1. Sự khác nhau của JVM,JRE,JDK:

JDK: là công cụ cho phép lập trình viên phát triển các ứng dụng của Java

JRE:môi trường thực thi

JVM:máy chủ

1. Khác nhau giữa bộ nhớ heap và stack:

Heap dùng để lưu đối tượng

Stack dùng để lưu biến cục bộ, hàm…

1. Biên dịch là quá trình dịch toàn bộ chương trình và không cần dịch lại 1 lần nào nữa
2. Platfom là ngôn ngữ cho phép lập trình viên xây dựng các ứng dụng khác nhau:

Java SE:lập trình hướng đối tượng

Java EE: xâu dựng wed

Java ME:xây dựng các ứng dụng điện thoại

Java FX: xây dựng các ứng dụng desktop

Android SDK: xây dựng các ứng dụng mobile

1. Nó là 1 nền tảng dựa trên các phần mềm chạy trên các nền tảng khác.Có 2 phần:

-Môi trường thực thi

-API

1. Java được biên dich thành 1 byte code, là 1 ngôn ngữ trung gian giữa mã nguồn và mã máy, byte code không phải 1 nền tảng cụ thể nên có thể thông dich trên bất kì nền tảng nào
2. Classcoder là hệ thống con của JVM được sử dụng để tải các lớp và interface.  
   Ví dụ: system, exception
3. Hợp lệ
4. Không
5. Empty
6. Chương trình vẫn biên dịch và run
7. Các biến local không được khởi tạo bởi bất kì giá trị nào
8. Tham trị: truyền vào p.thức bằng cách copy giá trị của đối số vào tham số  
   Tham chiếu: truyền vào p.thức bằng cách copy địa chỉ của đối số vào tham số
9. Đối tượng là những gì có trong cuộc sống được ltv phát hoạ lại qua các ngôn ngữ lập trình
10. Class là khuôn mẫu định nghĩa ra các thuộc tính hành vị cho các đối tượng tương đồng
11. Kiểu biến có thể chứa: local, instance, lớp
12. Biến dùng để khai báo method
13. Biến dùng để khai báo các thuộc tính của đối tượng
14. Ngôn ngữ lập trình dựa trên đối tượng thực hiện tất cả các tính năng trừ kế thừa
15. Null
16. Contructor như là 1 phương thức khởi tạo trạng thái đối tượng.Nó được gọi ra trong thời điểm tạo đối tượng
17. Cung cấp các giá trị mặc định cho đối tượng
18. Trả về đối tượng hiện thời
19. Không
20. Không
21. Là biến được đc đánh dấu bằng từ khoá stactic, tham chiếu đến các thuộc tính chung của đối tượng
22. Là 1 phương thức có từ khoá stactic
23. Để máy chủ gọi nó mà không tạo bất cứ đối tượng nào
24. Lưu trữ các thành phần chứa static
25. Sử dụng các khối static
26. C.trình vẫn biên dịch nhưng bắt lỗi

|  |  |
| --- | --- |
| **Phương thức static** | **Phương thức instance** |
| 1)Một phương thức được khai báo với từ khóa static được gọi là phương thức static. | Một phương thức không được khai báo với từ khóa static được gọi là phương thức instance. |
| 2)Không cần tạo đối tượng cũng gọi được phương thức static thông qua class. | Phải tạo đối tượng để gọi phương thức instance. |
| 3)Biến non-static không được truy cập trực tiếp trong phương thức static (hoặc khối static). | Biến static và non-static được truy cập trực tiếp trong phương thức instance. |

1. Tham chiếu đến đối tượng hiện tại
2. Kế thừa là 1 cơ chế trong đó 1 lớp thừa hưởng tất cả thuộc tính phương thức của lớp khác
3. Lớp object
4. Giảm thiểu sự phức tạp và đơn giản hoá code
5. Khai báo biến tham chiếu của 1 class trong class khác
6. Aggregation biểu diễn mối quan hệ yếu, còn composition biểu diễn quan hệ chặt chẽ.
7. Vì không an toàn và phức tạp
8. Từ khoá tham chiếu trực tiếp tới lớp cha
9. Không vì this() gọi tới lớp hiện tại, còn super gọi tới lớp cha
10. Tạo ra bản sao y hệt đối tượng ban đầu
11. Nếu 1 lớp có nhiều phương thức giống nhau nhưng tham số khác nhau
12. Không rõ ràng
13. Có
14. Nếu lớp con có phương thức giống lớp cha
15. Không vì static thuộc class ko phải thuộc đối tượng
16. static thuộc class ko phải thuộc đối tượng
17. có
18. không
19. trong vở

|  |  |  |
| --- | --- | --- |
| **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. |

1. Là biến không thể thay đổi giá trị của nó
2. Là phương thức không thể ghi đè
3. Là lớp không thể kế thừa
4. Là biến final không khởi tạo giá trị ban đầu
5. Có
6. Có
7. Đa hình tại runtime là quá trình gọi phương thức đã được ghi đè trong thời gian thực thi chương trình. Trong quá trình này, một phương thức được ghi đè được gọi thông qua biến tham chiếu của một lớp cha.
8. Không
9. Kiểu ràng buộc tĩnh của đối tượng được xác định tại lúc biên dịch, còn kiểu ràng buộc động của đối tượng được xác định tại runtime.
10. ẩn đi các chi tiết triển khai, chỉ hiển thị kết quả/ tính năng
11. Trừu tượng là ẩn đi cài đặt chi tiết còn đóng gói là gói code và data vào một khối duy nhất.
12. Là lớp được khai báo là abstract
13. Không
14. Không
15. Không
16. Là 1 bản thiết kế của 1 lớp
17. Không
18. Không
19. Là 1 interface không có phương thức

|  |  |
| --- | --- |
| **Lớp Abstract** | **Interface** |
| 1)Một lớp abstract có thể có phương thức có nội dung (các phương thức tường minh). | Interface chỉ có các phương thức trừu tượng. |
| 2)Một lớp abstract có thể có các biến instance. | Một interface không thể có các biến instance. |
| 3)Một lớp abstract có thể có constructor. | Một interface không thể có constructor. |
| 4)Một lớp abstract có thể có các phương thức static. | Một interface không thể có các phương thức static. |
| 5) Một lớp chỉ có thể extends một lớp abstract. | Một lớp thể implement nhiều interface. |

1. Không
2. Khi đối tượng đấy implement interface được tham chiếu
3. Là gói trong java 1 nhóm các kiểu tương tự trong 1 lớp, interface, package con
4. Không, vì nó tải ngầm dưới JVM
5. Được, JVM tải lớp đó 1 lần
6. Truy cập đến các thành viên trực tiếp không cần phải sử dụng tên class để truy cập
7. Là cơ chế xử lí lỗi runtime để c.trình chạy 1 cách bth
8. -Các lớp extend throwable ngoại trừ RuntimeException và error được gọi là checked, Checked exception được kiểm tra tại thời gian biên dịch.  
   -Các lớp được extends lớp RuntimeException được gọi là unchecked exception. Ví dụ như ArithmeticException, NullPointerException, ... Unchecked exception không được kiểm tra tại thời gian biên dịch.
9. Có thể theo sau là 1 catch hoặc finally
10. Khối finally trong là một khối được sử dụng để thực thi các phần code quan trọng như đóng kết nối, đóng stream, … Khối finally luôn luôn được thực thi dù cho exception có được xử lý hay không.
11. Có
12. Khối finally không được thực thi nếu chương trình bị thoát (bằng cách gọi System.exit() hoặc lỗi phần cứng.

|  |  |  |
| --- | --- | --- |
| **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. | Sau throws là một hoặc nhiều class. |
| 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. |

1. Được nhưng chỉ đối với unchecked exception còn checked exception thì không.
2. Việc chuyển tiếp đối tượng ngoại lệ sang phương thức gọi được gọi là sự truyền ngoại lệ.
3. Có. FileNotFoundException được kế thừa từ IOException. Các lớp con của Exception phải được bắt đầu tiên.
4. Error: là lỗi toàn bộ c.trình và lập trình viên không thể xử lí  
   Exception: là những bất thường mà lập trình viên có thể xử lí
5. Ý nghĩa đơn giản của immutable là không thể sửa đổi hoặc không thể thay đổi được. Một khi đối tượng String đã được tạo ra, giá trị của nó không thể thay đổi.
6. Bởi vì java sử dụng khái niệm về string literal. Giả sử có 5 biến tham chiếu, tất cả các tham chiếu đến một đối tượng "hello". Nếu một biến tham chiếu thay đổi giá trị của đối tượng, nó sẽ bị ảnh hưởng đến tất cả các biến tham chiếu khác. Đó là lý do tại sao đối tượng String trong java immutable (bất biến).
7. Có 2 cách:  
   dùng string literal  
   dùng từ khoá new
8. ….
9. Vì điều này khiến Java sử dụng bộ nhớ hiệu quả hơn (vì không có đối tượng String mới được tạo ra trong string constant pool).
10. ….
11. String là một đối tượng immutable (bất biến). StringBuffer là môt đối tượng mutable (có thể biến đổi).
12. StringBuffer là đồng bộ còn StringBuilder là KHÔNG đồng bộ.
13. Chúng ta có thể tạo ra lớp bất biến bằng cách khai báo class là final và thành viên dữ liệu của nó cũng là final
14. Phương thức toString() trả về biểu diễn chuỗi của bất kỳ đối tượng nào
15. String pool là một vùng nhớ đặc biệt nằm trong vùng nhớ Heap (Heap memory), dùng để lưu trữ các biến được khai báo theo kiểu String
16. Một lớp được khai báo bên trong một lớp khác được gọi là lớp lồng nhau. Có 4 loại lớp bên trong lớp lồng nhau: member inner class, local inner class, annonymous inner class và static nested.
17. Có. Inner class là non-static nested class nghĩa là inner class là một kiểu nested class.
18. Bất kỳ interface nào được khai báo bên trong interface hoặc class, được biết đến nested interface. Nó có thể là static hoặc default.
19. Có. Nó được biết đến như nested interface.
20. Có. Nhưng chúng không liên quan đến nhau.

|  |  |
| --- | --- |
| **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. |

Kích thước của ArrayList có thể thay đổi tăng hoặc giảm tùy vào số lượng phần tử trong nó.

Thêm và xóa phần tử ở bất kì vị trí nào trong mảng một cách nhanh chóng, hiệu quả.

Có nhiều phương thức hỗ trợ các thao tác như removeAll-xóa tất cả, searching-tìm kiếm, iterations-duyệt mảng, retainAll-sửa phần tử,…

Một ArrayList có thể chứa nhiều loại đối tượng như Integer, String, Scanner, hay một đối tượng do người dùng tự khai báo.

1. ….
2. **ArrayList** tốt hơn trong việc lưu trữ **và** truy xuất dữ liệu (get). **LinkedList** tốt hơn trong việc thao tác dữ liệu (thêm/ xóa).