

```
int y = 1;
double radius = 1.0;
int x = 5 * (3 / 2);
x = y + 1;
double area = radius * radius * 3.14159;
```

# Gán giá trị cho nhiều biến

- Ví dụ gán giá trị cho nhiều biến:

```
int i, j, k;
i = j = k = 1;
```

- **Lưu ý:** Kiểu dữ liệu của biến (bên vế trái) phải thích hợp với giá trị được gán (bên vế phải)

  Phép gán: **int i = 1.5** là không hợp lệ bởi vì biến i có kiểu dữ liệu là số nguyên (integer) trong khi đó giá trị 1.5 là số thực (double)

# Toán tử số học

| Toán tử | Mô tả | Ví dụ |
|---|---|---|
| + | Phép cộng | int result = 1 + 2; //result = 3 |
| - | Phép trừ | int result = 1 - 2; //result = -1 |
| * | Phép nhân | int result = 2 * 3; //result = 6 |
| / | Phép chia | int result = 3 / 2; //result = 1 |
| % | Phép chia lấy số dư | int result = 5 % 3; //result = 2 |

# Toán tử một ngôi

| Toán tử | Mô tả | Ví dụ |
|---|---|---|
| + | Toán tử cộng. | *int result = +1; //result = 1* |
| - | Toán tử trừ | *int result = -1; //result = -1* |
| ++ | Toán tử tăng 1 giá trị | *int result = 1;*<br>*int result++; //result = 2* |
| -- | Toán tử giảm 1 giá trị | *int result = 1;*<br>*int result--; //result = 0* |
| ! | Toán tử phủ định | *int value = true;*<br>*result = !value;//result – false* |

# Toán tử tăng và giảm

- Toán tử tăng (++) và giảm (--) sẽ cho kết quả khác nhau, tuỳ thuộc vào vị trí của nó so với toán hạng
- Nếu đặt trước (prefix) toán hạng thì giá trị sẽ được tăng hoặc giảm trước khi biểu thức được đánh giá
- Nếu đặt sau (postfix) toán hạng thì giá trị sẽ được tăng hoặc giảm sau khi biểu thức được đánh giá
- Ví dụ:

```
int i = 3;
int j = i++;// i = 4 và j = 3;
```

```
int i = 3;
int j = ++i;// i = 4 và j = 4;
```

# Toán tử so sánh (Comparission)

| Toán tử | Mô tả | Ví dụ |
|---|---|---|
| | | **int** a = 5;<br>**int** b = 6; |
| == | So sánh bằng | boolean result = a == b; //result = false |
| != | So sánh khác | boolean result = a != b; //result = true |
| > | Lớn hơn | boolean result = a > b; //result = false |
| >= | Lớn hơn hoặc bằng | boolean result = a >= b; //result = false |
| < | Nhỏ hơn | boolean result = a < b; //result = true |
| <= | Nhỏ hơn hoặc bằng | boolean result = a <= b; //result = true |

# Toán tử logic

| Toán tử | Mô tả | Ví dụ |
|---|---|---|
| | | **int** a = true;<br>**int** b = false; |
| && | AND (và): Trả về đúng nếu cả 2 vế đều đúng | boolean result = a && b; //result = false |
| \|\| | Trả về đúng nếu ít nhất một vế đúng | boolean result = a \|\| b; //result = true |
| ! | Phủ định | boolean result = !a; //result = false |

# Phép toán AND (&&)

| a | b | a && b |
|---|---|---|
| true | true | true |
| true | false | false |
| false | true | false |
| false | false | false |

# && và &

- && và & đều là hai phép toán AND
- Điểm khác biệt:
  - & bắt buộc cả 2 vế đều được đánh giá
  - && sẽ bỏ qua vế bên phải nếu về bên trái đã trả về false
- Đối với các trường hợp thông thường, nên sử dụng &&

# && và &: Ví dụ

```
int number1 = 5;
if(number1 > 5 && number1++ > 5){
    System.out.println("Is true");
} else {
    System.out.println("Is false. number1 = " + number1);
}
```

Biểu thức này được bỏ qua

*Is false. number1 = 5*

```
int number1 = 5;
if(number1 > 5 & number1++ > 5){
    System.out.println("Is true");
} else {
    System.out.println("Is false. number1 = " + number1);
}
```

Biểu thức này được thực thi

*Is false. number1 = 6*

# Phép toán OR (||)

| a | b | a \|\| b |
|---|---|---|
| true | true | true |
| true | false | true |
| false | true | true |
| false | false | false |

# || và |

- || và | đều là hai phép toán OR
- Điểm khác biệt:
  - | bắt buộc cả 2 vế đều được đánh giá
  - || sẽ bỏ qua vế bên phải nếu về bên trái đã trả về true
- Đối với các trường hợp thông thường, nên sử dụng ||

# || và |: Ví dụ

```
int number1 = 5;

if(number1 < 6 || number1++ > 5){
    System.out.println("Is true. number1 = " + number1);
}
```

Biểu thức này được bỏ qua

*Is true. number1 = 5*

```
int number1 = 5;

if(number1 < 6 | number1++ > 5){
    System.out.println("Is true. number1 = " + number1);
}
```

Biểu thức này được thực thi

*Is true. number1 = 6*

# Độ ưu tiên của các toán tử

- Trong một biểu thức có nhiều phép toán thì chúng sẽ lần lượt được đánh giá dựa vào độ ưu tiên
- Có thể sử dụng dấu ngoặc "()" để thay đổi độ ưu tiên của các toán tử
- Các toán tử có cùng độ ưu tiên thì sẽ thực hiện từ trái sang phải

| Operators | Precedence |
|---|---|
| postfix | expr++ expr-- |
| unary | ++expr --expr +expr -expr ~ ! |
| multiplicative | * / % |
| additive | + - |
| shift | << >> >>> |
| relational | < > <= >= instanceof |
| equality | == != |
| bitwise AND | & |
| bitwise exclusive OR | ^ |
| bitwise inclusive OR | \| |
| logical AND | && |
| logical OR | \|\| |
| ternary | ? : |
| assignment | = += -= *= /= %= &= ^= \|= <<= >>= >>>= |

# Độ ưu tiên của các toán tử: Ví dụ

```
int x = 5;
int y = 10;
boolean z = (++x * y--) < 5*10 && 6 > 3;
```

- Giá trị của biến z là false

# [Thực hành] Khai báo và sử dụng biến

# [Thực hành] Sử dụng toán tử

# [Bài tập] Ứng dụng chuyển đổi tiền tệ

# [Bài tập] Luyện tập sử dụng biến, kiểu dữ liệu và toán tử

# Tổng kết

- Biến là một tên gọi đại diện cho một vùng nhớ chứa giá trị
- Hằng là một tên gọi đại diện cho một giá trị cố định
- Kiểu dữ liệu quy định các giá trị mà một biến có thể nhận được và các thao tác có thể thực hiện với biến đó
- Java hỗ trợ nhiều kiểu dữ liệu khác nhau
- Có nhiều loại toán tử khác nhau, chẳng hạn: Phép gán, số học, so sánh, logic
- Các toán tử sẽ được đánh giá lần lượt phụ thuộc vào độ ưu tiên của chúng