**BÀI TẬP LẬP TRÌNH JAVA CHƯƠNG 1**

**Câu 1: Mô tả các kiểu dữ liệu cơ sở trên Java**

* Kiểu logic: **boolean**.
* Các kiểu số nguyên (integer): **byte, short, char, int, long**.
* Các kiểu số thực (real number) (Cũng được gọi là kiểu dấu chấm động (floating point)): **float, double**.

**Câu 2: Mô tả các cấu trúc lệnh trên Java**

* Cấu trúc tuần tự
* Cấu trúc rẽ nhánh
* Cấu trúc lặp

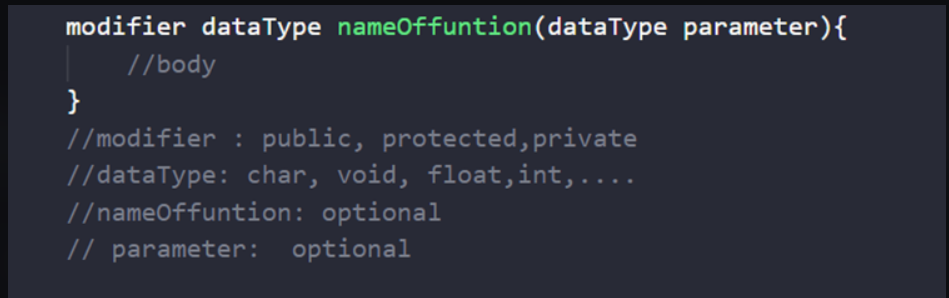
**Câu 3: Mô tả các môi trường lập trình Java**

* JVM (Java Virtual Machine): là môi trường dùng để chạy ứng dụng được viết bằng ngôn ngữ lập trình Java, giúp Java có thể chạy nhiều nền tảng (Platform) khác nhau.
* JRE (Java Runtime Environment): bao gồm các thư viện, JVM và những thành phần bổ sung để chạy những ứng dụng được viết bằng Java.
* JDK (Java Development Kit): là tập hợp JRE và những công cụ cần thiết để phát triển ứng dụng bằng Java.

**Câu 4: Liệt kê các đặc trưng của nên tảng lập trình Java**

* Đơn giản
* Mạnh mẽ
* Hướng đối tượng
* Bảo mật
* Phân tán
* Đa luồng
* Độc lập phần cứng và hệ điều hành

**Câu 5: Trình bày function trên Java**



* Hàm có trả về kết quả: dùng từ khoá return để trả về kết qủa mà nó đã tính toán được.
* Hàm không trả về kết quả: loại hàm này chúng ta chỉ cần gọi để sử dụng.

**Câu 6: So sánh các câu lệnh lặp trên Java**

* Vòng lặp For: Là câu lệnh lặp với số lần biết trước. Điều kiện xác định trước khi thực thi khối lệnh.

Cú pháp :

for (chỉ số bắt đầu; chỉ số kết thúc; giá trị tăng mỗi bước lặp)

{

Thực hiện câu lệnh;

}

* Vòng lặp While: Là câu lệnh lặp với số lần không biết trước, điều kiện xác định trước khi thực thi khối lệnh.

Cú pháp:

While (điều kiện lặp)

{

Thực hiện câu lệnh;

}

* Vòng lặp do-while: Là câu lệnh lặp với số lần ko biết trước, điều kiện xác định sau khi thực thi khối lệnh.

Cú pháp:

do

{

Thực hiện câu lệnh;

}

While (điều kiện lặp);

**Câu 7: Cho ví dụ về các hàm đệ quy và không đệ quy trên Java**

Ví dụ: hàm tính giai thừa của một số nguyên

* Không đệ quy:

private static int giaiThua(int n){

Int x=1;

For (int i=2;i<=n;i++)

{

x\*=i;

}

Return x;

}

* Đệ quy:

private static int giaiThua(int x){

if (i==0||i==1) return 1;

else

return i\*giaiThua(i-1);

}

**Câu 8: Tóm tắt kiểu dữ liệu String**

Thông thường, string là một chuỗi các ký tự. Nhưng, trong java string là một đối tượng biểu diễn một nối tiếp của các ký tự. Lớp java.lang.String được sử dụng để tạo đối tượng string.

Có 2 cách để tạo đối tượng String:

* Sử dụng string literal
* Sử dụng từ khóa new

Cách khởi tạo chuỗi trong Java:

* String s = new String();
* String s1 = "Xin chao";
* char chars[] = {’i’, ’B’, ’y’, ’t’, ’e’, ’C’, ’o’, ’d’, ’e’};
* String s2 = new String(chars);

**Câu 9: Cho ví dụ về các lớp bao trên Java**

//AutoBoxing

int i = 100;

Integer ib = i;

//AutoUnboxing

int iu = ib;

public static void main(String[] args) {

Integer integer = Integer.valueOf("121");

System.out.println(integer);

System.out.println("Integer to String: " + integer.toString()); }

**Câu 10: So sánh Java với các ngôn ngữ lập trình C, C++**

Điểm chung: đều dựa trên ngôn ngữ nguyên C, có nhiều nét tương đồng về từ khóa như: public, private, static và các kiểu dữ liệu thủy (ngoại trừ boolean), các toán tử. Đều hỗ trợ lập trình hướng đối tượng.

Điểm khác: Java loại bỏ biến con trỏ, thứ đã tồn tại ở C/C++, Java hỗ trợ các luồng (thread) tự nhiên.

**Câu 11: Lập trình tính tổng 1 + 2 + .... + n**

**import** java.util.Scanner;

**public** **class** baitapchuong1 {

**private** **static** Scanner *scanner* = **new** Scanner(System.***in***);

**public** **static** **int** *DEC\_10* = 10;

**public** **static** **void** main(String[] args) {

System.***out***.print("Nhập số nguyên dương n = ");

**int** n = *scanner*.nextInt();

System.***out***.printf("Tổng của các chữ số "

+ "của %d là: %d", n, *tong*(n));

}

**private** **static** **int** tong(**int** n) {

**var** tong=0;

**for** (**int** i = 0;i <= n;i++) {

tong+=i;

}

**return** tong;

}

}

**Câu 12: Lập trình tính n!**

**import** java.util.Scanner;

**public** **class** baitapchuong1 {

**private** **static** Scanner *scanner* = **new** Scanner(System.***in***);

**public** **static** **int** *DEC\_10* = 10;

**public** **static** **void** main(String[] args) {

System.***out***.print("Nhập số nguyên dương n = ");

**int** n = *scanner*.nextInt();

System.***out***.printf("Giai thua "

+ "của %d là: %d", n, *giaithua*(n));

}

**private** **static** **int** giaithua(**int** n) {

**var** giaithua=1;

**for** (**int** i = 1;i <= n;i++) {

giaithua\*=i;

}

**return** giaithua;

}

}

**Câu 13: Lập trình hàm kiểm tra số nguyên n có là số nguyên tố hay không**

**import** java.util.Scanner;

**public** **class** baitapchuong1 {

**public** **static** **void** main(String[] args) {

Scanner scanner = **new** Scanner(System.***in***);

System.***out***.print("Nhap so nguyen= ");

**int** n = scanner.nextInt();

**int** check = *songuyento*(n);

**if**( check == 1 ) System.***out***.println((**int**)n + " la so nguyen to" ) ;

**else** System.***out***.println(n+ " khong phai la so nguyen to" ) ;

}

**private** **static** **int** songuyento(**int** n){

**int** flag = 1;

**if** (n <2) **return** flag = 0;

**int** i = 2;

**while**(i <n){

**if**( n%i==0 ) {

flag = 0;

**break**;

}

i++;

}

**return** flag;

}

}

**Câu 14: Lập trình hàm nhập vào họ tên, trả về Họ Tên dạng proper**

**import** java.util.Scanner;

//import java.util.StringTokenizer;

**public** **class** baitapchuong1 {

**public** **static** **void** main(String[] args) {

Scanner Nhap = **new** Scanner(System.***in***);

String str = **new** String();

System.***out***.print("moi ban nhap 1 chuoi:");

str = Nhap.nextLine();

**char**[] charArray = str.toCharArray();

**boolean** foundSpace = **true**;

**for**(**int** i = 0; i < charArray.length; i++) {

**if**(Character.*isLetter*(charArray[i])) {

**if**(foundSpace) {

charArray[i] = Character.*toUpperCase*(charArray[i]);

foundSpace = **false**;

}

}

**else** {

foundSpace = **true**;

}

}

str = String.*valueOf*(charArray);

System.***out***.println("Chuỗi sau khi đổi: \n" + str);

}

}

**Câu 15: Lập trình hàm xuất ra mã nhị phân của số nguyên n**

**import** java.util.Scanner;

**public** **class** baitapchuong1 {

**private** **static** **void** xuatMaNhiPhan(**int** n) {

**if** (n<2) System.***out***.print(n);

**else** {

*xuatMaNhiPhan*(n/2);

System.***out***.print(n%2);

}

}

**static** baitapchuong1 *obj* =**new** baitapchuong1();

**private** **static** Scanner *scanner* = **new** Scanner(System.***in***);

**public** **static** **int** *DEC\_10* = 10;

**public** **static** **void** main(String[] args) {

System.***out***.print("Nhập số nguyên dương n = ");

**int** n = *scanner*.nextInt();

System.***out***.printf("gia tri nhi phan la ");

baitapchuong1.*xuatMaNhiPhan*(n);

System.***out***.println();

}

}

**Câu 16: Phân tích câu lệnh switch thành các lệnh if, lấy ví dụ minh họa**

Câu lệnh if và switch đều là câu lệnh rẽ nhánh nên ta có thể chuyển câu lệch switch thành if

Câu lệnh if có các nhánh rẽ(điều kiện) sẽ là từ khóa if/else còn câu lệnh switch sẽ là các case/default

Suy ra để chuyển switch thành if thì ta đưa các case thành các điều kiện trong câu lệnh if

Ví dụ:

switch (a) {

case 1:

//method

break;

case 2:

//method

break;

default:

//method

break;

}

Ta sẽ chuyển sang if như sau:

if(a = 1){

//method 1

}

if(a = 2){

//method 2

}

else{

//method 3

}

**Câu 17: Phân tích câu lệnh try catch, lấy ví dụ minh họa**

Câu lệnh try… catch là câu lệnh cho phép bạn thực hiện việc chạy code vẫn thành công khi các method trong từ try bị lỗi(các method sẽ bị bỏ đi) và thực thi tiếp tục các method trong từ khóa catch.

Ví dụ:

public class Main {

public static void main(String[ ] args) {

int[] myNumbers = {1, 2, 3};

System.out.println(myNumbers[10]); // error!

}

}

Đoạn code trên bị lỗi vì myNumbers[] chưa được khai báo nhưng nếu ta thêm câu lệnh try…catch vào đoạn code

public class Main {

public static void main(String[ ] args) {

try {

int[] myNumbers = {1, 2, 3};

System.out.println(myNumbers[10]);

} catch (Exception e) {

System.out.println("Something went wrong.");

}

}

}

Đoạn code sẽ thực hiện chạy thành công và sẽ có output là: Something went wrong.

**Câu 18: Biến đổi cấu trúc lệnh lặp for thành lệnh lặp điều kiện cho trước bằng ví dụ cụ thể**

**Vì while và for đêu có cấu trúc lặp nên ta có thể thay for thành while**

**Ví dụ: cấu trúc lệnh for:**

**Int i=0;**

**For(i= 0;i<10; i++){**

**//Method1;**

**//Method2;**

**….**

**}**

**Theo cấu trúc thì đầu tiên khai báo biến I =0 và ta sẽ so sánh i và n nếu i bé hơn 10 thì ta tăng I lên 1 đơn vị và thực hiện 2 method sau đó lặp lại đến khi i>=10 thì ta dừng lại và thoát for. Tương tự ta có thể làm thế với while như sau:**

**Int I = 0;**

**While(i<10){**

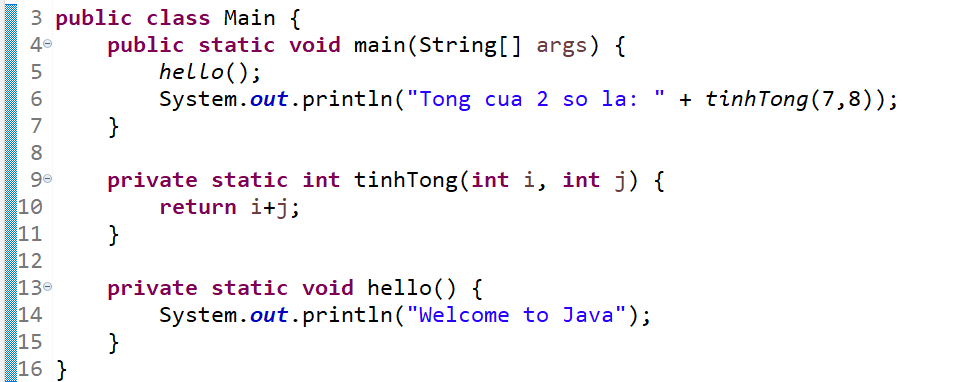
**I++;**

**//Method1;**

**//Method2;**

**}**

**Câu 19: Phân tích quá trình vận hành của chương trình Java bao gồm nhiều lời gọi hàm bằng ví dụ cụ thể**



|  |
| --- |
|  |
|  |  |

**Dòng lệnh số 5: là gọi lại hàm hello() tức là in ra màn hình chuỗi Welcome to Java**

**Dòng lệnh số 6: là câu lệnh in ra màn hình chuỗi Tong cua 2 so la: và gọi lại giá trị của hàm tinhTong**

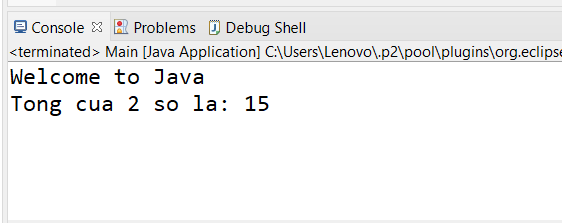
**Dòng lệnh số 9: là câu lệnh tạo hàm tinhTong() có kiểu số nguyên và có 2 tham số I,j là số nguyên**

**Dòng lệnh số 10: là câu lệnh trả về giá trị nguyên là tổng i+j**

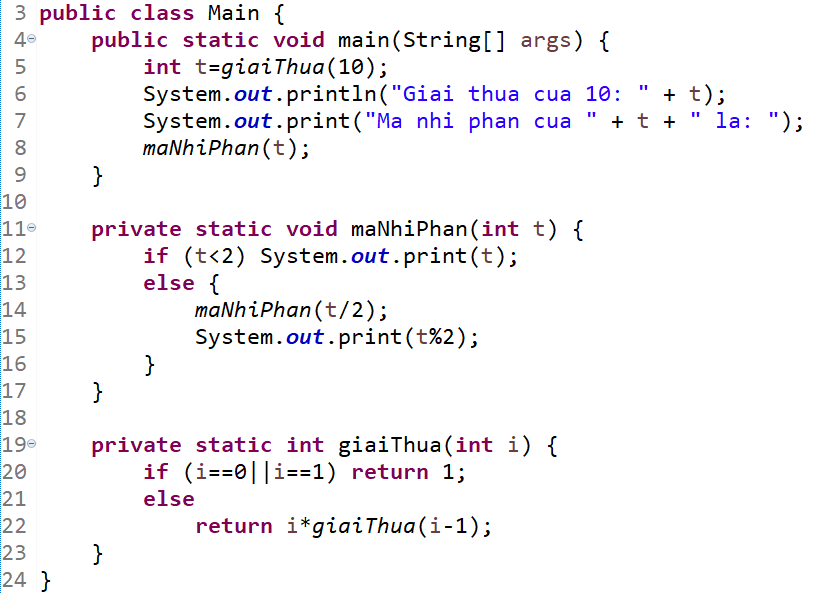
**Dòng lệnh số 13: là câu lệnh tạo hàm hello()**

**Dòng lệnh số 14: in ra màn hình chuỗi Welcome to Java khi hàm hello() được gọi**

**Và Output của chương trình sẽ là:**



**Câu 20: Phân tích quá trình vận hành của chương trình Java bao gồm nhiều lời gọi hàm đệ quy bằng ví dụ cụ thể**



Dòng 5: gán t bằng giai thừa của 10

Dòng 6,7: in ra màn hình chuỗi

Dòng 8: gọi hàm maNhiPhan

Dòng 11->24: tạo 2 hàm giaiThua và maNhiPhan

Và Output như sau:

