Lập trình JAVA cơ bản

Tuần 1

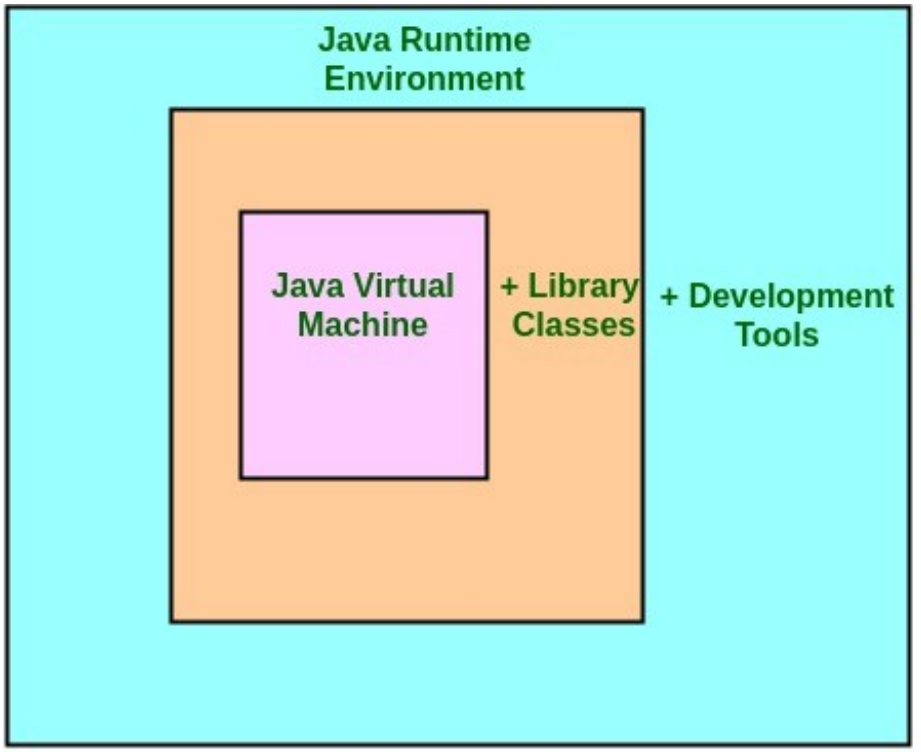
Giảng viên: Trần Đức Minh

Nội dung bài giảng

- Giới thiệu về Java
- Chương trình Java đầu tiên
- Giới thiệu về biến, các kiểu dữ liệu.
- Các cấu trúc điều khiển.
- Nhập dữ liệu.

Giới thiệu Java

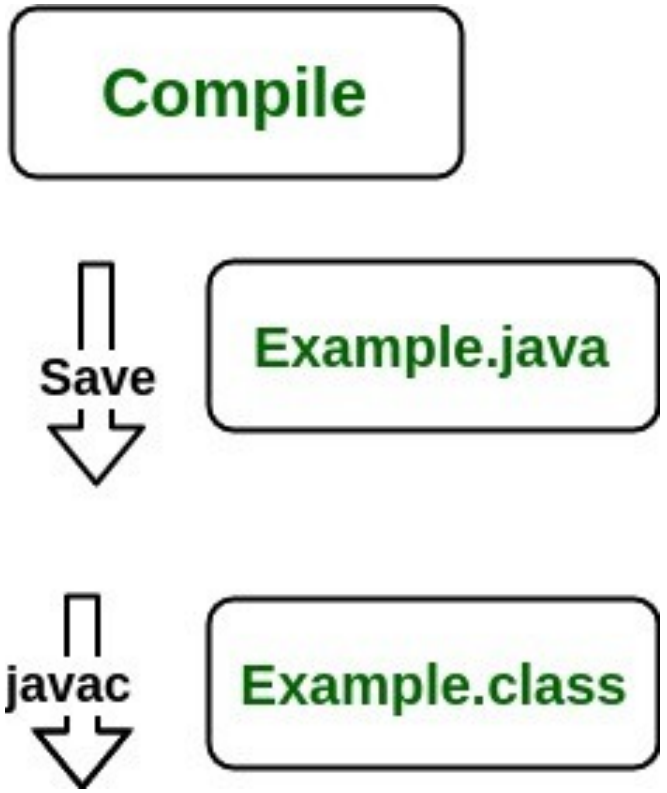
- JDK, JRE và JVM



JDK = JRE + Development Tool
JRE = JVM + Library Classes

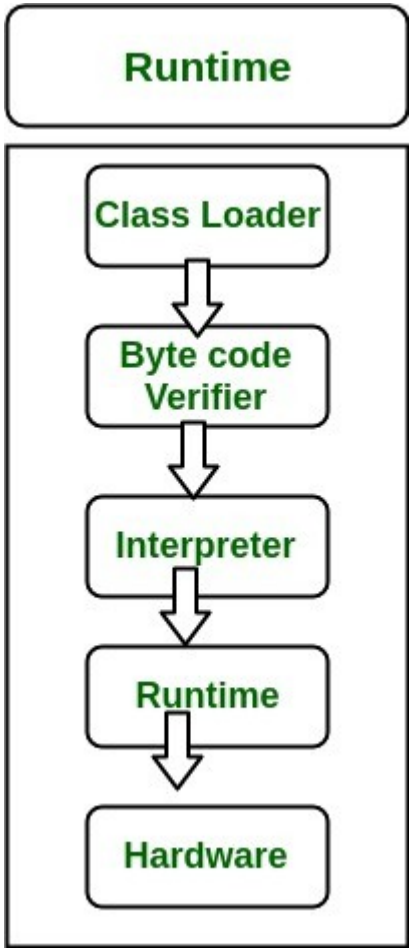
Giới thiệu Java

- Biên dịch chương trình trong Java



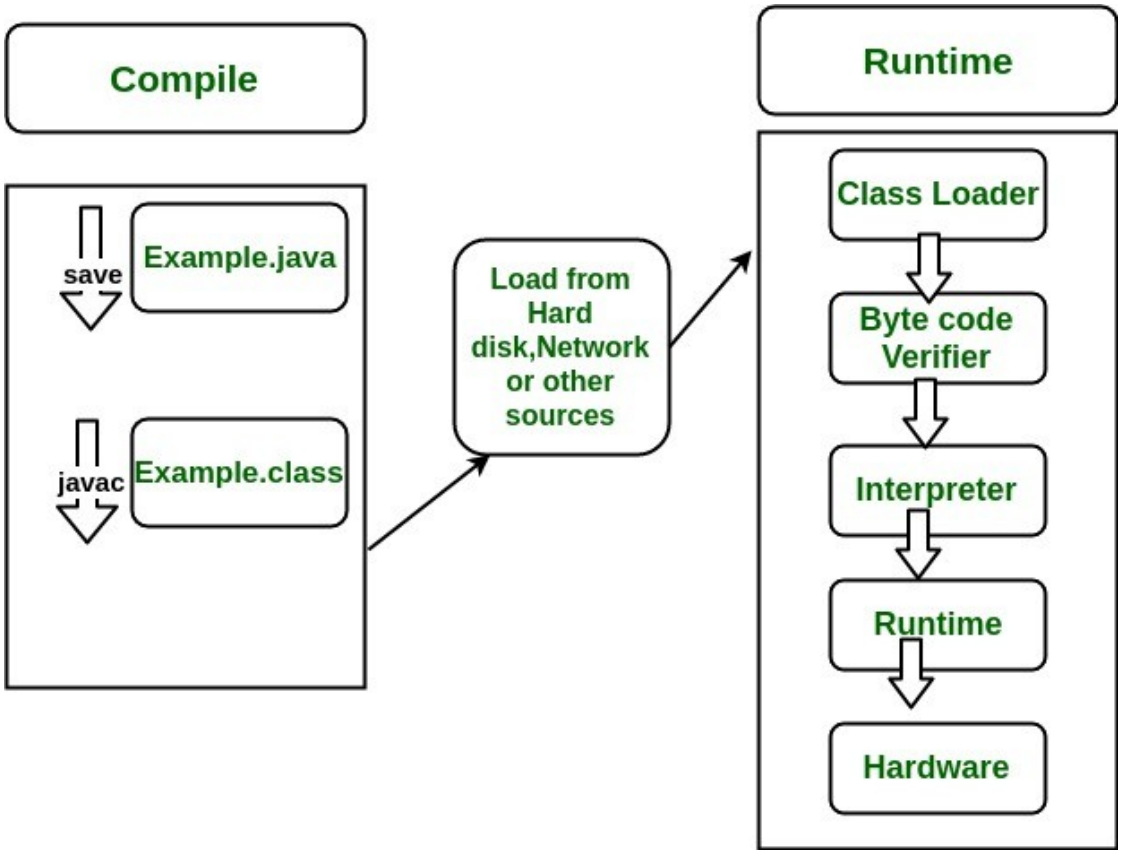
Giới thiệu Java

- Chạy mã bytecode trong Java



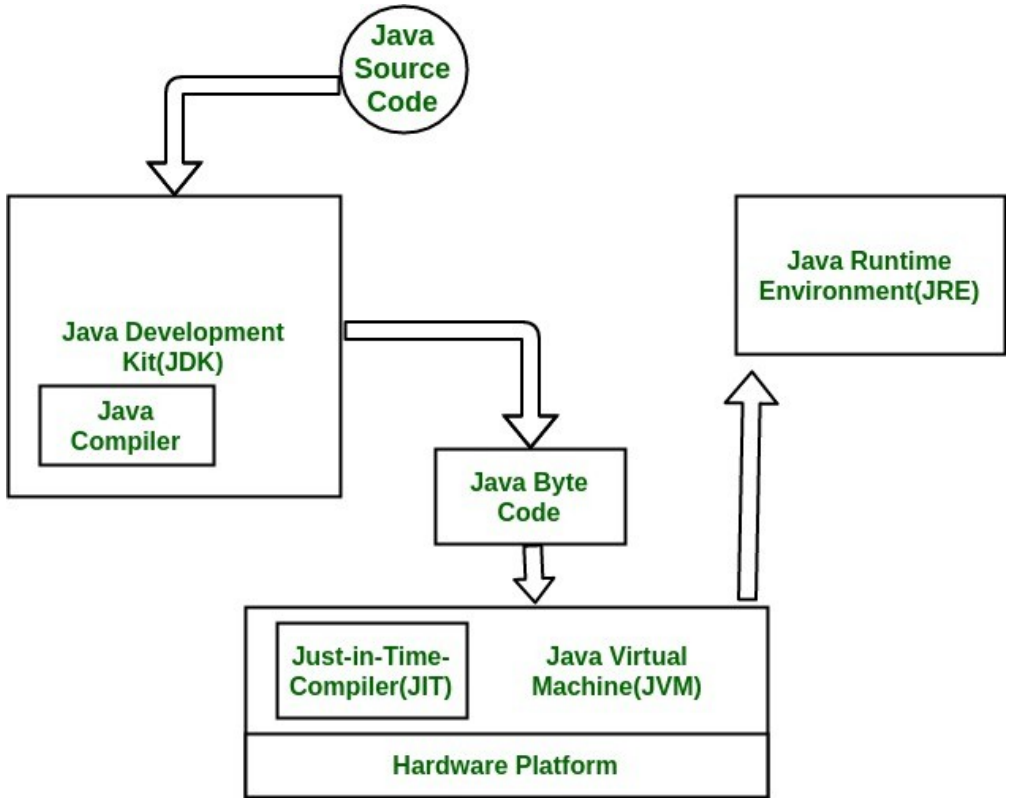
Giới thiệu Java

- Mô hình tổng quát



Giới thiệu tổng quát

- Mô hình tương tác giữa JDK và JRE
  - JIT: Biên dịch mã bytecode thành mã máy.



Các phiên bản

Các phiên bản Java đã phát hành:

- JDK 1.0 (23 tháng 01, 1996)
- JDK 1.1 (19 tháng 2 năm 1997)
  - JDK 1.1.5 *(Pumpkin)* 03 tháng 12 năm 1997
  - JDK 1.1.6 *(Abigail)* 24 tháng 4 năm 1998
  - JDK 1.1.7 *(Brutus)* 28 tháng 9 năm 1998
  - JDK 1.1.8 *(Chelsea)* 08 tháng 4 năm 1999
- J2SE 1.2 *(Playground)* 08 tháng 12 năm 1998
  - J2SE 1.2.1 *(không có)* 30 tháng 3 năm 1999
  - J2SE 1.2.2 *(Cricket)* 08 tháng 7 năm 1999
- J2SE 1.3 *(Kestrel)* 08 tháng 5 năm 2000
  - J2SE 1.3.1 *(Ladybird)* 17 tháng 5 năm 2001
- J2SE 1.4.0 *(Merlin)* 06 tháng 02, 2002
  - J2SE 1.4.1 *(Hopper)* 16 tháng 9 năm 2002
  - J2SE 1.4.2 *(Mantis)* 26 tháng 6 năm 2003
- J2SE 5 (1.5.0) *(Tiger)* 30 tháng 9 năm 2004
- Java SE 6 (còn gọi là *Mustang*), được công bố 11 tháng 12 năm 2006, thông tin chính tại http://java.sun.com/javase/6/. Các bản cập nhật 2 và 3 được đưa ra vào năm 2007, bản cập nhật 4 đưa ra tháng 1 năm 2008.
- JDK 6.18, 2010
- Java SE 7 (còn gọi là *Dolphin*), được bắt đầu từ tháng 8 năm 2006 và công bố ngày 28 tháng 7 năm 2011.
- JDK 8, 18 tháng 3 năm 2014
- JDK 9, 21 tháng 9 năm 2017
- JDK 10, 20 tháng 3 năm 2018

- Java SE 11, 18 tháng 9 năm 2018
- Java SE 12, 19 tháng 3 năm 2019

Chương trình đầu tiên

```java
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Lớp lập trình Java cơ bản !!!");
    }
}
```

- Các từ khóa trong Java là **case-sensitive**.
- **public** là từ khóa **Access Modifier** (xác định phạm vi truy cập).
- **class** là từ khóa dùng để định nghĩa một lớp.
- Tên lớp (**HelloWorld**) phải trùng với tên của file chứa lớp (**HelloWorld.java**).
- Phần thân của lớp chứa các thuộc tính và phương thức đều nằm giữa cặp dấu ngoặc nhọn **{ ... }**.

Chương trình đầu tiên

```java
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Lớp lập trình Java cơ bản !!!");
    }
}
```

- Điểm bắt đầu của chương trình là phương thức **main**.
- **public** là từ khóa **Access Modifier** của phương thức main.
- **static** là từ khóa cho phép truy cập phương thức main từ tên lớp mà không cần thiết tạo đối tượng lớp.
- **void** là từ khóa để xác định phương thức không có giá trị trả về.
- Tham số **String[] args** chứa thông tin được gửi từ bên ngoài vào chương trình.
- Phần thân của phương thức main được chứa bên trong **code block** (cặp dấu ngoặc nhọn **{ … }**).
- Lệnh **System.out.println(…)** được sử dụng để đưa thông tin ra màn hình.

Biến trong Java

- **Biến** (variable) là một cách để lưu trữ thông tin trong khi lập trình.
- Có nhiều **kiểu dữ liệu** (Data types) khác nhau được dùng để định nghĩa cho biến.
- Để **định nghĩa một biến**, ta cần xác định kiểu dữ liệu của biến, tên biến và có thể một giá trị khởi tạo nào đó cho biến (nếu cần).

```
int nonFirstValue;

int myFirstVariable = 10;

System.out.println(myFirstVariable);
```

Biểu thức

- Toán tử cơ bản: + - * / %
- Phân tích chương trình sau:

```java
public class HelloWorld {
    public static void main(String[] args) {
        int myFirstVariable = 5;
        int mySecondVariable = 5 * 2;
        int myThirdVariable = (10 + myFirstVariable) / (mySecondVariable - 1);
        System.out.println(myThirdVariable);
    }
}
```

Các kiểu dữ liệu cơ bản

- Java có 8 kiểu dữ liệu cơ bản
  - **byte**: số nguyên 1 byte, gồm cả số âm.
  - **short**: số nguyên 2 bytes, gồm cả số âm.
  - **int**: số nguyên 4 bytes, gồm cả số âm.
  - **long**: số nguyên 8 bytes, gồm cả số âm.
  - **float**: số thực 4 bytes.
    - Khoảng giá trị từ 1.4E-45 đến 3.4028235E+38
  - **double**: số thực 8 bytes.
    - Khoảng giá trị từ 4.9E-324 đến 1.7976931348623157E+308
  - **char**: chiếm 2 bytes trong bộ nhớ, lưu trữ ký tự Unicode và cũng có thể lưu trữ một số nguyên không âm với khoảng giá trị từ 0 đến 65535.
  - **boolean:** chiếm 1 byte trong bộ nhớ.

Số nguyên kiểu int

- Số nguyên kiểu **int** trong Java chiếm 4 **bytes** bên trong bộ nhớ.

```
public class HelloWorld {
    public static void main(String[] args) {
        int minIntValue = Integer.MIN_VALUE;
        int maxIntValue = Integer.MAX_VALUE;

        System.out.println(minIntValue);
        System.out.println(maxIntValue);
    }
}
```

- Java sử dụng **Wrapper class** (lớp bọc) để giúp ta thực hiện các thao tác trên các kiểu dữ liệu cơ bản. Như lớp bọc có tên Integer được dùng cho kiểu dữ liệu int.

Tràn số trên và tràn số dưới

- **Tràn số trên** (Overflow) nếu ta cố đưa vào một giá trị lớn hơn giá trị lớn nhất của kiểu dữ liệu.

```
int overflowNumber1 = Integer.MAX_VALUE + 1;
int overflowNumber2 = 2147483648;
System.out.println(overflowNumber1);
```

- **Tràn số dưới** (Underflow) nếu ta cố đưa vào một giá trị nhỏ hơn giá trị nhỏ nhất của kiểu dữ liệu.

```
int underflowNumber1 = Integer.MIN_VALUE – 1;
int myIntNumber = -2_147_483_648;
System.out.println(underflowNumber);
```

Số nguyên kiểu byte, short, long

```
byte myMaxByte = Byte.MAX_VALUE;
byte myMinByte = Byte.MIN_VALUE;

short myMaxShort = Short.MAX_VALUE;
short myMinShort = Short.MIN_VALUE;

long myLongNumber = 100L; // Tạo vùng nhớ 8 bytes
long myLongError = 2_147_483_647_234;
long myMaxLong = Long.MAX_VALUE;
long myMinLong = Long.MIN_VALUE;
```

Ép kiểu

- Phân tích đoạn code sau:

```
int myIntNumber = 52;
byte myByteValue = myIntNumber / 2;
short myShortValue = myIntNumber / 2;
long myLongValue = myIntNumber / 2;
```

- **Ép kiểu** (casting) được dùng để chuyển đổi kiểu dữ liệu của một số sang một kiểu dữ liệu khác.

```
byte myByteValue = (byte) (myIntNumber / 2);
```

- Java gán mặc định kiểu dữ liệu **int** cho một chữ số nguyên.

Số thực

- Phân tích đoạn code:

  float myFloatValue = 7.5;

  double myDoubleValue = 6.7;

- Khởi tạo giá trị

  float myFloatValue1 = 7.5**f**;

  float myFloatValue2 = (float) 7.5;

  double myDoubleValue = 6.7**d**;

- Java gán mặc định kiểu dữ liệu **double** cho một chữ số thực.

Số thực

- Phân tích kết quả đoạn code

```
int myIntValue = 7 / 3;
float myFloatValue = 7f / 3f;
double myDoubleValue = 7d / 3d;
```

- Phân tích từng câu lệnh bên dưới

```
float myFloatValue1 = 7 / 3;
float myFloatValue2 = 7f / 3;
float myFloatValue3 = 7f / 3.0;
double myDoubleValue = 7d / 3f;
```

Kiểu ký tự và kiểu logic

- Kiểu ký tự có thể lưu trữ các ký tự Unicode

  char myChar = **‘ộ’**;

  char myUnicodeChar = **‘\uCB04’**;

- **Unicode** là một chuẩn mã hóa ký tự quốc tế dành cho nhiều ngôn ngữ khác nhau trên thế giới. Do Unicode có thể mã hóa lên đến 65536 ký tự.
- Kiểu logic chỉ có 2 giá trị **true** hoặc **false**

  boolean myBooleanValue = **true**;

Kiểu dữ liệu String

- **String** là một kiểu dữ liệu trong Java, nhưng không phải kiểu dữ liệu cơ bản, nó là một lớp.

  String myString = “Chào các bạn sinh viên !!!”;

- String là một dãy các ký tự có độ dài tối đa lên đến Integer.MAX_VALUE (hơn 2 tỷ) ký tự.
- Ví dụ:

  String myString = “Chào các bạn sinh viên ”;

  String myClass = “lớp lập trình Java cơ bản.”;

  String myWholeString = myString + myClass;

  System.out.println(myWholeString);

Kiểu dữ liệu String

- Ví dụ:

```
String strBanQuyen = “\u00A9 Java Tutorials 2020”;
System.out.println(strBanQuyen);
```

- Phân tích đoạn code sau:

```
int mySoNguyen = 7 / 3;
float mySoThuc = 7f / 3f;
System.out.println(“Phần nguyên là: “ + mySoNguyen);
System.out.println(“Số thực là: “ + mySoThuc);
```

- Java sẽ tự động chuyển đổi kiểu số sang kiểu String trước khi thực thi phép cộng với kiểu dữ liệu String.

Chú thích

- Chú thích sẽ không tham gia vào quá trình biên dịch chương trình.
- Chú thích trên một dòng, Java sử dụng ký hiệu ***//***

```
int myPhanNguyen = 7 / 4; // Lấy phần nguyên
```

- Chú thích trên nhiều dòng, Java sử dụng ký hiệu ***/* ... */*** với các dòng được đặt giữa các ký hiệu này.

Các toán tử rút gọn

- ++ : Tăng giá trị biến số nguyên thêm 1 đơn vị.
  - ++a : Tăng giá trị a trước khi thực hiện biểu thức.
  - a++: Tăng giá trị a sau khi thực hiện biểu thức.
- -- : Giảm giá trị biến số nguyên đi 1 đơn vị.
  - --a : Giảm giá trị a trước khi thực hiện biểu thức.
  - a--: Giảm giá trị a sau khi thực hiện biểu thức.
- a += b: Tương đương với a = a + b
- a -= b: Tương đương với a = a - b
- a *= b: Tương đương với a = a * b
- a /= b: Tương đương với a = a / b
- a %= b: Tương đương với a = a % b

Các toán tử dùng với biểu thức logic

- Các toán tử so sánh: **>**, **<, >=**, **<=**, **==**, **!=**
- Các toán tử logic: **&&** (phép and), **||** (phép or)
- Toán tử phủ định: **!**
- Ví dụ:

```
boolean myValue1 = 6 > 7;
boolean myValue2 = (6 >= 7) && (9.0 <= 11.0);
boolean myValue3 = !myValue2;
```

Câu lệnh if

if (<biểu thức logic>) {

….

} else {

…

}

Toán tử Ternary

- Toán tử **ternary** (3 ngôi) trả về giá trị phụ thuộc vào biểu thức logic đứng trước dấu hỏi.

  <biểu thức logic> ? <giá trị 1> : <giá trị 2>

- Toán tử trả về **<giá trị 1>** khi **<biểu thức logic>** đúng, trả về **<giá trị 2>** khi **<biểu thức logic>** sai.
- Ví dụ:

```
int a = 5, b = 6;
int value = a > b ? a - b : a + b;
```

Phương thức

- Phương thức (method) trong Java bắt buộc phải được đặt bên trong một lớp nào đó.
- Ví dụ:

```
public class HelloWorld {
    public static int tinhTong(int soHang1, int soHang2){
        return soHang1 + soHang2;
    }
}
```

Phương thức

- Phân tích đoạn code sau:

```
public class HelloWorld {
    public static void main(String[] args) {
        int soNguyen1 = 11, soNguyen2 = 9;

        if (soNguyen1 > soNguyen2) {
            System.out.println("Hiệu của hai số là: " + tinhHieu(soNguyen1, soNguyen2));
        } else {
            System.out.println("Tổng của hai số là: " + tinhTong(soNguyen1, soNguyen2));
        }
    }

    public static int tinhTong(int soHang1, int soHang2) {
        return soHang1 + soHang2;
    }

    public static int tinhHieu(int soBiTru, int soTru) {
        return soBiTru - soTru;
    }
}
```

Quá tải phương thức

- **Quá tải phương thức** (Method overloading) là hành động xác định nhiều phương thức có cùng tên, nhưng các tham số khác nhau về số lượng hoặc kiểu dữ liệu.
- Không nên viết chương trình kiểu như sau:

```
public static int tongHaiSoNguyen(int soThu1, int soThu2) {
    return soThu1 + soThu2;
}

public static int tongBaSoNguyen(int soThu1, int soThu2, int soThu3) {
    return soThu1 + soThu2 + soThu3;
}

public static int tongBonSoNguyen(int soThu1, int soThu2, int soThu3, int soThu4) {
    return soThu1 + soThu2 + soThu3 + soThu4;
}
```

- Sử dụng quá tải phương thức giúp ta dễ nhớ tên phương thức và có tính mềm dẻo đối với các phương thức có tính chất tương tự nhưng chỉ khác kiểu dữ liệu.

Quá tải phương thức

- Phân tích đoạn code sau:

```
public class HelloWorld {
    public static void main(String[] args) {
        int soNguyen1 = 11, soNguyen2 = 9, soNguyen3 = 7;
        hienThiTong(soNguyen1, soNguyen2, soNguyen3);
    }

    public static void hienThiTong(int soThu1, int soThu2) {
        System.out.println("Tổng của 2 số nguyên: " + (soThu1 + soThu2));
    }

    public static void hienThiTong(int soThu1, int soThu2, int soThu3) {
        System.out.println("Tổng của 3 số nguyên: " + (soThu1 + soThu2 + soThu3));
    }

    public static void hienThiTong(double soThu1, double soThu2) {
        System.out.println("Tổng của 2 số thực: " + (soThu1 + soThu2));
    }
}
```

Câu lệnh switch

```
switch (<biểu thức>) {
    case <giá trị 1>:
        …
        break;
    case <giá trị 2>:
        …
        break;
    ….
    case <giá trị n>:
        …
        break;
    default:
        …
        break;
}
```

Câu lệnh switch

- Phân tích đoạn chương trình sau:

```
public class HelloWorld {
    public static void main(String[] args) {
        int thang = 11;
        System.out.println("Thang " + thang + " co " + getNgay(thang) + " ngay.");
    }

    public static int getNgay(int thang) {
        switch(thang) {
            case 1: case 3: case 5: case 7: case 8: case 10: case 12:
                return 31;

            case 4: case 6: case 9: case 11:
                return 30;

            case 2:
                return 29;
        }
        return 0;
    }
}
```

Vòng lặp for

for (<khởi tạo giá trị>; <biểu thức logic>; <biến điều khiển>) {

    <tăng/giảm biến điều khiển>

}

- Ví dụ:

```
public class HelloWorld {
    public static void main(String[] args) {
        int tong = 0;
        int n = 9;

        for(int i = 1; i <= 9; i++)
            tong += i;

        System.out.println("Tổng của " + n + " số tự nhiên đầu tiên là: " + tong);
    }
}
```

Vòng lặp while

while (<biểu thức logic>) {

…

}

- Ví dụ:

```
public class HelloWorld {
    public static void main(String[] args) {
        int tong = 0, n = 9, i = 1;

        while (i <= n) {
            tong += i;
            i++;
        }

        System.out.println("Tổng của " + n + " số tự nhiên đầu tiên là: " + tong);
    }
}
```

Vòng lặp do … while

do {

…

} **while** (<biểu thức logic>);

- Ví dụ:

```
public class HelloWorld {
    public static void main(String[] args) {
        int tong = 0, n = 1, i = 0;

        do {
            tong += i;
            i++;
        }
        while (i <= n);

        System.out.println("Tổng của " + n + " số tự nhiên đầu tiên là: " + tong);
    }
}
```

Sử dụng các vòng lặp

- **for** nên được sử dụng với các vòng lặp biết trước số lần lặp.
- **while** và **do … while** nên sử dụng với các vòng lặp không biết trước số lần lặp. Tuy nhiên:
  - **while** sử dụng với vòng lặp mà cần phải kiểm tra điều kiện lặp trước, sau đó mới thực hiện các câu lệnh trong code block.
  - **do … while** sử dụng khi cần thực hiện các câu lệnh trong code block 1 lần trước khi kiểm tra điều kiện.

Chuyển đổi dữ liệu từ chuỗi sang số

- Chúng ta sử dụng một số phương thức trong **Wrapper class** để biến đổi dữ liệu từ chuỗi sang số.
- Ví dụ:

```
String chuoiSoNguyen = “155”;
int soNguyen = Integer.parseInt(chuoiSoNguyen);

String chuoiSoThuc = “123.45”;
double soThuc = Double.parseDouble(chuoiSoThuc);
```

Đọc dữ liệu từ bàn phím

- Để đọc dữ liệu từ bàn phím ta sử dụng lớp Scanner trong gói thư viện java.util.*
- Ví dụ:

```java
import java.util.Scanner;

public class HelloWorld {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println("Nhập họ và tên: ");
        String strHoVaTen = scanner.nextLine();

        System.out.println("Nhập năm sinh: ");
        int iNamSinh = scanner.nextInt();

        System.out.println("Nhập điểm: ");
        double dDiem = scanner.nextDouble();

        scanner.close();
    }
}
```

Hết Tuần 1

Cảm ơn các bạn đã chú ý lắng nghe !!!