1. **TỔNG QUAN VỀ JAVA**

* Khái niệm :
* Java là một ngôn ngữ lập trình đơn giản được tạo ra và phát triển bởi James Gosling ở Sun Microsystems Inc và được công bố vào 5/1997.
* Java là một ngôn ngữ thực hiện tạo, dịch và kiểm tra mã nguồn một cách dễ dàng. Ngoài ra, nó còn giúp người lập trình có thể thực hiện tạo ra những module hoặc mã nguồn có khả năng tái sử dụng một cách dễ dàng.
* Đây là một ngôn ngữ lập trình được tạo ra dựa trên nguyên tắc thiết kế hướng đối tượng (OOP).
* Java là ngôn ngữ vừa thông dịch (interpret) và vừa biên dịch (compile).
* Java còn được viết đến với “WORA” tức là “write once run everywhere”. Điều này có nghĩa là chương trình sẽ được viết bằng java có thể được phát triển ở 1 hệ thống nhưng có thể chạy chương trình đó ở 1 hệ thống khác mà không cần phải điều chỉnh lại.
* Đặc điểm của Java:
* Đơn giản (Simple)
  + Dễ học và dễ thực hiện
  + Cú pháp dự trên C++
  + Cơ chế dọn rác tự động
* Hướng đối tượng (OOP)
  + Tất cả đều xoay quanh các đối tượng.
  + Được thiết kế và thực hiện hoàn toàn hướng đối tượng với các tính chất chính như :
    - Kế thừa (extends)
    - Đa hình (override , overloading)
    - Đóng gói (access modifier)
    - Trừu tượng (abstract class, interface)
* Bảo mật (Secure)
  + Cung cấp rất nhiều thư viện hỗ trợ cho security.
  + Chương trình Java chạy trong JVM
  + Java hỗ trợ cho việc xử lý lỗi và sẽ không cho phép người lập trình chạy chương trình mà chưa xử lý hết lỗi
  + Khác với các ngôn ngữ như C/C++ , java không hỗ trợ con trỏ, điều này giúp cho việc tránh sự truy cập vào các đối tượng trái phép trong java.
* Phân tán (Distributed)
  + Cung cấp khả năng tạo các chương trình phân tán bằng các công cụ như RMI hoặc EJB.
* Hiệu năng cao (High Performance)
  + Có tốc độ nhanh hơn so với các ngôn ngữ thông dịch truyền thống do bytecode của java khá giống với code lúc đầu.
  + Tuy nhiên là thông dịch nên nó tốc độ chậm hơn với các ngôn ngữ thuần biên dịch như C/C++.
* Đa tuyến (Multithreaded)
  + Java có khả năng tạo và thực thi nhiều chương trình cùng lúc nhờ vào sự điều khiển các thread.
* Thông dịch (Interpret)
  + Java vừa thông dịch và vừa biên dịch.
  + Java có khả năng thông dịch và run các file.class
* Khả chuyển (Portable)
  + Điều này được thể hiện bởi WORA ở bên trên
* Kiến trúc trung lập (Architecture)
  + Java là ngôn ngữ vừa biên dịch và vừa thông dịch nên khác với đa số các ngôn ngữ lập trình khác.
* Động (Dynamic)
  + Có khả năng tải các thư viện ở ngoài bên cạnh các thư viện đã có sẵn.
* Mạnh mẽ (Robust)
  + Java có thể được sử dụng để tạo chương trình trong nhất nhiều lĩnh vực (web app, mobile, game,…).
* Ứng dụng Java trong lập trình
  + Desktop application
  + Web application
  + Game
  + Robot
  + Hệ thống nhúng
  + Mobile
  + Sercutity
  + …
* Các loại Java đã ra đời
  + Java SE : phiên bản tiêu chuẩn, chứa toàn bộ thư viện chính và API và lập trình viên cần có
  + Java EE : chứa toàn bộ tính năng của Java SE, bên cạnh đó nó còn có được bổ sung thêm khả năng triển khai các phần mềm java có thể chịu lỗi, đa bậc và phân tán. Thường được sử dụng để triển khai các hệ thống đồ sộ, phân tán, yêu cầu các quyền truy cập vào CSDL, các web service.
  + Java ME : đây là phiên bản thường được sử dụng cho việc thực hiện tạo các chương trình cho mobile và hệ thống nhúng. Chứa các tính năng của phiên bản SE nhưng có thể các thư viện, API hỗ trợ cho mobile và nhúng.
* Các phiên bản Java đã được phát hành:

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Tên gọi | Công bố | Tính năng |
| JDK 1.0 | OAK | 23/01/1996 | Java Compiler |
| JDK 1.1 |  | 19/01/1997 | * JDBC * Inner Class * Java Beans * RMI * Reflection |
| J2SE 1.2 | Playground | 8/12/1998 | * Collections * Java String memory map for constraint * Just in time compiler * Jar signer for signing * Policy tool * JFC bao gồm : Swing 1.0 Drag, Drop,các thư viện Java 2D |
| J2SE 1.3 | Kestrel | 8/5/2000 | * Java Sound * Jar indexing * A huge lít ò enhancements in almost all in java area |
| J2SE 1.4 | Merlin | 6/2/2002 | * XML processing * Java print Service * Logging API * Java Web start * JDBC 3.0 API * Assertions * Preferences API * Chained Exception * IPv6 Support * Regular Exporessions * Image I/O API |
| J2SE 1.5 | Tiger | 30/09/2004 | * Generics * Enhanced for loop * Autoboxing/Unboxing * Typesafe Enums * Varargs * Static import * Metadata (Annotations) * Instrumentation |
| SE 6 | Mustang | 11/12/2006 | * Scripting Language Support * JDBC 4.0 API * Java Compiler API * Pluggable Annotations * Native PKI, Java GSS , Kerberos and LDAP support * Integrated Web Services * Lot more enhancements |
| ES 7 | Dolphin | 28/7/2011 | * Strings in switch Statement * Type inference for Genetic Instance creation * Multiple Exception Handling * Support for Dynamic Languages * Try with Resources * Jva nio Package * Binary Literals , underscore in literals * Diamod Syntax * Automatic null Handling |
| Java 8 |  | 18/3/2014 | * Lambda expression support in APIs * Stream API * Optionals * Functional interface and default methods * Nashorn * Annotation on Java types * Unsigned integer Arithmetic * Repeating annotations * New date and time api * Statically-lined JNI * Launch Java FX applications from jar files * Remove the permanent generation from GC |
| Java 9 |  | 9/2017 | * Java platform module system , interface private methods * Http2 client * JShell – REPL tool * Platform and JVM Logging * Process API Updates , Collection API updates * Improvements in Stream API, Multi-Release Jar files * Stack Walking * Java docs updates * Mescellaneous Other Features * @Deprecated tag changes |
| … |  |  |  |

1. **BIẾN , HẰNG VÀ TOÁN TỬ TRONG JAVA**

* Biến trong java
  + Trước hết cần tìm hiểu về datatype trong java
    - Trong java có 2 loại kiểu dữ liệu chính đó là : Nguyên thủy (primitive) và kiểu người dùng định dạng (user define)
      * Nguyên Thủy:
        + Số nguyên : byte (1 byte) , short (2 bytes), int (4 bytes) và long (8 bytes)
        + Số thực : float (4 bytes) và double (8 bytes)
        + Chuỗi : char (lưu 1 ký tự) , string (lưu chuỗi kí tự)
        + Logic : boolean (1 bit)
        + Enum : các giá trị đầu vào được định nghĩa sẵn
        + Time : Date , LocalDate , Calendar
  + Biến là vùng chứa giá trị dữ liệu trong suốt chương trình và biến sẽ có 1 kiểu dữ liệu và giá trị cần chứa.
    - Một biến là tên được đặt cho một địa chỉ bộ nhớ và là đơn vị cơ bản nhất để lưu trữ giá trị trong chương trình. Mọi hoạt động xảy ra với biến sẽ tác động đến vị trí bộ nhớ đó.
    - Biến trong java có thể được chỉnh sửa giá trị.
    - Biến trong java cần phải định nghĩa trước khi sử dụng.
    - Tên biến trong java có phân biệt hoa thường.
    - Định nghĩa 1 biến trong java :

<<datatype>> ten bien = gia tri;

Vd : int age = 20;

* Hằng trong Java:
  + Đây là 1 giá trị không đổi trong suốt chương trình của java
  + Cần phải khai báo giá trị từ đầu cho hằng
  + Khi khai báo hằng cần có từ kháo “final”

Vd : final string mu = “loser”;

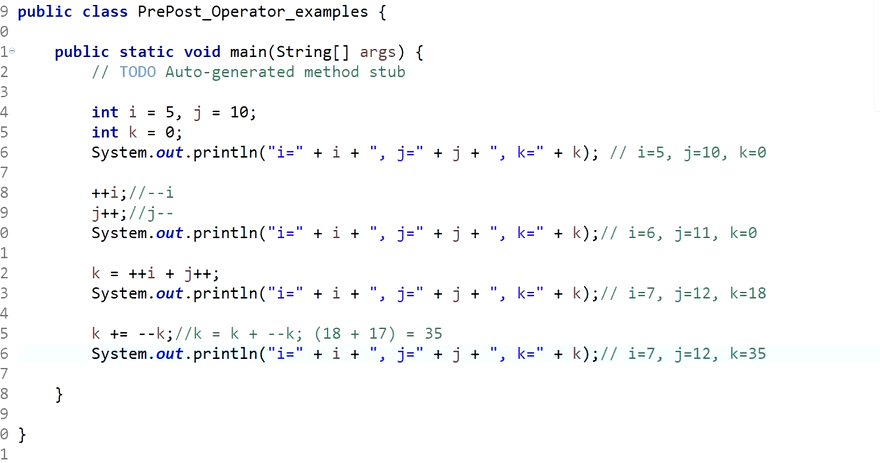
* Toán tử trong Java
  + Gồm 4 loại :
    - Toán tử số học
    - Toán tử gán
    - Toán tử so sánh
    - Toán tử logic

|  |  |
| --- | --- |
| Toán tử | Công dụng |
| + | Cộng |
| - | Trừ |
| \* | Nhân |
| / | Chia lấy phần nguyên |
| % | Chia lấy số dư |
| ++ | Tăng giá trị lên 1 đơn vị |
| -- | Giảm giá trị 1 đơn vị |
| == | So sánh bằng |
| != | So sánh khác |
| > | Lớn hơn |
| < | Bé lơn |
| >= | Lớn hơn hoặc bằng |
| <= | Bé hơn hoặc bằng |
| || | Hoặc |
| && | Và |
| ! | khác |
| = | Gán giá trị |
| += | Giá trị cũ cộng với giá trị nào đó  X+=5 ⬄ x = x + 5 |
| -= | Giá trị cũ trừ với giá trị nào đó  X-=5 ⬄ x = x - 5 |
| \*= | Giá trị cũ nhân với giá trị nào đó  X\*=5 ⬄ x = x \* 5 |
| /= | Giá trị cũ chia lấy phần nguyên với giá trị nào đó  X/=5 ⬄ x = x / 5 |
| %= | Giá trị cũ chia lấy số gư với giá trị nào đó  X%=5 ⬄ x = x % 5 |
| << | Dịch trái |
| >> | Dịch phải |

Tiền tố và hậu tố của toán tử tăng/giảm

Tiền tố : được thực hiện trước.

Hậu tố : được thực hiện sau



1. **CẤU TRÚC ĐIỀU KHIỂN TRONG JAVA**

**1.IF – ELSE**

Cú pháp đơn giản:

If(condition){

// todo

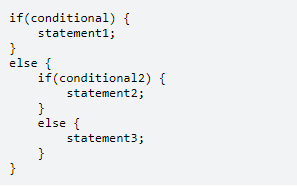
}else{

//todo

}

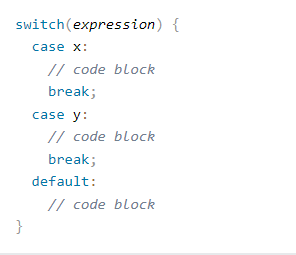
Trong đó : khi chương trình chạy đến cấu trúc if-else sẽ nhảy vào if, nếu condition trong if đúng thì sẽ thực thi đoạn code trong {} của if và và bỏ qua else. Còn nếu condition if sai thì sẽ thực thi {} của else.

Tương tự với if-else lồng nhau



**2. SWITCH-CASE**

Cú pháp



🡺 So sánh giá trị expression với các case trong switch. Nếu expression bằng 1 giá trị case nào đó thi sẽ thực thi code của case đó. Nếu không bằng case nào thì sẽ thực thi đoạn code ở default của switch.

1. **VÒNG LẶP**

* Khái niệm : thực hiện thực thi code lặp đi lặp lại nhiều lần.
* Các loại vòng lặp trong java:
  + While:

While(condition){

// code

};

Thực thi code lặp lại cho đến khi condition sai.

* + Do while:

do{

//code

}while(condition);

Tương tự như While , nhưng với do while thì đoạn code sẽ được thực thi ít nhất 1 lần.

* + For:
    - For i:

for(statement 1; statement 2; statement 3){

//code

}

Cần định nghĩa điều kiện dừng, bước nhảy của vòng lặp

* + - ForEach:

for(datatype tenbien : danh sach){

//code

}

1. **CHUỖI VÀ MẢNG**

* Mảng trong Java:
  + Lưu các giá trị có cùng kiểu dữ liệu
  + Không thể thay đổi kích thước khi đã khai báo
  + Có thể lưu trữ các thành phần trùng nhau
  + Có thể khai báo ở dạng 1 chiều hoặc đa chiều.
  + Vd khai báo :

kieudulieu a[] = {cac thanh phan mang};

kieudulieu [] a = {cac thanh phan mang};

kieudulieu [] a = new kieudulieu [<<kich thuoc mang>>];

kieudulieu [] a = new kieudulieu []{cac thanh phan mang};

…

* Truy xuất phần tử mảng:

<<ten mang>>[vi tri phan tu trong mang];

* Duyệt mảng:

Có thể sủ dụng fori hoặc foreach

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

System.out.println(a[i]);

}

Hoặc :

for(int i:a){

System.out.println(i);

}

* Sắp xếp mảng
  + Có thể sử dụng các thuật toán sắp xếp : select sort, bubble sort , insert sort, merge sort , binary sort, …
  + Có thể sử dụng Array.sort() để sắp xếp.
* Chuỗi trong java:
  + String là object lưu trữ 1 chuỗi các kí tự trong java.
  + Khai báo string :
    - Sử dụng string literal : String tenbien = “gia tri”;
    - Sử dụng từ khóa new : String tenbien = new String(“gia tri”);
  + Một số function thường sử dụng:
    - CharAt(index) : truy suất kí tự có vị trí index trong chuỗi
    - length() : độ dài của chuỗi
    - substring(int beginIdex , int endIdex) : cắt chuỗi từ vị trí beginIndex đến vị trí endIndex.
    - Equals(Object another) : so sánh giá trị chuỗi
    - toUpperCase() : chuyển chuỗi thành kiểu in hoa chuỗi
    - toLowerCase() : chuyển chuỗi thành kiểu in thường chuỗi
    - …
* String là bất biến trong java : có thể nói String là bất biến trong java tại vì nó cần như vậy để đảm bảo các yếu tố:
  + Đồng bộ
  + Bảo mật
  + Bộ nhớ đệm
  + Tải lớp

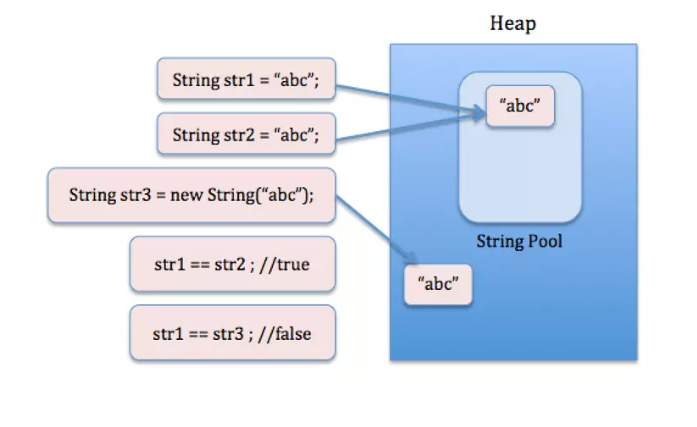
Vd 1:

1. **class** Testimmutablestring{
2. **public** **static** **void** main(String args[]){
3. String s="Sachin";
4. s.concat(" Tendulkar");
5. System.out.println(s); // Sachin
6. }
7. }

Vd2:

1. **class** Testimmutablestring1{
2. **public** **static** **void** main(String args[]){
3. String s="Sachin";
4. s=s.concat(" Tendulkar");
5. System.out.println(s);  // Sachin Tendulkar
6. }
7. }

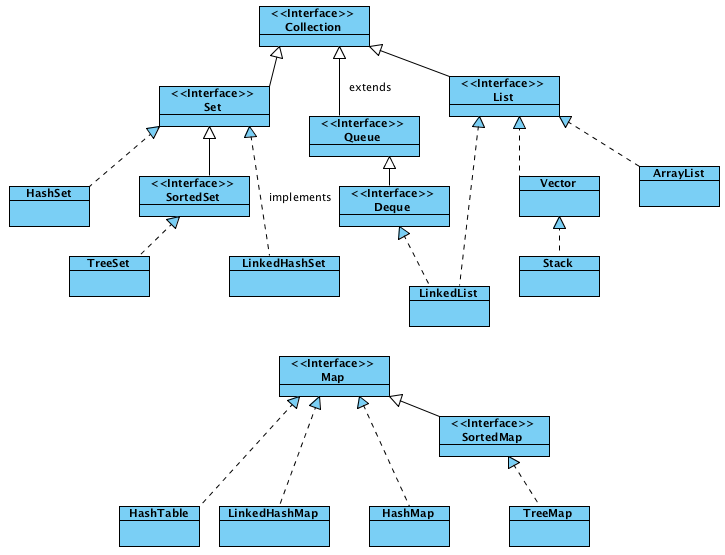
* String pool : là 1 vùng lưu trữ các giá trị được khởi tạo bởi cách khởi tạo String literal.



* Khi thực hiện khởi tạo 1 giá trị String bằng cách sử dụng String literal. Đầu tiên java sẽ kiểm tra trong spring pool xem có giá trị nào vằng với giá trị khởi tạo không. Nếu có thì sẽ thực hiện tham chiếu biến khởi tạo đó đến giá trị trong spring pool.
* Điều này giúp chúng ta có thể thấy được có thể có nhiều biến khi khởi tạo bằng Spring literal có thể tham chiếu đến cùng 1 địa chỉ trong bộ nhớ. Và khi khởi tạo bằng cách này, nếu so sánh giá trị chúng ta có thể sử dụng toán tử so sánh “==” để thực hiện so sánh.
* Khi thực hiện khởi tạo String bằng toán tử new thì java sẽ khởi tạo biến đó trong vùng nhớ Heap và như thế thì khi khởi tạo biến mới có cùng giá trị thì java vẫn sẽ tạo ô nhớ mới cho nó.
  + Khi khởi tạo bằng toán tử new thì lúc so sánh giá trị 2 biến String cần phải sử dụng function “Equal” để so sánh giá trị.

1. **COLLECTIONS**

* Khái niệm : Collection là một framework chuyên cung cấp các kĩ thuật và tính năng lưu trữ và thao tác (thêm, sửa, xóa,sắp xếp, tìm kiếm…) với danh sách các đối tượng.(chỉ thao tác với kiểu dữ liệu Object, không thao tác được với primitive).



* List : là một danh sách cấu trúc dữ liệu tuyến tính , trong đó các phần tử được sắp xếp theo một chiều nhất định.
  + Các phần tử trong list có thể trùng nhau.
  + Có thể thực hiện convert sang array.

|  |  |  |
| --- | --- | --- |
|  | **Mô tả** | **Tốc độ Insert, Remove** |
| ArrayList | * Giống mảng nhưng có thể thay đổi được kích thước mảng * Chỉ cần data và index * Phân mảnh nhiều (do các element phải sắp xếp liền nhau) | * Get phần tử rất nhanh (dựa vào index)   Do lưu trữ các phần tử trong list liền nhau trong bộ nhớ   * Insert và remove thì chậm do phải sắp sếp lại các phần tử |
| LinkedList | * Hoạt động trên cơ sở của cấu trúc dữ liệu danh sách liên kết đôi (double-linked list) * Sử dụng Queue(FIFO) * Cần data và địa chỉ của con trỏ next và prev * Không phân mảnh nhiều do các element không nhất thiết phải sắp xếp liền nhau | * Get phần tử chậm (không cho truy cập dựa vào index)   Do lưu trữ các phần tử trong list không liền nhau trong bộ nhớ   * Insert và remove thì nhanh |
| Vector | Tương tự với ArrayList |  |
| Stack | * Mỗi stack có thể biểu diễn bằng array, cấu trúc (Struct), con trỏ (Point) và linked list * Ứng dụng trong StackTrace (Exception)   Biá»u diá»n ngÄn xáº¿p (Stack) |  |

* Queue (Hàng đợi) : được thiết kế theo nguyên tắc FIFO (First In First Out). Phần tử được thêm vào trước thì sẽ ra trước.
* Set : lưu trữ các phần tử khác nhau.

|  |  |  |
| --- | --- | --- |
|  | **Mô tả** | **Tốc độ Insert, Remove** |
| HashSet | Các phần tử được lưu trữ dưới dạng bảng băm (HashTable) |  |
| LinkedHashSet | Giống HashSet nhưng được lưu theo LinkedList |  |
| SortedSet (interface) | Tổ chức Set theo thứ tự tăng dần hoặc giảm dần |  |
| TreeSet | Các phần tử đã được sắp sếp theo tree | Thêm, sửa, xóa và tìm kiếm 1 phần tử trong Set đều là O(log(n)) |
| EnumSet | Các phần tử trong set là các enum |  |

* Map:
  + Lưu trữ dữ liệu theo key và value.
  + Truy cập và lấy dữ liệu theo key
  + Khá giống Set nhưng map có thể lưu trữ các phần tử giống nhau, tuy nhiên key thì cần phải khác nhau.
  + Chuyển qua dạng Set thông qua function keySet().

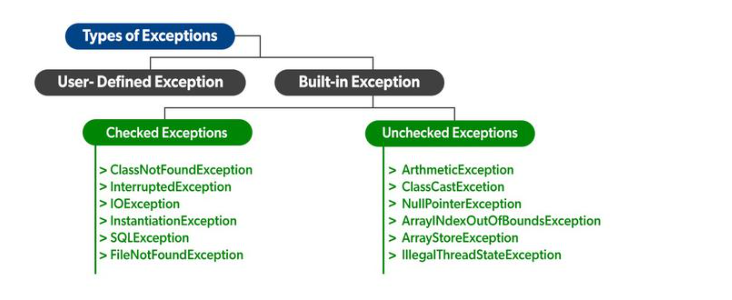
|  |  |  |
| --- | --- | --- |
|  | **Mô tả** | **Tốc độ Insert, Remove** |
| HashMap | * Các key được lưu trữ dưới dạng HashTable * Không biết trước được thứ tự của các phần tử |  |
| HashTable | * Giống HasMap nhưng Hastable cũ hơn HashMap |  |
| LinkedHashMap | Giống HashMap nhưng được lưu theo LinkedList |  |
| SortedMap (interface) | Tổ chức Map theo thứ tự tăng dần hoặc giảm dần | Thêm, sửa, xóa và tìm kiếm 1 phần tử trong Set đều là O(log(n)) |
| TreeMap | Các key đã được sắp sếp theo tree | Thêm, sửa, xóa và tìm kiếm 1 phần tử trong Map đều là O(log(n)) |
| EnumMap | Các key là enum |  |

1. **EXCEPTION**

* Phân biệt giữa Error và Exception

|  |  |
| --- | --- |
| Error | Exception |
| Các lỗi xảy ra trong chương trình mà ứng dụng không hề mong muốn, thường là các lỗi nghiêm trọng (lỗi phần cứng, out of memory, …) | Là các ngoại lệ xảy ra code. |
| Người lập trình không thể bắt và xử lí các lỗi này | Người lập trình có thể dễ dàng bắt và xử lí các lỗi này thông qua try catch hoặc throw lên tầng trên để xử lý. |

* Phân loại Exception : gồm 2 loại chính
  + Built-in Exception
    - Checked Exception : đây là loại exception được kiểm tra ở compiler time bởi compiler của Java.
    - Unchecked Exception : là ngoại lệ được kiêm tra ở Runtime.



* User-defined Exception

Đây là loại Custom Exception và là do chính người lập trình thực hiện tạo nên để bắt được các exception cụ thể nào đó.

Vd : có thể thực hiện tạo các Custom Exception cho các trường hợp như:

Tuổi chưa đủ 18, số điện thoại, gmail,… không đúng định dạng

* Stack trance:
  + Ngăn xếp stack trance mô phỏng cơ chế thực thi method hoặc xử lí exception.
  + Các method được thực thi theo cơ chế stack (LIFO)
  + Nếu 1 line gặp lỗi thì sẽ tìm kiếm bất kì xử lý lỗi nào trong method (VD: try…catch).
    - Nếu không có bất kì xử lý lỗi nào (không có try catch) thì nó sẽ throw lỗi lên parent method
    - Nếu parent cũng không có xử lý lỗi nào thì exception sẽ được đẩy lên thông qua stack và thông báo lỗi được in ra cửa sổ Output
  + Nếu không gặp lỗi thì method sẽ bị remove đi khỏi stack trace.
* Sử dụng try catch finally xử lý exception.

Cú pháp :

try {

// block of code to monitor for errors

// the code you think can raise an exception

} catch (ExceptionType1 exOb) {

// exception handler for ExceptionType1

} catch (ExceptionType2 exOb) {

// exception handler for ExceptionType2

}

// optional

finally { // block of code to be executed after try block ends

}

Sẽ có 2 trường hợp xảy ra :

1. Sẽ có không có exception xảy ra trong khối try

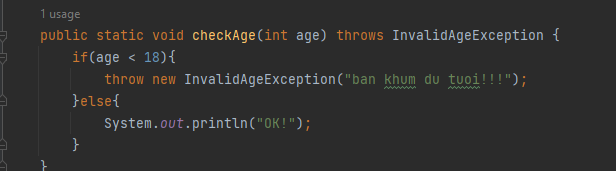
Khi đó khối lệnh bên trong catch sẽ không được thực thi.

2. Có exception xảy ra tronh khối try

Lúc đó khối catch sẽ được gọi để thực thi xử lý exception đó.

Khi try có nhiều catch bọc thì sẽ có thứ tự ưu tiên như sau:

* Chương trình sẽ duyệt tuần tự từng catch bao bọc try.
* Khi có catch tương ứng với exception xảy ra trong try thì catch đó sẽ được gọi để thực thi và bỏ qua các catch ở dưới nó.
* Khi catch ở phía trên là dạng subexception của một exception bên dưới.Thì khi exception xảy ra trong try, subexception sẽ không được gọi vì exception cha đã bao bọc nó.

 Và dù có exception hay không có exception xảy ra trong khối try{} thì đoạn finally vẫn sẽ được thực thi.

