**BỘ CÂU HỎI PHỎNG VẤN MODULE 2**

**ADVANCED PROGRAMMING WITH JAVA**

***Note 1: Overloading***

*Là khi truyền tham số vào cùng 1 hàm, có thể khác kiểu dữ liệu, cùng tham số hoặc khác tham số, nhưng chỉ chuyển được từ kiểu dữ liệu thấp lên cao, còn cao về thấp thì không được, sinh ra lỗi*

***Note 2: Tính chất "viết một lần chạy nhiều nơi" của java là gì?***

*Java code được biên dịch thành một byte code, là ngôn ngữ trung gian giữa mã nguồn và mã máy. Byte code này không phải là nền tảng cụ thể và do đó có thể được thông dịch cho bất kỳ nền tảng nào.*

*Java gồm JDK, JRE, JVM. Chương trình chạy, sẽ được biên dịch thành các byte code được sử dụng trên nhiều nền tảng khác nhau. Không cần biên dịch lại. Các byte code sẽ được thực thi trong JRE. JVM cung cấp môi trường không gian ảo để thông tin tới các nền tảng (mã máy).*

**Note 3: *Phân biệt truyền tham trị và truyền tham chiếu?***

|  |  |
| --- | --- |
| **Tham trị *(passed by value)*** | **Tham chiếu *(passed by reference)*** |
| *– Kiểu này dành cho các biến, các tham số khai báo kiểu dữ liệu cơ bản nguyên thủy gồm: byte, short, int, long, float, double, char.*  - Là việc ta truyền giá trị vào hàm số (giá trị bên ngoài k thay đổi)  - Tạo ra 1 vùng nhớ mới mượn giá trị. Khi thay đổi địa chỉ này, thì địa chỉ kia k bị thay đổi | *- Hàm, mảng, lớp*  - Là truyền biến vào (*Việc thay đổi giá trị của biến tham chiếu bên trong phương thức làm thay đổi giá trị gốc của nó.*  )  - Là sẽ copy địa chỉ nhớ ra thành 1 địa chỉ mới nhưng cả 2 địa chỉ cùng trỏ vô 1 vùng nhớ duy nhất. Nếu thay đổi dữ liệu 1 cái thì cái còn lại cũng sẽ bị thay đổi |

***Khác nhau Biến tham trị và Biến tham chiếu?***

Biến tham trị chứa giá trị của nó trong vùng nhớ được cấp

Biến tham chiếu chứa tham chiếu đến đối tượng trong vùng nhớ được cấp

*Tham trị (passed by value) trong Java*

*– Kiểu này dành cho các biến, các tham số khai báo kiểu dữ liệu cơ bản nguyên thủy gồm: byte, short, int, long, float, double, char.*

*Tham chiếu (passed by reference) trong Java*

*– Khi chúng ta gọi một phương thức và truyền một tham chiếu cho phương thức đó được gọi là truyền tham chiếu. Việc thay đổi giá trị của biến tham chiếu bên trong phương thức làm thay đổi giá trị gốc của nó.*

*– Trong Java, n có thể dùng biến tham chiếu vì nó lưu địa chỉ đối tượng được lưu trong bộ nhớ Heap nhưng biến tham chiếu được lưu trong bộ nhớ Stack*

Sự khác nhau giữa lớp và đối tượng trong java

Sự khác nhau giữa lớp và đối tượng trong java được thống kê trong bảng sau:

| **No** | **Đối tượng** | **Lớp** |
| --- | --- | --- |
| 1. | Đối tượng là thể hiện của 1 lớp. | Lớp là một khuân mẫu hay thiết kế để tạo ra các đối tượng. |
| 2. | Đối tượng là 1 thực thể trong thế giới thực như Bút chì, Xe đạp, ... | Lớp là một nhóm các đối tượng tương tự nhau. |
| 3. | Đối tượng là 1 thực thể vật lý | Lớp là 1 thực thể logic |
| 4. | Đối tượng được tạo ra chủ yếu từ từ khóa new. Ví dụ: Student s1=new Student(); | Lớp được khai báo bằng việc sử dụng từ khóa class. Ví dụ: class Student{} |
| 5. | Đối tượng có thể được tạo nhiều lần. | Lớp được khai báo 1 lần duy nhất. |
| 6. | Đối tượng được cấp bộ nhớ khi nó được tạo ra. | Lớp không được cấp bộ nhớ khi nó được tạo ra. |
| 7. | Có rất nhiều cách để tạo ra đối tượng trong java như từ khóa new, phương thức newInstance(), phương thức clone(), phương thức factory và deserialization. | Chỉ có một cách để định nghĩa lớp trong java sử dụng từ khoá class. |

***1 Sự khác nhau giữa JDK, JRE và JVM?***

Sự khác biệt giữa JDK, JRE và JVM

Dưới đây là những điểm khác biệt chính giữa JDK so với JRE và JVM:

|  |  |  |
| --- | --- | --- |
| **JDK** | **JRE** | **JVM** |
| Java Development Kit  (Bộ phát triển Java) | Java Runtime Environment.  (Môi trường thời gian chạy Java) | Java Virtual Machine  (Máy ảo Java) |
| JDK là một bộ công cụ phát triển phần mềm để phát triển các ứng dụng bằng Java. | Nó là một gói phần mềm cung cấp **array líst, interface, clacss, kiểu dữ liệu nguyên thủy**,các thư viện lớp Java với các thành phần cần thiết để chạy mã Java. | JVM thực thi mã byte Java và cung cấp môi trường để thực thi nó. |
| JDK phụ thuộc vào nền tảng. | JRE cũng phụ thuộc vào nền tảng. | JVM không phụ thuộc nhiều vào nền tảng. |
| Nó chứa các công cụ để phát triển, gỡ lỗi và giám sát mã java. | Nó chứa các thư viện lớp và các tệp hỗ trợ khác mà JVM yêu cầu để thực thi chương trình. | Các công cụ phát triển phần mềm không được bao gồm trong JVM. |
| Nó là tập cha của JRE | Nó là tập con của JDK. Có Scaner, Long. | JVM là một tập con của JRE. |
| JDK cho phép các nhà phát triển tạo các chương trình Java có thể được thực thi và chạy bởi JRE và JVM. | JRE là một phần của Java tạo ra JVM. | Nó là thành phần nền tảng Java thực thi mã nguồn. |
| JDK đi kèm với trình cài đặt. | JRE chỉ chứa môi trường để thực thi mã nguồn. | JVM được đóng gói trong cả phần mềm JDK và JRE. |

***2 Sự khác nhau giữa bộ nhớ heap và stack trong java?***

Bộ nhớ Heap – Heap Memory

* Heap memory là bộ nhớ được sử dụng bởi Java Runtime để cấp phát bộ nhớ cho các đối tượng (object) và String.
* Bất kỳ khi nào có một đối tượng được tạo, nó sẽ được tạo lưu ở bộ nhớ Heap.
* Bộ dọn rác (Garbage Collection) chạy trên heap memory để giải phóng bộ nhớ được sử dụng bởi các đối tượng ko có bất kỳ tham chiếu nào.

Bộ nhớ Stack – Stack Memory

* Stack memory được sử dụng cho quá trình thực thi của mỗi thread.
* Stack memory bao gồm các giá trị cụ thể của method: các biến local và các tham chiếu tới các đối tượng chứa ở trong heap memory được tham chiếu bởi method.
* Stack memory được tham chiếu theo thứ tự LIFO (Last In First Out – vào cuối cùng thì ra đầu tiên). Tức là lưu trữ kiểu ngăn xếp (stack). Khi có một method được thực thi, một block được tạo ra trong stack memory để chứa các biến nguyên thủy local và các tham chiếu tới các object. Khi methode kết thúc, block đó sẽ không còn được sử dụng và được phục vụ cho method tiếp theo.
* Stack memory có kích thước rất nhỏ so với Heap memory.

***4 Platform là gì?***

*(Platform là thuật ngữ lập trình chỉ nền tảng kết nối tạo ra môi trường mà trong đó các phần mềm được thực thi. Các mô hình Platform thực tiễn có thể là phần cứng hay hệ điều hành, trình duyệt web thậm chí cả các giao diện lập trình ứng dụng có liên quan…*

*Platform là nền tảng bắt buộc phải có của mọi quy trình phát triển phần mềm. Mỗi platform sẽ có những chức năng hỗ trợ khác nhau. Kèm với đó là các hạn chế còn tồn tại. Ngoài ra, nó cũng hỗ trợ các quá trình phát triển bằng cách cung cấp một hoặc một vài chức năng cơ bản cấp thấp. )*

Platform (Nền tảng) là một nhóm các công nghệ được sử dụng làm cơ sở cho các ứng dụng, quy trình hoặc công nghệ khác phát triển tiếp.

***5 Sự khác nhau giữa Java platform và các platform khác?***

***Cách 1:***

*Java Platform (nền tảng Java) là một tập hợp các chương trình giúp phát triển và chạy các chương trình được viết bằng ngôn ngữ lập trình Java.*

*Java Platform bao gồm một công cụ thực thi (execution engine), trình biên dịch (compiler) và một bộ thư viện Java.*

*JAVA là ngôn ngữ độc lập với nền tảng. Nó không dành riêng cho bất kỳ bộ xử lý hoặc hệ điều hành.*

***Cách 2:***

*Java platform khác với hầu hết các nền tảng khác theo nghĩa nó là một nền tảng dựa trên phần mềm chạy trên các nền tảng phần cứng khác. Nó có hai thành phần:*

1. *Môi trường thực thi (Runtime)*
2. *API(Application Programming Interface) - Giao diện lập trình ứng dụng*

***6 Tính chất "viết một lần chạy nhiều nơi" của java là gì?***

*Java code được biên dịch thành một byte code, là ngôn ngữ trung gian giữa mã nguồn và mã máy. Byte code này không phải là nền tảng cụ thể và do đó có thể được thông dịch cho bất kỳ nền tảng nào.*

*Java gồm JDK, JRE, JVM. Chương trình chạy, sẽ được biên dịch thành các byte code được sử dụng trên nhiều nền tảng khác nhau. Các byte code sẽ được thực thi trong JRE. JVM cung cấp môi trường không gian ảo để thông tin tới các nền tảng.*

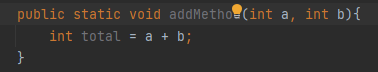
***7 Classloader trong java là gì?***

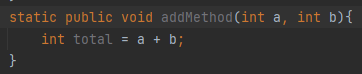
*Classloader là một hệ thống con của JVM được sử dụng để tải các lớp và các interface (giao diện). Có rất nhiều loại Classloader ví dụ: Bootstrap classloader, Extension classloader, System classloader, Plugin classloader…*

***10 Nếu không cung cấp bất kỳ đối số nào trên command line, thì mảng String của hàm main là empty hay null?*** *Mảng String là empty, không phải null.*

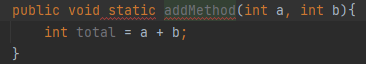
***11 Chuyện gì xảy ra nếu khai báo static public void thay vì public static void?***

*Chương trình được biên dịch và run đúng.*

*=> vd static void*

*=> đổi chỗ static và public*

*NHƯNG KHI ĐỔI CHỔ CHO VOID THÌ PHÁT SINH LỖI NHƯ HÌNH DƯỚI*

**

***12 Giá trị mặc định của các biến local là gì?***

*Các biến local không được khởi tạo với bất kỳ giá trị mặc định nào, bất kể là nguyên thủy hay tham chiếu đối tượng.*

***13 Khác nhau Biến tham trị và Biến tham chiếu?***

*Tham trị (passed by value) trong Java*

*– Kiểu này dành cho các biến, các tham số khai báo kiểu dữ liệu cơ bản nguyên thủy gồm: byte, short, int, long, float, double, char.*

*Tham chiếu (passed by reference) trong Java*

*– Khi chúng ta gọi một phương thức và truyền một tham chiếu cho phương thức đó được gọi là truyền tham chiếu. Việc thay đổi giá trị của biến tham chiếu bên trong phương thức làm thay đổi giá trị gốc của nó.*

*– Trong Java, n có thể dùng biến tham chiếu vì nó lưu địa chỉ đối tượng được lưu trong bộ nhớ Heap nhưng biến tham chiếu được lưu trong bộ nhớ Stack*

***14 Bạn hiểu gì về Đối tượng?***

* Đối tượng là một phần của Lớp được truy xuất từ lớp,
* có định danh, thuộc tính và hành vi

- *(Một thực thể có trạng thái và hành vi. Ví dụ như xe đạp, bàn, ghế, … Nó có thể mang tính vật lý hoặc logic.)*

***15 Định nghĩa Lớp (class)?***

* Lớp là khuôn mẫu để sinh ra đối tượng
* *Một tập hợp các đối tượng. Nó là một thực thể logic.*

***16 Kiểu biến nào một lớp có thể chứa?*** Kiểu dữ liệu nguyên thủy, kiểu tham chiếu, biến statics, instance, object

***17 Biến local hay biến cục bộ là gì?*** Nằm trong phương thức, khai báo trong 1 hàm, sau khi khai báo sẽ bị hủy đi. Toàn cục: cài dặt ở trong thuộc tích trong lớp.

***18 Biến instance là gì?*** *Mô tả lại 1 đối tượng trong lớp.*

Là giá trị khởi tạo của biến tham chiếu đến đối tượng. Giá trị khởi tạo ban đầu là null

***19 Sự khác biệt giữa ngôn ngữ lập trình hướng đối tượng và ngôn ngữ lập trình dựa trên đối tượng là gì?***

*Lập trình hướng thủ tục (POP) và Lập trình hướng đối tượng (OOP) đều là phương pháp lập trình, sử dụng ngôn ngữ cấp cao.*

|  |  |  |
| --- | --- | --- |
| *CƠ SỞ ĐỂ SO SÁNH* | *POP* | *OOP* |
| *Căn bản* | *Thủ tục / Cấu trúc định hướng.* | *Hướng đối tượng.* |
| *Hướng tiếp cận* | *Từ trên xuống* | *Từ dưới lên* |
| *Nền tảng* | *Trọng tâm chính là "làm thế nào để hoàn thành nhiệm vụ" tức là về thủ tục hoặc cấu trúc của một chương trình.* | *Trọng tâm chính là "bảo mật dữ liệu". Do đó, chỉ các đối tượng được phép truy cập các thực thể của một lớp.* |
| *Division* | *Chương trình lớn được chia thành các đơn vị gọi là chức năng.* | *Toàn bộ chương trình được chia thành các đối tượng.* |
| *Chế độ truy cập thực thể* | *Không có specifier truy cập.* | *Có tình xác định truy cập là "public", "private", "protected".* |
| *Overloading or Polymorphism* | *Neither it overload functions nor operators.* | *It overloads functions, constructors, and operators.* |
| *Thừa kế* | *Không hỗ trợ thừa kế* | *Thừa kế được hỗ trợ ở ba trạng thái "public", "private", "protected"* |
| *Bảo mật* | *Không có cách ẩn dữ liệu thích hợp, vì vậy dữ liệu không an toàn* | *Dữ liệu được ẩn trong ba chế độ"public", "private", "protected" do đó bảo mật dữ liệu tăng lên.* |
| *Chia sẻ dữ liệu* | *Dữ liệu toàn hệ thống được chia sẻ giữa các chức năng trong chương trình.* | *Dữ liệu được chia sẻ giữa các đối tượng thông qua các chức năng.* |
| *Function và Class* | *Không có khái niệm về Function và Class* | *Các lớp hoặc hàm có thể trở giao tiếp với nhau với từ khóa. (tùy các ngôn ngữ khác nhau thì từ khóa sẽ khác nhau)* |
| *Virtual classes hoặc virtual function* | *Không có khái niệm về lớp ảo* | *Khái niệm về chức năng ảo xuất hiện trong quá trình kế thừa.* |
| *Các ngôn ngữ thường sử dụng* | *TC, VB, GIỚI THIỆU, Pascal* | *C ++, JAVA,* [*VB.NET*](http://vb.net/)*, C # .NET, Ruby,...* |

***21 Constructor là gì?***

*Constructor (phương thức khởi tạo) được gọi để tạo đối tượng sử dụng toán tử new. Constructor là một phương thức đặc biệt. Chúng có 3 đặc điểm nổi bật sau:*

* *Constructor phải có cùng tên với tên lớp của nó*
* *Constructor không có kiểu dữ liệu trả về*
* *Constructor được gọi sử dụng toán tử new khi một đối tượng được tạo. Constructor đóng vai trò quan trọng trong việc khởi tạo đối tượng.*

***22 Mục đích của constructor là gì?***

Constructor trong Java là một kiểu phương thức đặc biệt mà *được sử dụng* để khởi tạo đối tượng. Constructor được triệu hồi tại thời gian tạo đối tượng. Nó xây dựng giá trị, cung cấp dữ liệu cho đối tượng, đó là lý do nó được gọi là Constructor.

*Mục đích của Constructor mặc định?*

Constructor mặc định cung cấp các giá trị mặc định cho đối tượng như 0, null, … tùy thuộc vào kiểu dữ liệu.

*Ví dụ Constructor mặc định mà hiển thị các giá trị mặc định*

class Student3{

int id;

String name;

*// ở đây hai thuộc tính là id và name*

// được khởi tạo bằng constructor mặc định do                       compiler cung cấp

void display(){System.out.println(id+" "+name);}

public static void main(String args[]){

Student3 s1=new Student3();

Student3 s2=new Student3();

s1.display();

s2.display();

}

}

*Giải thích: Trong lớp trên, bạn không tạo bất cứ Constructor nào, vì thế Compiler cung cấp cho bạn một Constructor mặc định. Ở đây các giá trị 0 và null được cung cấp bởi Constructor mặc định.*

***23 Constructor trả về kiểu giá trị gì?***

Constructor là 1 phương thức đặc biệt. Không có kiểu giá trị trả về.

Có, đó là instance (sự thể hiện) của lớp hiện tại. (bạn không thể sử dụng kiểu trả về, tuy vậy nó trả về một giá trị).

Nguồn: Vietjack.com

***24 Constructor được kế thừa không?*** *Không*

*\*Lớp con có thể gọi constructor của lớp cha\**

*Cách giải thích 1 : Sẽ không có ý nghĩa gì khi một hàm tạo(constructor) của lớp cha tạo ra đối tượng của lớp cha lại đi tạo ra 1 hàm tạo của lớp con.*

*Cách giải thích 2: Một lớp con kế thừa tất cả các thành viên (các trường, các phương thức và các lớp lồng nhau) từ lớp cha của nó. Các constructor không phải là thành viên, vì vậy chúng không được kế thừa bởi các lớp con, nhưng hàm tạo của lớp cha có thể được gọi từ lớp con.*

*Vd:*

*class A {*

*A();*

*}*

*class B extends A{*

*B();*

*}*

***25 Có thể tạo constructor final không?***

Không! Báo lỗi : “ Modifier 'final' not allowed here”

https://lh3.googleusercontent.com/vblph95T1Wrh7ifXtyW19Nvhq5rfrhy1dKJWXlwc8h8TGygwd94fQ8k8Ydub5vTPGgArUWH48cAobSDR0EEOrTDanHaLXuppRhROne0F2Cjc0VMbEn1wFbbp-c6XVIks5BT_jVehWMcaFeEHJQhttps://lh5.googleusercontent.com/IO6uGqyK8I5IcUbK-AJz3MtlALNBUmJtBEiwVQBu7QViI0YbFvJZbXt2MhkRhpN_EC4IN7zo3-CwlRr6KXvqwzjKcUKC2aKbj51W1d6uMaqCy81cEdm9LEdOk7vtQeftb2j-wS4_0plg008b4Q

***26 Biến static là gì?***

*Từ khóa static trong Java được sử dụng chính để quản trị bộ nhớ. Chúng ta có thể áp dụng từ khóa static với biến (cũng được gọi là biến lớp, biến class), phương thức (cũng được gọi là phương thức lớp), khối, các lớp được lặp. Từ khóa static thuộc về lớp chứ không thuộc về instance (sự thể hiện) của lớp.*

*Biến static trong Java*

*Khi bạn khai báo một biến là static, thì biến đó được gọi là biến tĩnh, hay biến static.*

*Biến static có thể được sử dụng để tham chiếu thuộc tính chung của tất cả đối tượng (mà không là duy nhất cho mỗi đối tượng), ví dụ như tên công ty của nhân viên, tên trường học của các sinh viên.*

*Biến static lấy bộ nhớ chỉ một lần trong Class Area tại thời gian tải lớp đó.*

*Lợi thế của biến static*

*Biến static giúp bộ nhớ chương trình của bạn được sử dụng hiệu quả hơn (tiết kiệm bộ nhớ).*

***27 Phương thức static là gì?***

Phương thức static trong Java

Nếu bạn áp dụng từ khóa static với bất cứ phương thức nào, thì phương thức đó được gọi là phương thức static.

* Một phương thức static thuộc lớp chứ không phải đối tượng của lớp.
* Một phương thức static có thể được triệu hồi mà không cần tạo một instance của một lớp.
* Phương thức static có thể truy cập thành viên dữ liệu static và có thể thay đổi giá trị của nó.

Một số ràng buộc

 • Phương Thức static chỉ có thể gọi các phương thức static khác

 • Phương Thức static chỉ có thể truy xuất các biến static

 • Phương Thức Static Không Thể sử Dụng Từ Khóa This hoặc super

 • Có Thể Khởi Tạo Biến Static thông qua khối khởi tạo static

• Vídụ: class Application{

public static String language;

 static {

 if(System.getProperty("lang").equals("en"))

{ language = "english";

 } else { language = "spanish"; }

} }

***28 Tại sao phương thức main là static?***

*bản thân main () là một phương thức tĩnh và do đó không chạy trên bất kỳ khởi tạo cụ thể nào của lớp; thay vào đó nó đang chạy trên chính lớp đó. Một phương thức tĩnh không thể truy cập một biến không tĩnh.*

*Điều tương tự cũng áp dụng cho các phương thức khởi tạo (phương thức không tĩnh); một phương thức tĩnh không thể gọi trực tiếp một phương thức không tĩnh.*

*Hãy luôn nhớ: static = class, non static = instance. Đặt phương thức được gọi bởi JVM (main ()) thành phương thức tĩnh có nghĩa là JVM không phải tạo một khởi tạo của lớp để bắt đầu chạy mã.*

Bởi vì không cần thiết phải tạo đối tượng để gọi phương thức static. Nếu nó là phương thức non-static, JVM đầu tiên tạo đối tượng và sau đó gọi phương thức main() mà có thể gây ra vấn đề về cấp phát bộ nhớ bộ nhớ phụ.

***29 Khối static là gì?***

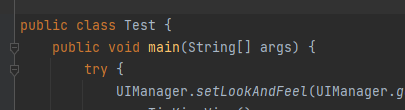
*Khối static (static blocks)*

* *Khối static được dùng để khởi tạo hoặc thay đổi giá trị của các biến static.*
* *Nó được thực thi trước phương thức main tại thời gian tải lớp.*
* *Một class có thể có nhiều static blocks.*

***30. Chúng ta có thể thực thi một chương trình không có phương thức main() không?*** Phải có lớp main() để hệ thống java runtime khởi tạo và chạy chương trình. Có, một trong các cách đó là khối static trong phiên bản trước của JDK. Không phải là JDK 1.7

***31 Chuyện gì xảy ra khi phương thức main không có static?***

Không thể chạy chương trình

https://lh4.googleusercontent.com/OzWyBoH7rNdb-Ef-RBhlddOcn5wRU5F_1BX8mDODkcKAk3GG_qt4F3EqjjLsTGLBbvRIZ2y1pJaJhZ6K8eZMtjDCG20hJ7DhVBkb39ThsRi79bcPG0GidI3QfcbMErojymUM1ThZqUVciR-KDw

***32 Sự khác nhau giữa phương thức static và phương thức instance?***

Phương thức static trực thuộc lớp

Phương thức instance thuộc đối tượng

***33 this trong java là gì?*** This là ám chỉ đối tượng đang thực hiện

***34 Kế thừa là gì?***

Kế thừa là cơ chế cho phép lớp con được thừa hưởng tất cả các đặc điểm và hành vi được định nghĩa tại lớp cha

***35 Lớp nào là lớp cha cho tất cả các lớp.*** Lớp object

***36 Tại sao đa kế thừa không được hỗ trợ trong java.***

Tuy nhiên Java không cho phép đa thừa kế.

Một lớp ở Java chỉ được kế thừa trực tiếp từ một lớp cha, đặc tính này được gọi là đơn kế thừa.

Vì một lớp kế thừa từ 2 lớp cha trở lên thì nó không biết phải thực hiện hành vi của lớp cha nào.

***40 super trong java là gì?***

Từ khóa super của lớp con, dung để gọi tới phương thức khởi tạo, constructor của,m của lớp cha mà lớp cha cho phép.

Cú pháp là super() hoặc super(parameters).

***41 Có thể sử dụng cả this() và super() trong một constructor?***

*từ khóa this tham chiếu tới bản thân đối tượng, và khi được dùng trong một khởi tạo tử, nó gọi tới khởi tạo tử của cùng một lớp. Tương tự như thế, từ khóa super tham chiếu tới lớp cha của class mà từ khóa super xuất hiện. Và nó được sử dụng theo hai cách:*

* *Để gọi tới khởi tạo tử ở lớp cha.*
* *Để gọi tới phương thức ở lớp cha.*

***42 Object cloning là gì?*** Tạo ra một bản sao giống hết đối tượng ban đầu

Object cloning là cách để tạo ra một bản sao chính xác của một đối tượng bị clone. Phương thức clone() được sử dụng để tạo ra một object mới.

Class của đối tượng mà chúng ta muốn clone phải được implements từ interface java.lang.Cloneable. Nếu chúng ta không implements interface Cloneable, phương thức clone() sinh ra lỗi ngoại lệ CloneNotSupportedException.

Phương thức phương thức clone() được định nghĩa trong lớp Object có cú pháp như sau:

protected native Object clone() throws CloneNotSupportedException;

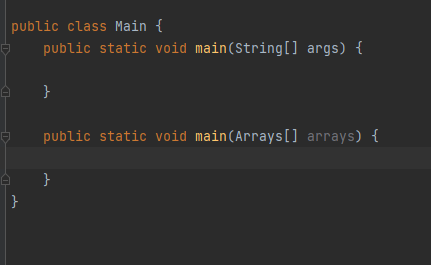
***43 Overloading (nạp chồng) phương thức là gì?***

Overloading là một kĩ thuật cho phép trong cùng một class có thể có nhiều phương thức cùng tên nhưng khác nhau về số lượng tham số hoặc kiểu dữ liệu tham số.

***44. Tại sao overloading phương thức* không xảy ra *khi thay đổi kiểu giá trị trả về?***

Trong java, nạp chồng phương thức không thể được thực hiện khi chỉ thay đổi kiểu giá trị trả về của phương thức. Kiểu giá trị trả về có thể giống hoặc khác. *Giá trị trả về có thể giống hoặc khác*, nhưng tham số phải khác nhau.

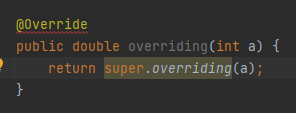
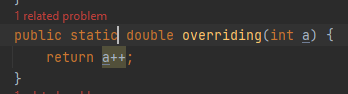
***45 Có thể overload phương thức main() không?*** Được



***46 Ghi đè (overriding) phương thức là gì?***

Là việc phương thức ở lớp con có thể định nghĩa (ghi đè) lại phương thức của lớp cha

***47 Có thể ghi đè phương thức static không?*** Không



***48 Tại sao không thể ghi đè phương thức static?***

Vì static trực thuộc lớp, không phải của đối tượng. Nên không thể ghi đè được phương thức. **vì mặc định của static đã là final**

***49 Có thể ghi đè phương thức đã nạp chồng?*** Có

***50 Có thể ghi đè biến instance không?*** Không

***51 Sự khác nhau giữa nạp chồng và ghi đè là gì?***

|  |  |  |
| --- | --- | --- |
| No. | Nạp chồng phương thức (overloading) | Ghi đè phương thức (overriding) |
| 1) | Nạp chồng phương thức được sử dụng để giúp code của chương trình *dễ đọc hơn*. | Ghi đè phương thức được sử dụng để cung cấp *cài đặt cụ thể* cho phương thức được khai báo ở lớp cha. |
| 2) | Nạp chồng được thực hiện *bên trong một class*. | Ghi đè phương thức xảy ra *trong 2 class* có quan hệ kế thừa. |
| 3) | Nạp chồng phương thức thì *tham số phải khác nhau*. | Ghi đè phương thức thì *tham số phải giống nhau.* |
| 4) | Nạp chồng phương thức là ví dụ về *đa hình lúc biên dịch*. | Ghi đè phương thức là ví dụ về *đa hình lúc runtime*. |
| 5) | Trong java, nạp chồng phương thức không thể được thực hiện khi chỉ thay đổi kiểu giá trị trả về của phương thức. Kiểu giá trị trả về có thể giống hoặc khác. *Giá trị trả về có thể giống hoặc khác*, nhưng tham số phải khác nhau. | Giá trị trả về phải giống nhau. |

***52 Biến final là gì?*** Biến final là biến không thể khai báo lại

***53 Phương thức final là gì?***

Phương thức final là phương thức không thể khai báo lại, ghi đè

***54 Lớp final là gì?*** Lớp final là lớp không cho kế thừa

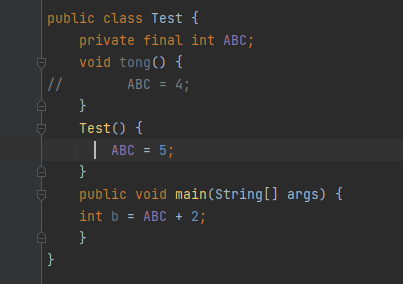
***55 Biến final blank là gì?***

***2 cách khai báo final(khai báo gán ở contrustor hoặc là ở hàm main)***

Một biến final không được khởi tạo giá trị ban đầu là biến FINAL BLANK

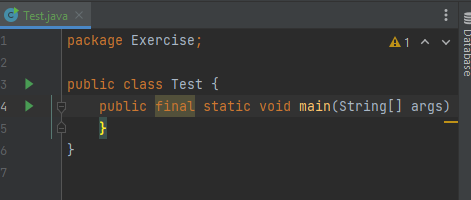
*(Nếu như vậy phải cam kết sẽ khai báo ở constructor)*

***56 Có thể khởi tạo giá trị cho biến final blank không?*** Có thể



Khởi tạo biến final blank thì phải khởi tạo nó ở Constructor. (*Như hình trên)*

***57 Có thể khai báo phương thức main là final không?*** Có!

https://lh3.googleusercontent.com/RFc9cPNjZuebOdNrNrxrMn4lHHu1rkMMlg6E_LpPgJtu7JttkKm2D2ZD0EahhOLlH6f82UI7QL8ULKp5STPWrKFResfvZMPTnWyo7JA68u5KKt4hVRLnmR9bQ-lKbQyM29oSA_d2WL9k1-3rTg

***58 Đa hình tại runtime là gì?***

*Ví dụ 1: chúng ta tạo hai lớp Bike và Splenda. Lớp Splenda kế thừa lớp Bike và ghi đè phương thức run() của nó. Chúng ta gọi phương thức run bởi biến tham chiếu của lớp cha. Khi nó tham chiếu tới đối tượng của lớp con và phương thức lớp con ghi đè phương thức của lớp cha, phương thức lớp con được triệu hồi tại runtime.*

*Khi việc gọi phương thức được quyết định bởi JVM chứ không phải Compiler, vì thế đó là đa hình tại runtime.*

*public class Bike {*

*public void run() {*

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

*}*

*}*

*public class Splendor extends Bike {*

*public void run() {*

*System.out.println("running safely with 60km");*

*}*

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

*Bike b = new Splendor(); // upcasting*

*b.run();*

*}*

*}*

***59. Có thể thực hiện đa hình lúc runtime với các thành viên dữ liệu không?***

**Không**

Phương thức bị ghi đè không là thành viên dữ liệu, vì thế đa hình tại runtime không thể có được bởi thành viên dữ liệu. Trong ví dụ sau đây, cả hai lớp có một thành viên dữ liệu là speedlimit, chúng ta truy cập thành viên dữ liệu bởi biến tham chiếu của lớp cha mà tham chiếu tới đối tượng lớp con. Khi chúng ta truy cập thành viên dữ liệu mà không bị ghi đè, thì nó sẽ luôn luôn truy cập thành viên dữ liệu của lớp cha.

class Bike{

 int speedlimit = 90;

}

class Honda3 extends Bike{

 int speedlimit = 150;

 public static void main(String args[]){

  Bike obj=new Honda3();

  System.out.println(obj.speedlimit);//90

}

***60 Sự khác nhau giữa ràng buộc tĩnh và ràng buộc động là gì?***

*Ràng buộc tĩnh*

*Khi kiểu của đối tượng được xác định tại lúc biên dịch được gọi là ràng buộc tĩnh.*

*Nếu phương thức là private, final hoặc static thì đó là ràng buộc tĩnh.*

*Ví dụ về ràng buộc tĩnh:*

*class Dog{*

*private void eat() {*

*System.out.println("dog is eating...");*

*}*

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

*Dog d1=new Dog();*

*d1.eat();*

*}*

*}*

*Ràng buộc động*

*Khi kiểu của một đối tượng được xác định tại runtime được gọi là ràng buộc động.*

*Ví dụ về ràng buộc động:*

*class Animal{*

*void eat(){System.out.println("animal is eating...");}*

*}*

*class Dog extends Animal{*

*void eat(){System.out.println("dog is eating...");}*

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

*Animal a=new Dog();*

*a.eat();*

*}*

*}*

***61 Trừu tượng là gì?*** Là sự không hình dung được, khó xác định

***62 Sự khác nhau giữa trừu tượng và đóng gói là gì?***

Trừu tượng vẫn thấy được, những khó hình dung được đó là gì.

Đóng gói: không thể thấy được bên trong

***63 Lớp trừu tượng là gì?***

Abstract Class là 1 lớp trừu tượng đến mức không thể tạo ra đối tượng

***64 Có phương thức trừu tượng không nằm trong lớp trừu tượng không****?*Có!

Một lớp abstract có thể có phương thức và thuộc tính bình thường

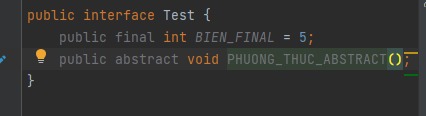
***65 Có thể sử dụng cả abstract và final cho một phương thức không?*** Không!

Nó sẽ báo lỗi “Illegal combination of modifiers: 'final' and 'abstract'”

***66 Có thể tạo thể hiện của lớp trừu tượng không?*** Không!

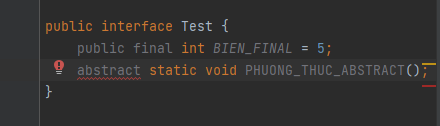
***67 Interface là gì?***

Interface là có cấu trúc tương tự như CLASS nhưng chỉ chứa final và abstract method



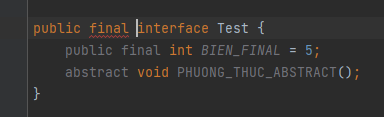
***68. Có thể khai báo một phương thức của interface với từ khóa static không?***

Không!



https://lh4.googleusercontent.com/xjXxEhcEYx4Dl9NeMTP1sEBFUqb_B4WIV3ECcHad9xYwfnwxHXOisr3VBSxnR48kn7ZUJdtpoF95btje9jcInrQyZca8oLbHKrzo5opOsIMGJ5_zac1o9YJxYwsm2UkVIJPdnLqOy0Yp5GC4UQ

***69 Một interface có thể là final không?***Không!



Báo lỗi: “Modifier “final” not allowed here”

***71 Sự khác nhau giữa lớp abstract và interface là gì?***

Abstract class

Variables:  Không có hạn chế

Constructors: Các hàm tạo được gọi bởi các lớp con thông qua chuỗi khởi tạo. Một lớp trừu tượng không được khởi tạo bằng cách sử dụng toán tử new.

Methods: Không có hạn chế.

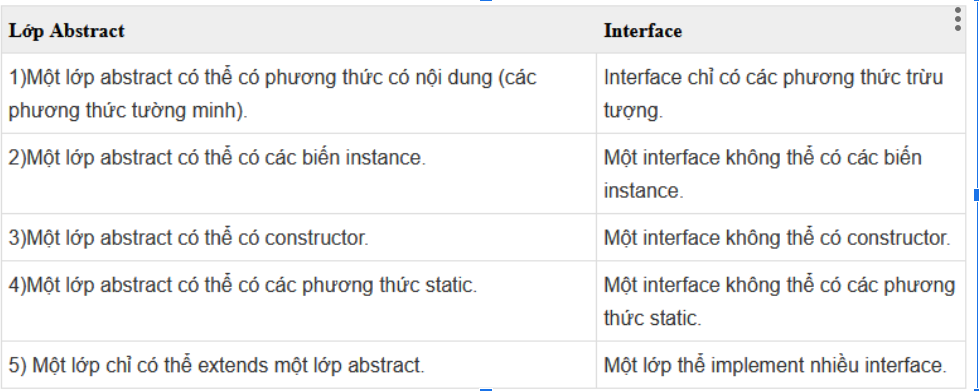
Interface

Variables : public static final

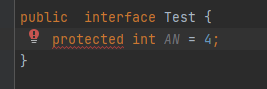
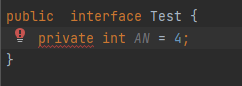
Constructors: Không có hàm tạo. Một giao diện không thể được khởi tạo

bằng cách sử dụng toán tử new.

Methods : Tất cả các phương thức phải là public, abstract



***72 Có thể định nghĩa private hoặc protected cho các biến trong interface không?*** *Không được!*

**

*Báo lỗi: “Modifier “private/ protected” not allowed here”*

***73 Khi nào một tham chiếu đối tượng có thể được ép sang kiểu interface tham chiếu?*** Chỉ cần implement interface tham chiếu

***74 Package là gì?***

*• (Package (gói) là cách để phân loại các lớp và interface thành các nhóm có liên quan đến nhau và tổ chức chúng thành các đơn vị để quản lý)*

*Một package (gói) trong Java là một nhóm các kiểu lớp, Interface và package con tương tự nhau. Package trong Java có thể được phân loại thành:*

*Package đã xây dựng sẵn và package do người dùng định nghĩa.*

*Có nhiều package đã xây dựng sẵn như java, lang, awt, javax, swing, net, io, util, sql, …*

***75 Có cần import package import java.lang không? tại sao?***

Không! Vì nó được tải ngầm cùng lúc với JVM

***76 Có thể import package/lớp giống nhau hai lần không? JVM sẽ tải package hai lần khi chạy không?***

*Nếu bạn muốn đặt hai lớp public trong một package, bạn có hai java source file chứa một lớp public nhưng giữ tên package là giống nhau. Ví dụ:*

*/\*Luu duoi dang A.java  \*/*

*package codegym;*

*public class A {*

*}*

*/\*Luu duoi dang B.java  \*/*

*package codegym;*

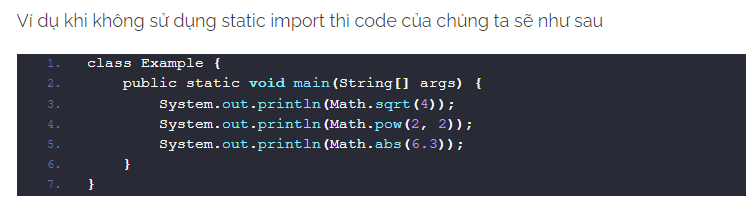
*public class B {*

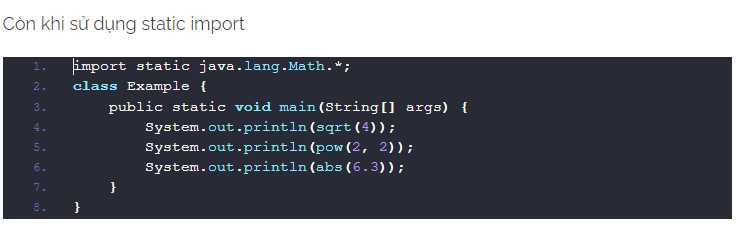
*}*

***77 Static import là gì?***

Với static import chúng ta có thể truy cập đến các thành viên của lớp một cách trực tiếp.

Không cần phải sử dụng tên class để truy cập





***78 Xử lý ngoại lệ (handling exception) là gì?***

Xử lý ngoại lệ (Exception Handling) trong java là một cơ chế xử lý các lỗi runtime để có thể duy trì luồng bình thường của ứng dụng.

Quá trình xử lý exception được gọi là catch exception, nếu Runtime System không xử lý được ngoại lệ thì chương trình sẽ kết thúc.

***79 Sự khác biệt giữa checked exception và unchecked exception là gì?***

Điểm khác biệt cơ bản giữa checked exception và unchecked exception chính là thời điểm xác định được exception.

Đơn giản:

checked là lúc chưa chạy chương trình đã báo lỗi (vd: lỗi cú pháp, lỗi logic,..)

Unchecked là lúc đã chạy chương trình mới báo lỗi.

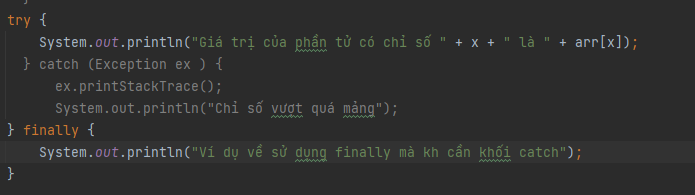
***80 Có phải mỗi khối try phải đi kèm với một khối catch?***

Không. Try đi nhiều khối catch vẫn được.

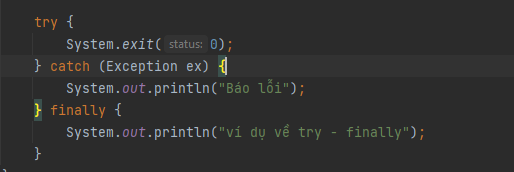
***81 Khối finally là gì?***

Khối finally trong java được sử dụng được thực thi để đóng kết nối, đóng các stream

***82 Khối finally có thể được sử dụng mà không cần khối catch không?***



***83 Có trường hợp nào khối finally không được thực thi không?* Có!**



***84 Sự khác nhau giữa throw và throws là gì?***

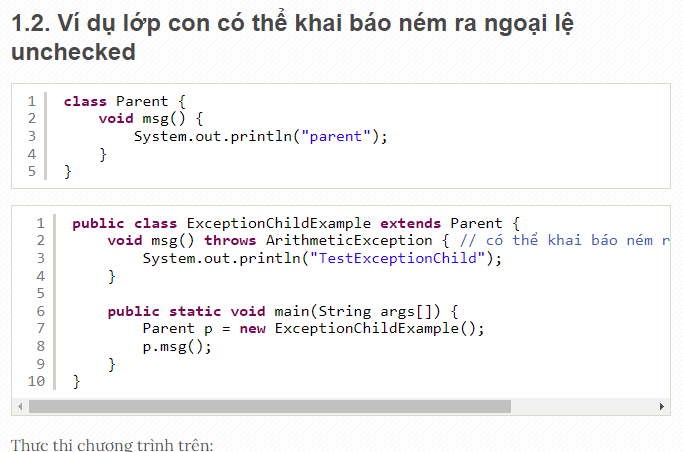
|  |  |  |
| --- | --- | --- |
| *No.* | *Throw* | *throws* |
| *1)* | *Từ khóa throw trong java được sử dụng để ném ra một ngoại lệ rõ ràng.* | *Từ khóa throws trong java được sử dụng để khai báo một ngoại lệ.* |
| *2)* | *Ngoại lệ checked không được truyền ra nếu chỉ sử dụng từ khóa throw.* | *Ngoại lệ checked được truyền ra ngay cả khi chỉ sử dụng từ khóa throws.* |
| *3)* | *Sau throw là một instance.*  *Vd: throw new Exception()* | *Sau throws là một hoặc nhiều class.*  *Throws IOException, SQLException* |
| *4)* | *Throw được sử dụng trong phương thức.* | *Throws được khai báo ngay sau dấu đóng ngoặc đơn của phương thức.* |
| *5)* | *Bạn không thể throw nhiều exceptions.* | *Bạn có thể khai báo nhiều exceptions, Ví dụ:*  *public void method()throws IOException,SQLException.* |

***85 Có thể khai báo phương thức overriding của lớp con một ngoại lệ nếu phương thức của lớp cha không throw một ngoại lệ?***

Nếu phương thức của lớp cha không khai báo ném ra exception, phương thức được ghi đè của lớp cha không thể khai báo ném ra ngoại lệ checked, nhưng ngoại lệ unchecked thì có thể



Compli runtime error

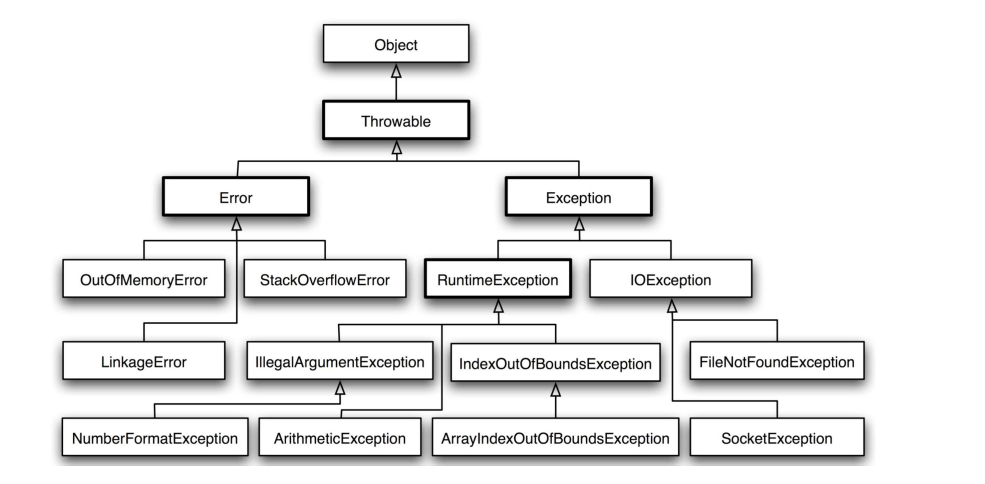


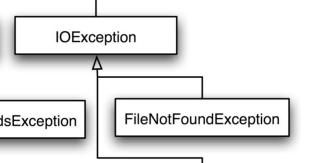
TestExceptionChild

***86 Việc tuyên truyền ngoại lệ là gì?***

***87 Có vấn đề gì không với thứ tự của các lệnh catch được viết cho***

*FileNotFoundException và IOException?* Vẫn được



Nhìn vào đây ta có thể thấy, FileNotFoundException là lớp con của IOException, việc này sẽ không ảnh hưởng tới việc ném ra các exception

***88 Điểm khác nhau giữa Error và Exception?***

- *Error* là lỗi của chương trình, còn exception là lỗi chương trình được phát hiện và có thể xử lý.

* Error: không thể xử lý bằng mã chương trình
* exception: có thể xử lý bằng mã

***91 Có bao nhiêu cách để tạo ra một đối tượng String trong java?***

Có hai cách để tạo đối tượng String

1. Sử dụng string literal

Vd:

https://lh3.googleusercontent.com/-yaGnk5PUmkRcC-JZwTWY0x9ZoNwhVbZXAOPZn-dd68PRkmDkrSq_M7u1o7zwYzyOZrlgDy2U4Up6C39EH4OYNMq4QJeTG9RtnLUkfOBfugwn89tRt7raYUmJ9rMcj1uJ_cb-1mjqOmIwDiq3g

1. Sử dụng từ khóa new

*https://lh3.googleusercontent.com/sT-R8knSBhZbSULTAHS8FmxSqcGh7_dOvgK5w-5R0zdJ8sMi4eLTIU_yiJXwxI1sPXM_Zheo0wzcq3dxW1ZNkrqhvEylCpy8JoIYbsLTQK6GhiCXt-8X42SvZK6fkLECqsRZYUZCmV_1vP4ojQ*

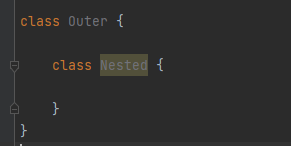
***98 Mục đích của phương thức toString() trong java là gì?***

Là phương thức thể hiện của đối tượng trả về kiểu dữ liệu chuỗi

***99 Em biết gì về String Pool?***

String pool là một vùng nhớ đặc biệt nằm trong vùng nhớ Heap (Heap memory), dùng để lưu trữ các biến được khai theo kiểu String.

***100 Nested class (lớp lồng nhau) là gì?***



Một lớp outer được bao bọc bên ngoài lớp nested.

***101 Có sự khác nhau giữa nested class và inner class không?***

*Điểm khác nhau giữa Inner Class và Nested Class trong Java*

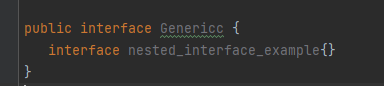
*Inner Class là một phần của Nested Class. Trong Nested Class, có hai loại là non-static và static. Các Nested Class mà không phải ở dạng static thì đó là Inner Class. Trong đó, loại Non-static Nested Class bao gồm:*

* *Member Inner Class: là một lớp được tạo bên trong lớp và bên ngoài phương thức.*
* *Anonymous Inner Class: là một lớp được tạo để triển khai (kế thừa) Interface hoặc kế thừa lớp. Tên của nó được quyết định bởi Compiler.*
* *Local Inner Class: Một lớp được tạo bên trong phương thức.*

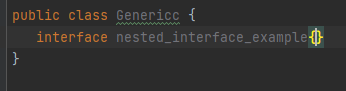
*Loại static Nested Class: một lớp static được tạo bên trong một lớp.*

*102 Nested interface là gì?*

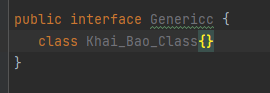
***Một interface được tạo ra bên trong một lớp hoặc một interface.***

**

***103 Có thể khai báo interface trong class không?*** Có

**

***104 Có thể khai báo một class trong interface không?*** Có

**

***105 Sự khác nhau giữa Array và ArrayList là gì?***

*Sự khác nhau giữa Array và ArrayList được tóm tắt lại như trong bảng sau:*

|  |  |
| --- | --- |
| *Array* | *ArrayList* |
| *1) Kích thước cố định.* | *Kích thước có thể thay đổi được.* |
| *2) Có thể lưu trữ dữ liệu kiểu nguyên thủy và đối tượng.* | *Chỉ có thể lưu trữ dữ liệu kiểu đối tượng. Kể từ Java 5, kiểu nguyên thủy được tự động chuyển đổi trong các đối tượng được gọi là auto-boxing.* |
| *3) Tốc độ lưu trữ và thao tác nhanh hơn.* | *Tốc độ lưu trữ vào thao tác chậm hơn.* |
| *4) Chỉ có thuộc tính length.* | *Có nhiều phương thức để thao tác với dữ liệu.* |

Array:

* 1.Tính chất tĩnh
* 2.Có chiều dài cố định .. vd arrays[10] => chiều dài là 10
* 3.Không thể sử dụng Generic

ArrayList

* 1.Tính chất động
* 2.có thể định hình tùy thuộc vào việc tải
* 3. Có thể sử dụng Generic
* 4. Cung cấp phương thức size() để tính toán mảng arraylist
* 5. ArrayList chỉ có thể chứa *kiểu dữ liệu ĐỐI TƯỢNG*

***106 Lợi thế của ArrayList so với các mảng?***

* ArrayList cung cấp nhiều phương thức để làm việc : add(), get() contains()

***107 Hoạt động xóa trong LinkedList là nhanh hơn trong ArrayList, tại sao?***

ArrayList muốn *truy xuất* đến phần tử THỨ mấy trong danh sách chỉ cần gọi vị trí đó ra là được.

LinkedList phải duyệt qua các phần tử ( tức là duyệt qua các NODE) thì mới truy xuất được đến phần tử cần lấy

***108 Bạn quyết định khi nào sử dụng ArrayList và LinkedList?***

- Nếu *thường xuyên* truy cập vào danh sách, thêm hoặc xóa các phần tử theo dãy thì sử dụng LINKEDLIST

- Nếu cần *truy cập ngẫu nhiên* MÀ không cần chèn, xóa phần tử thì sử dụng ArrayList

***109 Sự khác nhau giữa ArrayList và LinkedList là gì?***

|  |  |
| --- | --- |
| ArrayList | LinkedList |
| 1) ArrayList nội bộ sử dụng mảng động để lưu trữ các phần tử. | LinkedList nội bộ sử dụng danh sách liên kết doubly để lưu trữ các phần tử. |
| 2) Thao tác với ArrayList là chậm bởi vì nó sử dụng nội bộ mảng. Nếu bất kỳ phần tử nào được xóa khỏi mảng, tất cả các bit được chuyển trong bộ nhớ. | Thao tác với LinkedList là nhanh hơn so với ArrayList bởi vì nó sử dụng danh sách liên kết doubly do đó không cần chuyển đổi bit nào trong bộ nhớ. |
| 3) [Lớp ArrayList trong java](https://viettuts.vn/java-collection/arraylist-trong-java) chỉ có thể hoạt động như một list vì nó chỉ implements giao tiếp List. | [Lớp LinkedList trong java](https://viettuts.vn/java-collection/linkedlist-trong-java) có thể hoạt động như một *list và queue(hàng đợi) vì nó implements các giao tiếp List và Deque*. |
| 4) ArrayList là tốt hơn trong việc lưu trữ và truy cập dữ liệu. | LinkedList là tốt hơn trong việc *thao tác dữ liệu* |

***111 Sự khác nhau giữa Iterator và ListIterator là gì?***

Iterator duyệt các phần tử chỉ theo một chiều hướng là chuyển tiếp, trong khi listiterator duyệt các phần tử theo hai hướng là chuyển tiếp và ngược lại.

Iterator có thể được sử dụng trong List, Set, Queue

ListIterator chỉ có thể  được sử dụng trong List

***112 Sự khác nhau giữa List và Set là gì?***

List: sắp xếp danh sách các phần tử có thể TRÙNG LẶP

Set: sắp xếp danh sách các phần tử không trùng lặp

***113 Sự khác nhau giữa Set và Map là gì?***

Set chỉ chứa value

Map chứa các cặp key, value <cặp entry> <key/value>

***114 Sự khác nhau giữa Stack và Queue là gì?***

Stack (ngăn xếp):

* sắp xếp phần tử trong danh sách theo cơ chế FILO ( First In - Last Out)
* Số con trỏ sử dụng : MỘT

Queue (hàng đợi):

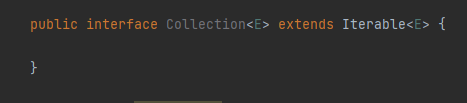
* sắp xếp phần tử trong danh sách theo cơ chế FIFO ( First In - First Out)
* Nó có thể trỏ tới hai đầu (trước) (sau)
* Có những biến thể khác : hàng đợi vòng tròn, hàng đợi ưu tiên (priotity queue),hàng đợi kết thúc

***115 Sự khác nhau giữa Collection(là một Interface) và Collections (là một class) là gì?***

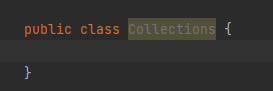
Collection là một INTERFACE cao cấp nhất của Collection Framework.

Trong khi đó, collections bao gồm các phương thức STATIC được sử dụng để thao tác trên các đối tượng của Collection (list, arraylist, linkedlist, map, set ,.. )

Interface java.util.Collection được định nghĩa như sau:



Interface java.util.Collections được định nghĩa như sau:



***116 Sự khác nhau giữa Comparable và Comparator là gì?***

**Comparable và Comparator (không phụ thuộc comparable, tùy biến hơn) cả hai đều là các** giao diện và có thể được sử dụng để *sắp xếp* các phần tử của Collection.

|  |  |
| --- | --- |
| Comparable | Comparator |
| 1) Bạn phải implements giao tiếp comparable cho đối tượng cần được so sánhhttps://lh4.googleusercontent.com/utAFYCdxPYcSJ0DK_7QqjUSsGN3ZHoSSP3HjhPwK4Ioo1P3eJTqP3csmicocWf92oEul8YfcmnLb-TXX4_uH08gDPZDvcL_93q3c3n0aTO6RVHMhK-BcJKGzMRYO09jKyVOG6e8GofIXsI3H5Q | Bạn KHÔNG cần phải implements giao tiếp Comparator cho lớp đối tượng cần so sánh |
| 2)Comparable cung cấp phương thức compareTo() để sắp xếp các phần tử  https://lh5.googleusercontent.com/Aypp8ezez2GvNmDA2M-I_33bFRZM_n2G94-d2F8GBMjjo1P5GdkDmAcJlqPdZSnURAeGoe8IuLV8sRY-khY9LMizsxi9m6Ib6aA5Ll6hyB1teXrWAIm-gsvxGiWK2boy9cEuUFrOxVMFxKYRFg | Comparator cung cấp phương thức compare() để sắp xếp các phần tử |
| 3) Comparable thuộc về java.lang package. | Comparator thuộc về java.util package. |
| 4) Chúng ta có thể sắp xếp các phần tử của kiểu Comparable bởi phương thức Collections.sort(List) . | Chúng ta có thể sắp xếp các phần tử của kiểu Comparator bởi phương thức Collections.sort(List,Comparator). |

***117 Phương thức hashCode() là gì? đọc lại hashCode()?***

Phương thức hashCode() chuỗi để định danh, trả về một giá trị băm (số nguyên)

***118 Tại sao chúng ta phải ghi đè phương thức equals()?***

Cài đặt mặc định của lớp equals () do Java.lang.Object

so sánh vị trí bộ nhớ và

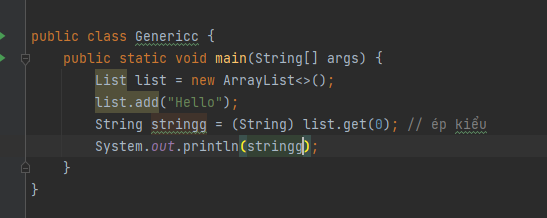
chỉ trả về true nếu hai biến tham chiếu đang trỏ đến cùng một vị trí bộ nhớ,

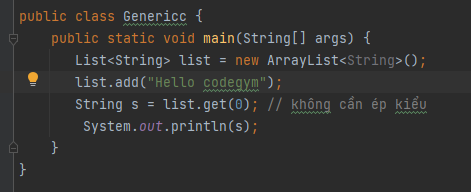
nghĩa là về cơ bản chúng là cùng một đối tượng.

***120 Lợi ích của generic collection là gì?***

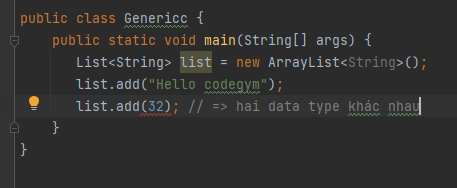
1. Kiểu đối tượng an toàn: Chúng ta chỉ có thể lưu một kiểu đối tượng duy nhất trong generics. Nó không cho phép lưu trữ 2 đối tượng có kiểu khác nhau.

2. Không cần phải ép kiểu:





3. Kiểm tra lúc biên dịch:



https://lh5.googleusercontent.com/E3UyigvVbhmXw5sp_QFLDea9vyYBiNq_IgrA4Z7ajK5itA0rMiL9T-xrWUITc0br7dNrSPtIN0-LjBQDVla49pOwkuB3pZngOqGV1A0ZNRVkXyNkySNnA179nasC4cylGKXeB6qVDv_yJ3e3yQ

***121 Tại sao Generic được sử dụng trong Java?***

Việc thao tác với Generic sẽ giúp tiết kiệm thời gian, nhanh

Java Generics cho phép chúng ta viết một phương thức duy nhất có thể thực hiện các hoạt động trong nhiều loại đối tượng hỗ trợ phương thức đó.

CÓ thể: Sử dụng nhiều tham số kiểu trong các lớp Generics trong Java

Vd: Generic<String, Int>

***122 Bạn hiểu gì về Access Modifier?***

Có hai loại Modifier trong Java, đó là: Access Modifier và Non-access Modifier. Access Modifier trong Java xác định phạm vi có thể truy cập của thành viên dữ liệu, phương thức, constructor hoặc lớp. Có 4 loại Access Modifier là: private, default, protected, và public.

* default: Truy cập trong nội bộ package
* private: Truy cập trong nội bộ lớp
* public: Thành phần công khai, truy cập tự do từ bên ngoài
* protected: Thành phần được bảo vệ, bị hạn chế truy nhập từ bên ngoài

***123 Phân biệt public, protected, private, default?***

*- public: có thể truy cập từ bất cứ đâu*

*- private: các phương thức và thuộc tính chỉ được phép truy xuất trong cùng một lớp*

*• protected: các phương thức và thuộc tính được phép truy xuất trong cùng một lớp và ở các lớp con(kế thừa)*

*• default: Nếu Không Có Access Modifier thì mức default sẽ được áp dụng. Lớp và các thành phần của lớp được truy xuất ở những nơi trong cùng một package*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Access Modifier* | *Bên trong lớp* | *Bên trong package* | *Bên ngoài package chỉ bởi lớp con* | *Bên ngoài package* |
| *private* | *C* | *K* | *K* | *K* |
| *default* | *C* | *C* | *K* | *K* |
| *protected* | *C* | *C* | *C* | *K* |
| *public* | *C* | *C* | *C* | *C* |

***124 Bạn hiểu gì về Đa luồng (Multi-Thread)?***

Multi-thread (đa luồng) là tiến trình thực hiện nhiều tiến trình  đồng thời.

+ Khi play nhạc, chúng ta vẫn có thể tương tác được với nút điều khiển như: Play, pause, next, back … vì luồng phát nhạc là luồng riêng biệt với luồng tiếp nhận tương tác của người dùng.

+ Ở exel. Việc mở nhiều sheet và tương tác cũng là ví dụ về thread

***125 Phân biệt Multi-tasking, multi-processing, multi-threading***

Multitasking*: Là khả năng chạy đồng thời một hoặc nhiều chương trình cùng một lúc trên một hệ điều hành. Hệ điều hành quản lý việc này và sắp xếp lịch phù hợp cho các chương trình đó. Ví dụ, trên hệ điều hành Windows chúng ta có làm việc đồng thời với các chương trình khác nhau như: Microsoft Word, Excel, Media Player,…*

*Multi-thread (đa luồng) là một tiến trình thực hiện nhiều luồng đồng thời. Một ứng dụng Java ngoài luồng chính có thể có các luồng khác thực thi đồng thời làm ứng dụng chạy nhanh và hiệu quả hơn.*

*VD: Trình duyệt web hay các chương trình chơi nhạc là 1 ví dụ điển hình về đa luồng.*

*Đa tiến trình (Multiprocessing).*

* *Mỗi tiến trình có địa chỉ riêng trong bộ nhớ, tức là mỗi tiến trình phân bổ vùng nhớ riêng biệt.*
* *Tiến trình là nặng.*
* *Sự giao tiếp giữa các tiến trình có chi phí cao.*
* *Chuyển đổi từ tiến trình này sang tiến trình khác đòi hỏi thời gian để đăng ký việc lưu và tải các bản đồ bộ nhớ, các danh sách cập nhật, …*

***126 Thread có thể được tạo bằng hai cách nào? So sánh***

Thread có thể tạo bằng hai cách:

1. Kế thừa lớp Thread

1. implements lại lớp Runnable

***127 Phân biệt run(), start()***

*Cả 2 phương thức này đều có chức năng gọi đến phương thức run() của thread và thực thi các dòng code bên trong phương thức này. Vậy sự khác nhau giữa chúng là:*

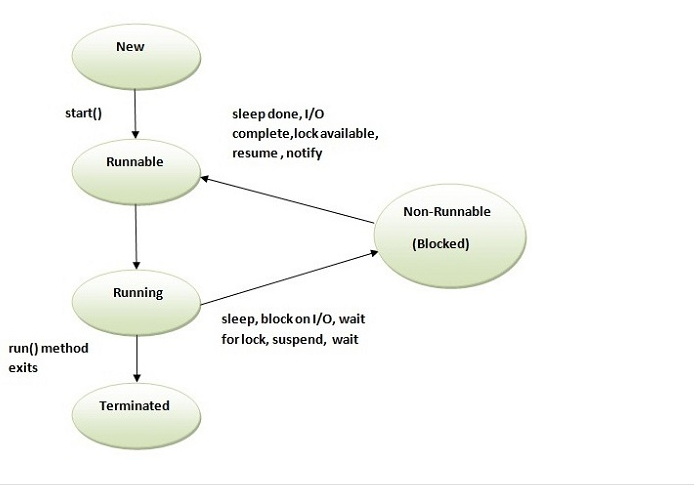
*- Khi chúng ta gọi phương thức start(), 1 thread mới sẽ được tạo và code trong phương thức run() sẽ được thực thi trong thread mới này. Nếu chúng ta gọi phương thức run(), không có thread mới nào được tạo và code trong run() sẽ được thực thi tại thread hiện tại.*

*- Ngoài ra, chúng ta không thể gọi phương thức start() 2 lần trên đối tượng thread, việc này sẽ quăng ra lỗi IllegalStateException trong Java. Trong khi đó, chúng ta có thể gọi phương thức run() 2 lần trên đối tượng thread.*

***128 Vòng đời Thread***

*Vòng đời của thread trong java được kiểm soát bởi JVM. Các trạng thái của thread java như sau:*

1. *New*
2. *Runnable*
3. *Running*
4. *Non-Runnable (Blocked)*
5. *Terminated*

**

***129 Giải thích trình dọn rác (Garbage Collector) trong Java?***

Ưu điểm của Garbage Collection

* Nó làm cho việc sử dụng bộ nhớ java hiệu quả bởi vì bộ thu gom rác (Garbage Collection) loại bỏ các đối tượng không được tham chiếu ra khỏi bộ nhớ heap.
* GC là một phần của JVM và nó được được thực thi tự động, vì vậy chúng ta không cần phải nỗ lực nhiều để giải phóng bộ nhớ.

Garbage Collection (Bộ thu gom rác) trong Java được định nghĩa như một quá trình tự động thực thi nhiệm vụ quản lý bộ nhớ.

Code Java được dịch sang bytecode rồi chạy trên máy ảo Java hay viết tắt là JVM. Trong quá trình chạy chương trình, các đối tượng được tạo ở vùng nhớ heap, một phần bộ nhớ dành cho chương trình. Sau cùng, sẽ có một vài đối tượng mà chương trình không cần dùng đến. Các đối tượng này sẽ được garbage collector truy tìm và xóa bỏ để thu hồi lại dung lượng bộ nhớ.

***130 Các cách mà một Thread có thể đi vào trạng thái đợi (waiting state)?***

*Một Thread có thể đi vào trạng thái đợi (Waiting state) bằng việc triệu hồi phương thức sleep() của nó, bằng việc được khóa trên IO, hoặc thất bại trong việc cố gắng thu được lock của đối tượng, hoặc bởi triệu hồi phương thức wait() của đối tượng. Nó cũng có thể đi vào trạng thái đợi bởi triệu hồi phương thức suspend() của nó (phương thức này đã cũ).*

***131. Cách mà Đa luồng (Multi-Thread) diễn ra trên một máy tính với một CPU đơn?***

*Scheduler của hệ điều hành cấp phát thời gian thực thi cho các Task. Bằng việc nhanh nhóng chuyển đổi giữa các Task đang thực thi, nó tạo cho chúng ta cảm tưởng rằng các Task này được thực thi đồng thời.*

***134 Phương thức sleep() và wait() khác nhau ở điểm nào?***

*Khi wait() method được gọi, thread hiện tại sẽ rơi vào trạng thái chờ cho đến khi một thread khác đánh thức nó,* hoặc *tự đánh thức sau một khoảng thời gian được chỉ định.*

*sleep() sẽ tạm dừng thread hiện tại cho đến khi nó được đánh thức bởi khoảng thời gian được chỉ định.*

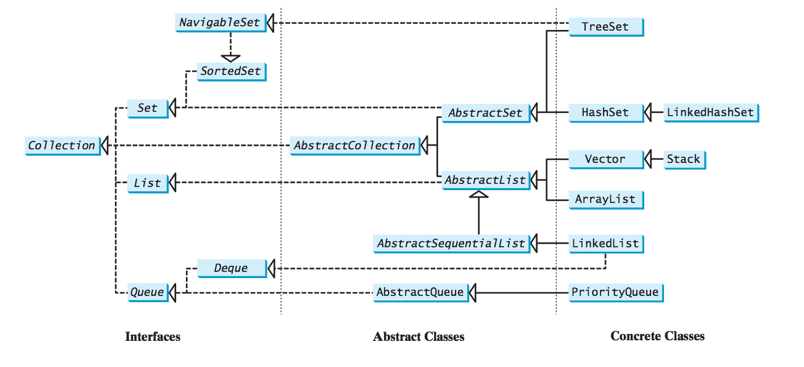
1. Từ khóa this trong java

Từ khóa this trong java là một biến tham chiếu được sử dụng để tham chiếu tới đối tượng của lớp hiện tại.

Từ khóa this có 6 cách sử dụng sau:

* Tham chiếu tới biến instance của lớp hiện tại.
* Gọi phương thức (method) của lớp hiện tại.
* Gọi hàm dựng (constructor) của lớp hiện tại.
* Trả về instance của lớp hiện tại
* Được truyền như một tham số trong phương thức (method).
* Được truyền như một tham số trong hàm dựng (constructor).

Tính đóng gói (Encapsulation) là khả năng che giấu thông tin của đối tượng với môi trường bên ngoài.



Cú pháp biểu thức chính quy

Một biểu thức chính quy bao gồm các ký tự nguyên thủy và các ký hiệu đặc biệt.Phía dưới là bảng liệt kê một số cú pháp được sử dụng thường xuyên cho các biểu thức thông thường.

|  |  |  |
| --- | --- | --- |
| T | Biểu thức chính quy | Mô tả |
| 1 | . | Khớp (match) với bất kỳ ký tự nào |
| 2 | ^regex | Biểu thức chính quy phải  khớp tại điểm bắt đầu |
| 3 | regex$ | Biểu thức chính quy phải khớp ở cuối dòng. |
| 4 | [abc] | Thiết lập định nghĩa, có thể khớp với a hoặc b hoặc c. |
| 5 | [abc][vz] | Thiết lập định nghĩa, có thể khớp với a hoặc b hoặc c theo sau là v hay z. |
| 6 | [^abc] | Khi dấu ^ xuất hiện như là nhân vật đầu tiên trong dấu ngoặc vuông, nó phủ nhận mô hình. Điều này có thể khớp với bất kỳ ký tự nào ngoại trừ a hoặc b hoặc c. |
| 7 | [a-d1-7] | Phạm vi: phù hợp với một chuỗi giữa a và điểm d và con số từ 1 đến 7. |
| 8 | X|Z | Tìm X hoặc Z. |
| 9 | XZ | Tìm X và theo sau là Z. |
| 10 | $ | Kiểm tra kết thúc dòng. |
|  | | |
| 11 | \d | Số bất kỳ, viết ngắn gọn cho [0-9] |
| 12 | \D | Ký tự không phải là số, viết ngắn gon cho [^0-9] |
| 13 | \s | Ký tự khoảng trắng, viết ngắn gọn cho [ \t\n\x0b\r\f] |
| 14 | \S | Ký tự không phải khoản trắng, viết ngắn gọn cho [^\s] |
| 15 | \w | Ký tự chữ, viết ngắn gọn cho [a-zA-Z\_0-9] |
| 16 | \W | Ký tự không phải chữ, viết ngắn gọn cho [^\w] |
| 17 | \S+ | Một số ký tự không phải khoảng trắng (Một hoặc nhiều) |
| 18 | \b | Ký tự thuộc a-z hoặc A-Z hoặc 0-9 hoặc \_, viết ngắn gọn cho [a-zA-Z0-9\_]. |
|  | | |
| 19 | \* | Xuất hiện 0 hoặc nhiều lần, viết ngắn gọn cho {0,} |
| 20 | + | Xuất hiện 1 hoặc nhiều lần, viết ngắn gọn cho {1,} |
| 21 | ? | Xuất hiện 0 hoặc 1 lần, ? viết ngắn gọn cho {0,1}. |
| 22 | {X} | Xuất hiện X lần, {} |
| 23 | {X,Y} | Xuất hiện trong khoảng X tới Y lần. |
| 24 | \*? | \* có nghĩa là xuất hiện 0 hoặc nhiều lần, thêm ? phía sau nghĩa là tìm kiếm khớp nhỏ nhất. |