Bài 03:
Kiểu dữ liệu – Biến & hằng – Các phép toán – Ép kiểu

Giảng Viên: ThS. Giang Hào Côn

3.1/ Các kiểu dữ liệu (Data Type)

- Trong Java, tất cả đầu vào, đầu ra phải được khai báo trước khi sử dụng. Khi khai báo, cần chỉ rõ **loại dữ liệu** có thể được lưu trữ trong đầu vào, đầu ra thông qua kiểu dữ liệu của chúng. Trong Java, các kiểu dữ liệu được chia thành 2 loại:
- **Kiểu dữ liệu primitive** (kiểu dữ liệu nguyên thủy hay kiểu dữ liệu cơ bản) được định nghĩa sẵn trong Java. Gồm **8 kiểu dữ liệu** là boolean, byte, short, int, long, double, float, char.
- **Kiểu dữ liệu non-primitive** thường là các lớp (class) do lập trình viên tự định nghĩa (ngoại trừ String, Arrays,..).

3.1.1/ Kiểu dữ liệu boolean

- Đây là kiểu dữ liệu chỉ nhận một trong 2 giá trị true hoặc false. Giá trị mặc định là false.

```
class Main {
    public static void main(String[] args) {
        boolean flag = true;
        System.out.println(flag);//true
    }
}
```

3.1.2/ Kiểu dữ liệu byte

- Kiểu dữ liệu byte có giá trị từ -128 đến 127, được biểu diễn bởi 8 bit. Giá trị mặc định là 0.
- Nếu chắc chắn rằng giá trị của một biến sẽ nằm trong khoảng -128 đến 127 thì nên sử dụng kiểu byte thay kiểu int để tiết kiệm bộ nhớ.

```
class Main {
    public static void main(String[] args) {
        byte range;
        range = 124;
        System.out.println(range);//124
    }
}
```

3.1.3/ Kiểu dữ liệu short

- Kiểu dữ liệu short có giá trị từ -32768 đến 32767, được biểu diễn bởi 16 bit. Giá trị mặc định là 0.
- Nếu chắc chắn rằng giá trị của một biến sẽ nằm trong khoảng -32768 đến 32767 thì nên sử dụng kiểu short thay kiểu int hoặc long để tiết kiệm bộ nhớ..

```
class Main {
    public static void main(String[] args) {
        short temperature;
        temperature = -200;
        System.out.println(temperature);//-200
    }
}
```

3.1.4/ Kiểu dữ liệu int

- Kiểu dữ liệu int có giá trị từ -2^{31} to $2^{31}-1$, được biểu diễn bởi 32 bit. Giá trị mặc định là 0.

```
class Main {
    public static void main(String[] args) {
        int range = -4250000;
        System.out.println(range);//-4250000
    }
}
```

3.1.5/ Kiểu dữ liệu long

- Kiểu dữ liệu long có giá trị từ -2^{63} to $2^{63}-1$, được biểu diễn bởi 64 bit. Giá trị mặc định là 0..

```
class Main {
    public static void main(String[] args) {
        long range = -42332200000L;
        System.out.println(range);//-42332200000
    }
}
```

3.1.6/ Kiểu dữ liệu double

- Kiểu dữ liệu double lưu trữ số thực với dấu chấm động (floating-point), được biểu diễn bởi 64 bit. Giá trị mặc định là 0.0 (0.0d).

```
class Main {
    public static void main(String[] args) {
        double number = -42.3d;
        System.out.println(number);//-42.3
    }
}
```

3.1.7/ Kiểu dữ liệu float

- Kiểu dữ liệu float lưu trữ số thực với dấu chấm động (floating-point), được biểu diễn bởi 32 bit. Giá trị mặc định là 0.0 (0.0f).

```
class Main {
    public static void main(String[] args) {
        float number = -42.3f;
        System.out.println(number);//-42.3
    }
}
```

3.1.8/ Kiểu dữ liệu char

- Kiểu dữ liệu char dùng để lưu trữ các ký tự Unicode, được biểu diễn bởi 16 bit. Giá trị mặc định là '\u0000'.
- Giá trị ký tự nhỏ nhất của kiểu char là '\u0000' (0) và giá trị ký tự lớn nhất là '\uffff'.

```
class Main {
    public static void main(String[] args) {
        char letter = '\u0051';
        System.out.println(letter);//ký tự Q

        char letter1 = '9';
        System.out.println(letter1);//9

        char letter2 = 65;
        System.out.println(letter2);//A
    }
}
```

Nếu một số nguyên được gán cho biến char (ví dụ char letter2 = 65;), Java sẽ lấy ký tự tương ứng số nguyên đó trong bảng mã ASCII. Ví dụ, trong bảng mã ASCII thì 65 tương ứng ký tự 'A'.

3.1.9/ Kiểu dữ liệu String

- Java hỗ trợ kiểu dữ liệu String, lưu trữ chuỗi ký tự thông qua lớp java.lang.String. Đây không phải là kiểu dữ liệu cơ bản mà nó là một lớp được định nghĩa sẵn trong Java.

```
class Main {
    public static void main(String[] args) {
        String myString = "Java Programming";
        System.out.println(myString);//Java Programming
    }
}
```

3.1.9/ Kiểu dữ liệu String

- Ghép chuỗi với String.

```
String message = "Welcome " + "to " + "Java!";//Welcome to Java!
String s = "Chuong " + 2;//Chuong 2
String s1 = "Hello" + " World!";//Hello World!
String s2 = s + " - " + s1;//Chuong 2 - Hello World!
```

- Ghép chuỗi ký tự thành số.

```
int intValue = Integer.parseInt("123");//123
double doubleValue = Double.parseDouble("123.45");//123.45
```

Mỗi số lỗi về kiểu dữ liệu phổ biến

- Chia số nguyên ngoài ý muốn

```
int number1 = 1;
int number2 = 2;
double average2 = (number1 + number2)/2;
System.out.println(average2);//1.0
double average20 = (number1 + number2)/2.0;
System.out.println(average20);//1.5
```

3.2/ Biến (variable) & Hằng (constant)

1. Khai báo biến trong Java

Một biến (variable) sẽ được cấp phát một vùng nhớ trong bộ nhớ để lưu trữ dữ liệu. Mỗi biến phải được đặt một tên duy nhất (identifier). Cú pháp khai báo biến:

```
datatype variableName;
```


```
//Khai bao x la mot bien nguyen (integer)
int x;
//Khai bao bankinh la mot bien so thuc (double)
double bankinh;
//Khai bao a la mot bien ky tu (char)
char a;
```

3.2/ Biến (variable) & Hằng (constant)

2. Quy ước đặt tên biến (identifier)

– Tên biến là một chuỗi các ký tự gồm các chữ, số, dấu gạch dưới (_), và dấu dollar ($). Tên biến không thể là một từ khóa. Tên biến không thể là true, false hoặc null. Tên biến có thể có độ dài bất kỳ.

– Tên biến phải bắt đầu bởi một chữ, dấu gạch dưới (_), hoặc dấu dollar ($), không thể bắt đầu bởi một số.

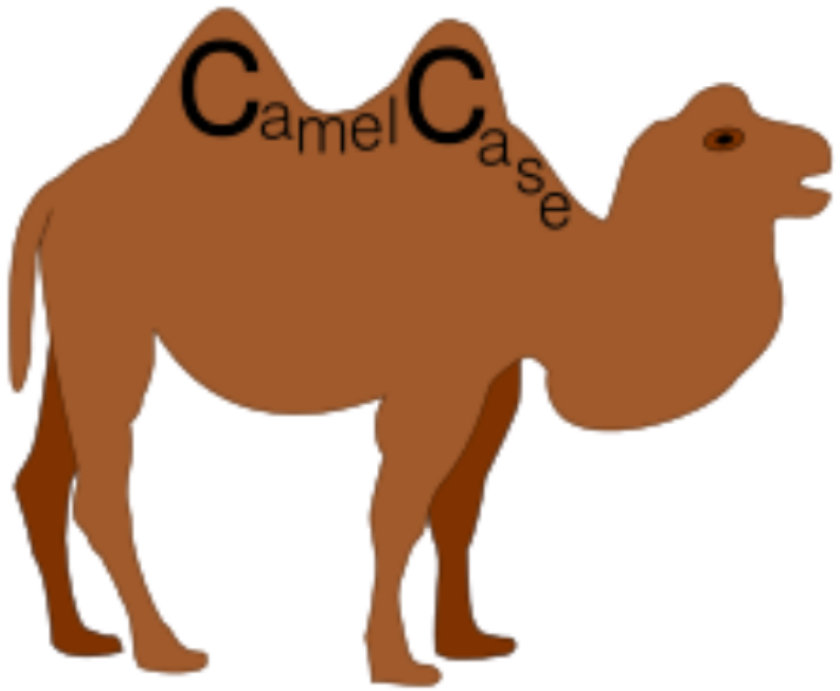
```
int age;  // valid
int _age;  // valid
int $age;  // valid
int 1age;  // invalid
```

3.2/ Biến (variable) & Hằng (constant)

2. Quy ước đặt tên biến (identifier)

Camel case

- Tuân theo nguyên tắc về đặt tên cho Identified trong lập trình
- Bắt đầu các từ phải là viết chữ hoa, các ký tự còn lại phải là chữ thường, *duy nhất từ đầu tiên phải viết chữ thường toàn bộ*. VD **iPhone**, **eBay**, **fedEx**, **chieuDai**, **tyGia**, …



3.2/ Biến (variable) & Hằng (constant)

2. Quy ước đặt tên biến (identifier)

- Trong tên biến không được sử dụng khoảng trắng

```
int my age;  // invalid variables
```

- Trong Java, tên biến có phân biệt chữ hoa chữ thường. Ví dụ, biến age và AGE là 2 biến khác nhau.

```
int age = 24;
int AGE = 25;

System.out.println(age);  // print 24
System.out.println(AGE);  // print 25
```

3.2/ Biến (variable) & Hằng (constant)

3. Gán giá trị cho biến (variable) trong Java

Các biến trong Java có thể được gán giá trị với toán tử bằng **"="**. Giá trị của biến có thể thay đổi trong chương trình. Cú pháp lệnh gán:

```
variable = expression;
```

Ví dụ:

```
x = 1;// Gan so nguyen 1 cho bien x
bankinh = 1.0;// Gán so thuc 1.0 cho bien ban kinh
a = 'A';// Gán ký tự 'A' cho bien a
```

3.2/ Biến (variable) & Hằng (constant)

3. Gán giá trị cho biến (variable) trong Java

Các loại giá trị có thể gán trực tiếp cho biến trong Java là:

Giá trị luận lý (boolean): Có thể gán giá trị true hoặc false cho biến kiểu boolean.

```
boolean flag1 = false;
boolean flag2 = true;
```

Giá trị số nguyên (interger): Có thể gán các số binary (base 2), decimal (base 10), octal (base 8), hexadecimal (base 16) cho biến trong Java.

```
// binary
int binaryNumber = 0b10010;
// octal
int octalNumber = 027;
```

```
// hexadecimal
int hexNumber = 0x2F; // 0x represents hexadecimal
// binary
int binNumber = 0b10010; // 0b represents binary
```

3.2/ Biến (variable) & Hằng (constant)

3. Gán giá trị cho biến (variable) trong Java

Các loại giá trị có thể gán trực tiếp cho biến trong Java là:

Giá trị số thực (float): Các biến trong Java có thể được gán bởi số thực với dấu chấm động (floating point).

```
double myDouble = 3.4;
float myFloat = 3.4F;

// 3.445*10^2
double myDoubleScientific = 3.445e2;
```

3.2/ Biến (variable) & Hằng (constant)

3. Gán giá trị cho biến (variable) trong Java

Các loại giá trị có thể gán trực tiếp cho biến trong Java là:

Giá trị ký tự (char): Các biến trong Java có thể được gán bởi các ký tự. Lưu ý: Một số ký tự đặc biệt cũng có thể được gán cho biến trong Java như \b (backspace), \t (tab), \n (new line),...

```
char letter = 'a';
```

3.2/ Biến (variable) & Hằng (constant)

3. Gán giá trị cho biến (variable) trong Java

Các loại giá trị có thể gán trực tiếp cho biến trong Java là:

Giá trị chuỗi (string): Một chuỗi ký tự cũng có thể được gán cho biến trong Java.

```
String str1 = "Java Programming";
String str2 = "Gochocit";
```

3.2/ Biến (variable) & Hằng (constant)

4. Khai báo và khởi tạo biến trong một dòng lệnh

Cú pháp:

```
datatype variableName = expression;
```

Ví dụ:

```
int x = 1;
double d = 3.6;
```

3.3/ Hằng (constant) trong Java

Hằng (constant) là một loại biến đặc biệt mà giá trị của nó sẽ không thể thay đổi. Chúng ta sử dụng từ khóa final để khai báo hằng trong Java. Cú pháp:

```
final datatype CONSTANTNAME = VALUE;
```

Ví dụ:

```
final double PI = 3.14159;
final int SIZE = 3;
```

3.4/ Nhập xuất cơ bản trong Java

1) Xuất (output) trong Java

```
System.out.println();
hoặc
System.out.print();
hoặc
System.out.printf();
```

```
class Main {
    public static void main(String[] args) {
        System.out.println("Java programming is interesting.");
    }
}
```

```
class Main {
    public static void main(String[] args) {
        System.out.println("1. println ");
        System.out.println("2. println ");

        System.out.print("1. print ");
        System.out.print("2. print");
    }
}
```

Kết quả

```
1. println
2. println
1. print 2. Print
```

Sự khác nhau giữa các hàm println(), print() và printf() ?

3.4/ Nhập xuất cơ bản trong Java

1) Xuất (output) trong Java

```
class Main {
    public static void main(String[] args) {
        Double number = -10.6;

        System.out.println(5);
        System.out.println(number);

        System.out.println("I am " + "awesome.");
        System.out.println("Number = " + number);
    }
}
```

Kết quả

```
5
-10.6
I am awesome.
Number = -10.6
```

3.4/ Nhập xuất cơ bản trong Java

2) Nhập (input) trong Java

Trong Java, chúng ta có thể sử dụng đối tượng của lớp Scanner để lấy giá trị được nhập vào bởi người dùng. Đầu tiên, cần **import** gói java.util.Scanner.

```
import java.util.Scanner;
```

Sau đó, cần tạo một đối tượng của lớp Scanner. Đối tượng này sẽ giúp chúng ta lấy giá trị nhập vào của người dùng.

```
// create an object of Scanner
Scanner input = new Scanner(System.in);

// take input from the user
int number = input.nextInt();
```

3.4/ Nhập xuất cơ bản trong Java

2) Nhập (input) trong Java

Các phương thức của lớp Scanner

Phương thức	Mô tả
nextByte()	*Đọc một số nguyên kiểu byte*
nextShort()	*Đọc một số nguyên kiểu short*
nextInt()	*Đọc một số nguyên kiểu int*
nextLong()	*Đọc một số nguyên kiểu long*
nextFloat()	*Đọc một số kiểu float*
nextDouble()	*Đọc một số kiểu double*
next()	*Đọc một string kết thúc trước một ký tự trắng*
nextLine()	*Đọc một line of text (kết thúc bằng phím Enter)*

3.4/ Nhập xuất cơ bản trong Java

2) Nhập (input) trong Java

Một số ví dụ nhập trong Java sử dụng lớp Scanner

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

        System.out.print("Enter an integer: ");
        int number = input.nextInt();
        System.out.println("You entered " + number);
```

3.4/ Nhập xuất cơ bản trong Java

2) Nhập (input) trong Java

Một số ví dụ nhập trong Java sử dụng lớp Scanner

```
// Getting float input
System.out.print("Enter float: ");
float myFloat = input.nextFloat();
System.out.println("Float entered = " + myFloat);

// Getting double input
System.out.print("Enter double: ");
double myDouble = input.nextDouble();
System.out.println("Double entered = " + myDouble);
```

3.4/ Nhập xuất cơ bản trong Java

2) Nhập (input) trong Java

Một số ví dụ nhập trong Java sử dụng lớp Scanner

```
        // Getting String input
        System.out.print("Enter text: ");
        String myString = input.next();
        System.out.println("Text entered = " + myString);

        // closing the scanner object
        input.close();
    }
}
```

3.5/ Các toán tử (operators) trong Java

Các toán tử trong in Java có thể được chia thành các loại toán tử sau:

- Toán tử số học (Arithmetic Operators)
- Toán tử gán (Assignment Operators)
- Toán tử quan hệ (Relational Operators)
- Toán tử logic (Logical Operators)
- Toán tử một ngôi (Unary Operators)
- Toán tử trên bit (Bitwise Operators)

3.5/ Các toán tử (operators) trong Java

1) Toán tử số học (Arithmetic Operators)

Ký hiệu	Phép toán	Ví dụ	Kết quả
+	*Cộng (Addition)*	25 + 3	28
–	*Trừ (Subtraction)*	25.0 – 0.3	24.7
*	*Nhân (Multiplication)*	25 * 3	75
/	*Chia (Division)*	5.0 / 2.0	2.5
%	*Chia dư (Remainder)*	25 % 3	1

3.5/ Các toán tử (operators) trong Java

1) Toán tử số học (Arithmetic Operators)

```
class Main {
    public static void main(String[] args) {
        // declare variables
        int a = 12, b = 5;

        // addition operator
        System.out.println("a + b = " + (a + b));

        // subtraction operator
        System.out.println("a - b = " + (a - b));

        // multiplication operator
        System.out.println("a * b = " + (a * b));

        // division operator
        System.out.println("a / b = " + (a / b));

        // modulo operator
        System.out.println("a % b = " + (a % b));
    }
}
```

Kết quả

```
a + b = 17
a - b = 7
a * b = 60
a / b = 2
a % b = 2
```

3.5/ Các toán tử (operators) trong Java

2) Toán tử gán (Assignment Operators)

Một số toán tử gán tắt

Ký hiệu	Phép toán	Ví dụ	Kết quả
+=	*Gán cộng (Addition assignment)*	*i += 3*	*i = i + 3*
-=	*Gán trừ (Subtraction assignment)*	*i -= 3*	*i = i – 3*
*=	*Gán nhân (Multiplication assignment)*	*i *= 3*	*i = i * 3*
/=	*Gán chia (Division assignment)*	*i /= 3*	*i = i / 3*
%=	*Gán chia dư (Remainder assignment)*	*i %= 3*	*i = i % 3*

3.5/ Các toán tử (operators) trong Java

2) Toán tử gán (Assignment Operators)

```
class Main {
    public static void main(String[] args) {
        // create variables
        int a = 4;
        int var;

        // assign value using =
        var = a;
        System.out.println("var using =: " + var);

        // assign value using =+
        var += a;
        System.out.println("var using +=: " + var);

        // assign value using =*
        var *= a;
        System.out.println("var using *=: " + var);
    }
}
```

Kết quả

```
var using =: 4
var using +=: 8
var using *=: 32
```

3.5/ Các toán tử (operators) trong Java

3) Toán tử quan hệ (Relational Operators)

Ký hiệu	Phép toán	Ví dụ (bankinh = 5)	Kết quả
<	*Nhỏ hơn*	*bankinh < o*	*false*
<=	*Nhỏ hơn hoặc bằng*	*bankinh <= o*	*false*
>	*Lớn hơn*	*bankinh > o*	*true*
>=	*Lớn hơn hoặc bằng*	*bankinh >= o*	*true*
==	*Bằng*	*bankinh == o*	*false*
!=	*Khác*	*bankinh != o*	*true*

3.5/ Các toán tử (operators) trong Java

3) Toán tử quan hệ (Relational Operators)

```
class Main {
    public static void main(String[] args) {
        // create variables
        int a = 7, b = 11;

        // value of a and b
        System.out.println("a is " + a + " and b is " + b);

        // == operator
        System.out.println(a == b);  // false

        // != operator
        System.out.println(a != b);  // true

        // > operator
        System.out.println(a > b);  // false

        // < operator
        System.out.println(a < b);  // true

        // >= operator
        System.out.println(a >= b);  // false

        // <= operator
        System.out.println(a <= b);  // true
    }
}
```

3.5/ Các toán tử (operators) trong Java

4) Toán tử logic (Logical Operators)

```
class Main {
    public static void main(String[] args) {
        // && operator
        System.out.println((5 > 3) && (8 > 5));  // true
        System.out.println((5 > 3) && (8 < 5));  // false

        // || operator
        System.out.println((5 < 3) || (8 > 5));  // true
        System.out.println((5 > 3) || (8 < 5));  // true
        System.out.println((5 < 3) || (8 < 5));  // false

        // ! operator
        System.out.println(!(5 == 3));  // true
        System.out.println(!(5 > 3));  // false
    }
}
```

Toán tử logic giúp kiểm tra một biểu thức là **đúng (true)** hay **sai (false)**.

3.5/ Các toán tử (operators) trong Java

5) Toán tử một ngôi (Unary Operators)

Ký hiệu	Phép toán	Mô tả	Ví dụ (giả sử i = 1)
++biến	Tăng trước	Tăng "biến" thêm 1 trước, sau đó thực hiện câu lệnh	int j = ++i //j = 2, i = 2
biến++	Tăng sau	Thực hiện câu lệnh trước, sau đó tăng "biến" thêm 1	int j = i++ //j = 1, i = 2
–biến	Giảm trước	Giảm "biến" bớt 1 trước, sau đó thực hiện câu lệnh	int j = –i //j = 0, i = 0
biến–	Giảm sau	Thực hiện câu lệnh trước, sau đó giảm "biến" bớt 1	int j = i– //j = 1, i = 0

3.5/ Các toán tử (operators) trong Java

5) Toán tử một ngôi (Unary Operators)

```
class Main {
    public static void main(String[] args) {
        // declare variables
        int a = 12, b = 12;
        int result1, result2;

        // original value
        System.out.println("Value of a: " + a);

        // increment operator
        result1 = ++a;
        System.out.println("After increment: " + result1);

        System.out.println("Value of b: " + b);

        // decrement operator
        result2 = --b;
        System.out.println("After decrement: " + result2);
    }
}
```

Kết quả

```
Value of a: 12
After increment: 13
Value of b: 12
After decrement: 11
```

3.6/ Chuyển kiểu (ép kiểu) - Casting

Đặt vấn đề

Khi khai báo biến để sử dụng trong chương trình của mình, đồng nghĩa với việc bạn "*Đề nghị được cấp phát một vùng nhớ dùng cho mục đích lưu trữ dữ liệu với không gian ... - tuỳ thuộc vào kiểu đã được khai báo*", và như vậy chương trình của bạn đôi khi lại phải đối mặt với một số vấn đề khi xử lý. Hãy xét đoạn chương trình sau

```
int  a = 7, b = 2;
float c = a / b;
System.out.printf("%d / %d = %2.1f", a,b,c);
```

Câu hỏi đặt ra là kết quả in ra trên màn hình là gì ?

3.6/ Chuyển kiểu (ép kiểu) - Casting

Đặt vấn đề

Khi khai báo biến để sử dụng trong chương trình của mình, đồng nghĩa với việc bạn "*Đề nghị được cấp phát một vùng nhớ dùng cho mục đích lưu trữ dữ liệu với không gian ... - tuỳ thuộc vào kiểu đã được khai báo*", và như vậy chương trình của bạn đôi khi lại phải đối mặt với một số vấn đề khi xử lý. Hãy xét đoạn chương trình sau

```
int  a = 7, b = 2;
float c = a / b;
System.out.printf("%d / %d = %2.1f", a,b,c);
```

Câu hỏi đặt ra là kết quả in ra trên màn hình là gì ?

3.6/ Chuyển kiểu (ép kiểu) - Casting

Đặt vấn đề

đoạn chương trình trên sẽ được viết lại như sau:

```
int  a = 7, b = 2;
float c = (float) a / b;
printf("%d / %d = %2.1f", a,b,c);
```

Casting – Ép kiểu

Lúc này, kết quả in ra trên màn hình sẽ đúng như ta muốn

7 / 2 = 3.5

3.6/ Chuyển kiểu (ép kiểu) - Casting

Cú pháp

(**kiểu_dữ_liệu**) <biến_cần_chuyển_kiểu>;
(**kiểu_dữ_liệu**) <dữ_liệu_cần_chuyển_kiểu>;

Ép kiểu ngầm định

```
float  v = 23;        //--- Tự động chuyển v = 23.0
int  f = 7.23;        //--- Tự động chuyển f = 7
```

3.6/ Chuyển kiểu (ép kiểu) - Casting

Cú pháp

(**kiểu_dữ_liệu**) <biến_cần_chuyển_kiểu>;
(**kiểu_dữ_liệu**) <dữ_liệu_cần_chuyển_kiểu>;

Ép kiểu tường minh

```
int  a=10, b=4;
float  c = (float) a/b; //--- Chuyển giá trị của a thành số thực
                        //--- sau đó thực hiện phép chia a cho b
                        //--- Lúc này, b tạm thời cũng được
                        //--- chuyển thành số thực (một cách tạm thời)
```

3.6/ Chuyển kiểu (ép kiểu) - Casting

Ví dụ 01:

```
class Main {
  public static void main(String[] args) {
    // create int type variable
    int num = 10;
    System.out.println("The integer value: " + num);

    // convert into double type
    double data = num;
    System.out.println("The double value: " + data);
  }
}
```

Output

```
The integer value: 10
The double value: 10.0
```

3.6/ Chuyển kiểu (ép kiểu) - Casting

Ví dụ 02

```
class Main {
  public static void main(String[] args) {
    // create double type variable
    double num = 10.99;
    System.out.println("The double value: " + num);

    // convert into int type
    int data = (int)num;
    System.out.println("The integer value: " + data);
  }
}
```

Output

```
The double value: 10.99
The integer value: 10
```

3.6/ Chuyển kiểu (ép kiểu) - Casting

Ví dụ 03

```
class Main {
  public static void main(String[] args) {
    // create int type variable
    int num = 10;
    System.out.println("The integer value is: " + num);

    // converts int to string type
    String data = String.valueOf(num);
    System.out.println("The string value is: " + data);
  }
}
```

Output

```
The integer value is: 10
The string value is: 10
```

3.6/ Chuyển kiểu (ép kiểu) - Casting

Ví dụ 04

```
class Main {
  public static void main(String[] args) {
    // create string type variable
    String data = "10";
    System.out.println("The string value is: " + data);

    // convert string variable to int
    int num = Integer.parseInt(data);
    System.out.println("The integer value is: " + num);
  }
}
```

Output

```
The string value is: 10
The integer value is: 10
```

Câu hỏi ôn tập

1) Kiểu dữ liệu là gì ? liệt kê các kiểu dữ liệu cơ bản(primitive data type) của Java ?
2) Identifier là gì ? Các quy ước liên quan đến Identifier
3) Đối tượng của lớp Scanner dùng để làm gì ? liệt kê các phương thức của Scanner Object
4) Có bao nhiêu loại phép toán, trình bày độ ưu tiên của phép toán
5) Hiểu thế nào về casting?