Practice OCA :

1. Mảng là các đối tượng vì vậy tham chiếu được thay đổi trong method gọi đến mảng đó được phản ảnh mọi nơi mà mảng được sử dụng.
2. Biến nguyên thủy được truyền giá trị , trong phương thức khi gọi tham số đã truyền một biến nguyên thủy khác với biến trong phương thức vì vậy thành viên thể hiện không thay đổi giá trị
3. Java.lang.Exception là một checked Exception có nghĩa là , phương thức throw ngoại lệ này phải khai báo nó trong mệnh đề throws do đó phương thức của bạn phải khai báo throws exception.
4. Object[] oa = new Object[10] , oa chứa 10 giá trị mặc định của object khai báo
5. Overload phương thức cùng tên khác tham số hoặc thứ tự , số tham số , kiểu trả về không quan trọng
6. RuntimeException ko cần phải handle trong khi Exception phải handle trong try catch or throws
7. Class chỉ khai bảo public khi trùng tên file.
8. Final class ko thể kế thừa , nhưng đc kế thừa class khác , phương thức final ko thể override nhưng có thể kế thừa , biến final ko thể thay đổi giá trị , nếu biến final trống đc khởi tạo bằng constructor và static final trống khởi tạo bằng static block
9. Static block code chạy trước rồi đến instance block rồi đến constructor , trong trg hợp kế thừa thì super luôn chạy trc nhưng ưu tiên nhất là static chạy trước rồi đến instance block code và constructor của super rồi đến subclass.
10. Phương thức static có thể dùng chung quản lý bởi class ko phải instance , biến static the same.
11. Từ khóa native chỉ dùng cho phương thức
12. Quy tắc Switch : Chỉ String , byte , char , short , int và các wrapper Class của nó , có cả enums được sử dụng làm kiểu một biến chuyển đổi , Chuỗi đc cho phép từ Java 7 . Các hằng số case phải được gán biến switch, nếu biến switch là String , biến case là String . Biến Switch phải đủ lớn để chứa tất cả các hằng số case , ví dụ biến char thì không có hằng số case nào vượt quá 65535 vì scope của char là 0 – 65535. Tất cả các biến case phải là constants. Không có hai trong số các biểu thức hằng số case liên quan đến biến switch có thể cùng giá trị. Nhiều nhất một biến default có thể liên kết với biến switch.
13. Mảng là một đối tượng , nhưng mảng của đối tượng không thể là mảng của các đối tượng khác. Object[] obj = new int[10] là sai. Size của mảng phải được khai báo bên phải. Int [] i ,j là I , j đều là mảng int , int I [] , j là I là mảng int còn j là số int.
14. Mảng đa chiều , size đầu tiên phải được chỉ định , các size sau có thể không cần chỉ định vì mảng đa chiều ko cần phải đối xứng. Mảng một chiều độ dài phải được chỉ định.
15. Các biến chứa tham chiếu đến các đối tượng thực tế . Việc chỉ định cho một tham chiếu đến đối tượng thực tế khác chỉ thay đổi giá trị của tham chiếu và không thay đổi giá trị đối tượng thực tế , toString override tại lớp String chỉ in ra chuỗi thực tế.
16. Bắt đầu từ java 7 bạn có thể sử dụng dấu gạch dưới giữa các số để giúp viết số dài dễ hơn ví dụ nếu viết 1000000 có thể viết 1\_000\_000. Lưu ý bạn không thể bắt đầu hoặc kết thúc một giá trị với dấu gạch dưới do đó 100\_ và \_100 là các giá trị không hợp lệ, \_100 là tên biến hợp lệ , bạn có thể sử dụng dấu gạch dưới cho tất cả các loại số bao gồm long , double , binary , float , hex ví dụ : int hex = 0xCAFE\_BASE , float f = 9898\_7878.333\_333f , int bin = 0b111\_0000\_1100\_1100;
17. Biến nào (hoặc phương thức tĩnh) sẽ được sử dụng tùy thuộc vào lớp mà biến được khai báo. Phương thức cá thể nào sẽ được sử dụng phụ thuộc vào lớp thực tế của đối tượng được tham chiếu bởi biến.
18. String trong cùng một lớp cùng 1 package hoặc trong các lớp khác nhau cùng 1 package hoặc các lớp khác nhau trong các package khác nhau biểu thị các tham chiếu đến cùng một đối tượng String , Các chuỗi được tính bằng các biểu thức không đổi được tính toán tại thời điểm biên dịch và sau đó được xử lý nếu chúng là chữ. Chuỗi được tính toán trong thời gian chạy run time , được tạo mới và do đó nó là khác biệt, Kết quả của việc thực hiện intern() một chuỗi là cùng một chuỗi bất kỳ chuỗi kí tự có sẵn nào có cùng nội dung.
19. Thực hiện toán tử luôn thực hiện bên trái trước rồi đến bên phải.
20. String , StringBuilder , StringBuffer đều là các final class , Boolean , Integer , Long , Short là final , Number không phải là final , int , long , double extended Number , System là final
21. Có các trường hoặc phương thức mơ hồ không tự gây ra bất kỳ vấn đề nào nhưng việc tham khảo các trường / phương thức đó theo cách không rõ ràng sẽ gây ra lỗi thời gian biên dịch. Vì vậy, bạn không thể gọi: System.out.println (VALUE); bởi vì nó sẽ mơ hồ (có hai định nghĩa GIÁ TRỊ). Nhưng các dòng sau là hợp lệ: TestClass tc = new TestClass (); System.out.println (((T1) tc) .VALUE); Tuy nhiên, không cần phải sử dụng cast rõ ràng để gọi phương thức m1 (): ((T2) tc) .m1 (); tc.m1 () cũng tốt vì mặc dù m1 () được khai báo trong cả hai giao diện, định nghĩa cho cả hai đều giải quyết rõ ràng chỉ một m1 (), được định nghĩa trong TestClass.
22. Để ghi đè một phương thức trong lớp con, phương thức ghi đè (nghĩa là phương thức trong lớp con) PHẢI CÓ: cùng một kiểu trả về trong trường hợp nguyên thủy (một lớp con được phép cho các lớp, đây còn được gọi là kiểu trả về covariant đối với trường hợp là đối tượng). cùng loại và thứ tự của các tham số. nó chỉ có thể ném ra những ngoại lệ được khai báo trong mệnh đề ném của phương thức của siêu lớp hoặc các ngoại lệ là các lớp con của các ngoại lệ được khai báo. Nó cũng có thể chọn KHÔNG ném bất kỳ ngoại lệ. Tên của các loại tham số không quan trọng. Ví dụ, void methodX (int i) giống như void methodX (int k)
23. Một lớp có thể có một phương thức tên là Main. Mặc dù, vì nó không giống như main, nên nó sẽ không được coi là phương thức chính tiêu chuẩn mà JVM có thể gọi khi chương trình được thực thi.
24. bạn nên nhớ rằng loại kết quả sẽ là int ngay cả khi cả hai toán hạng đều thuộc loại nhỏ hơn một số nguyên Do đó, sau đây sẽ không biên dịch - byte b1 = 1; byte b2 = 2; byte b = b1 + b2; // kết quả là kiểu int, không thể gán trực tiếp cho một byte Bạn phải sử dụng cast: byte b = (byte) (b1 + b2); // OK ngay Tương tự, khi một trong các toán hạng có kiểu dài, float hoặc double và toán hạng khác có kích thước nhỏ hơn, kết quả sẽ lần lượt là dài, float hoặc double.
25. RuntimeException là các ngoại lệ xảy ra bên trong chương trình , CheckedException là các ngoại lệ xảy ra bên ngoài chương trình, trong trường hợp một chương trình gọi đến một phương thức có đoạn code viết sai cố gắng trỏ đến một giá trị null , khi đó sử dụng RuntimeException là hợp lệ vì nó xảy ra bên trong chương trình và không lường trước đc nó cần được xác định trong khi Test và không nên xảy khi chương trình sẵn sàng để triển khai , trong trường hợp một chương trình gọi đến một phương thức truy cập tới một tệp và nó có thể không triển khai khi tệp không tồn tại do đó nó lường trước đc tình huống này và lỗi do một nguồn bên ngoài nên sử dụng CheckedException là hợp lệ.
26. Trình khởi tạo static hoặc cá thể block code chỉ ném ra RuntimeException, nếu bạn cố gắng ném một CheckedException từ một block code khởi tạo static hoặc cá thể ra bên ngoài, mã sẽ không compile. Bất kỳ ngoại lệ nào được ném trong một block static được gói vào ExceptionInitializerError và ném ra ExceptionInitializerError.
27. Đối với package không có tên thì không thể truy cập vào Class nằm trong package đó vì không có cách nào để import , đồng thời các thành phần Static cũng không thể import được.
28. Các biến Static có thể được để lại mà không cần khởi tạo rõ ràng và nó sẽ nhận được các giá trị mặc định , các biến final phải được khởi tạo rõ ràng riêng đối với static final phải được khởi tạo rõ ràng và có thể đặt trong static block code không thể đặt trong hàm khởi tạo vì static có thể đc gọi trước khi có thể hiển của một lớp được tạo nên nó không thể đợi cho đến khi hàm khởi tạo được gọi.
29. Garbage Collection: một đối tượng đủ điều kiện để Garbage Collection khi đối tượng đó không có bất kì một tham chiếu nào , bạn không thể đề xuất JVM gọi trực tiếp Garbage Collection nhưng bạn có thể đề xuất Garbage Collection bằng lệnh System.gc().
30. Đối với giá trị được protected , một lớp khác package extend Class chứa biến protected sẽ chỉ kế thừa biến này mà không thể đọc hoặc sửa đổi giá trị của biến của một thể hiện Class.
31. Ví dụ : boolean a; int b = 10 ; system.out.print(a = b ==10); sẽ in ra true vì giá trị b == 10 được thực hiện trc trả về true rồi gắn vào biến a; kiểu của biến sẽ được khai báo trước rồi được gán giá trị sau.
32. Các biến local phải được khởi tại bằng cách gán giá trị cho nó nếu có sử dụng đến nó nếu không trình biên dịch sẽ báo lỗi.
33. Chỉ cho phép thu hẹp ngầm đối với các biến nguyên thủy byte , char,short , int đối với biến long , double, float thì phải ép kiểu. Giá trị mặc định của long, double, float là một số int.
34. Interfaces , classes và enums đều là “Types” (Kiểu). Java cho phép một lớp triển khai nhiều interfaces. Bằng cách này , Java hỗ trợ đa kế thừa. Mặt khác, “State” (Trạng thái) được biểu diễn bằng các trường instance. Chỉ một lớp mới có thể có các trường thể hiện và do đó , chỉ một lớp mới có thể có trạng thái. (Các trường được xác định trong một interface luôn ở trạng thái static , ngay cả khi bạn không chỉ định từ khóa static một cách rõ ràng. Do đó , một interface không có bất kỳ trạng thái nào.) Vì một lớp chỉ được phép extend từ một lớp , nó chỉ có thể kế thừa một trạng thái. Do đó , Java không hỗ trợ đa kế thừa trạng thái. Trong java 8 cho phép một interface chứa các triển khai ở dạng phương thức mặc định và phương thức static. Tuy nhiên , Java 8 vẫn không cho phép một lớp kế thừa nhiều triển khai của cùng một phương thức từ hai giao diện không liên quan khác nhau. Ví dụ:

interface Movable{  
   public default void move(int dx) { System.out.println("in m");}  
}  
interface Growable{  
   public default void move(int dx) { System.out.println("in g");}  
}  
  
class Animal implements Movable, Growable{  
}

(Animal thực hiện Movable , Growable trình biên dịch sẽ phàn nàn rằng class Animal kế thừa các giá trị mặc định không liên quan cho move(int) từ các kiểu Movable và Growable. Để biên dịch nó bạn phải cung cấp một cách thực thi phương thức move(int) một cách rõ ràng trong lớp Animal.)

1. Khi khởi tạo một thể hiện tới một đối tượng lớp con kế thừa một lớp cha thì phải đảm bảo lớp cha được khởi tạo trước nên constructor của lớp con sẽ mặc định gọi tới super() của lớp cha ở dòng đầu tiên , trong trường hợp lớp cha ko có hàm constructor no-arg thì super() sẽ fail và cần phải gọi constructor của lớp cha ngay trong dòng đầu tiên của constructor của lớp con , có thể gọi tới constructor của lớp con ngay dòng đầu tiên nhưng phải đảm bảo sau đó constructor của lớp cha phải được xảy ra trước.
2. Tính đa hình làm cho code năng động hơn vì nó quyết định phương thức nào sẽ được gọi trong run time phụ thuộc vào lớp thức tế của đối tượng mặt khác nó làm cho code tái sử dụng hơn.
3. Ko thể ghi đè phương thức private và phương thức static của lớp cha , lớp con thỏa mái có phương thức static và phương thức thành phần cùng tên với phương thức private và static của lớp cha.
4. Lưu ý rằng câu lệnh switch so sánh đối tượng String trong biểu thức của nó với các biểu thức được liên kết với mỗi nhãn trường hợp như thể nó đang sử dụng phương thức String.equals; do đó, việc so sánh các đối tượng String trong các câu lệnh switch là phân biệt chữ hoa chữ thường. Trình biên dịch Java thường tạo ra bytecode hiệu quả hơn từ các câu lệnh switch sử dụng các đối tượng String hơn là từ các câu lệnh if-then-else được xâu chuỗi.
5. Đây là quy tắc được đưa ra trong JLS liên quan đến việc gán các giá trị không đổi cho các biến nguyên thủy mà không cần ép kiểu rõ ràng:

Một chuyển đổi nguyên thủy thu hẹp có thể được sử dụng nếu tất cả các điều kiện sau được thỏa mãn:

Biểu thức là một biểu thức hằng thời gian biên dịch kiểu byte, char, short hoặc int.

Kiểu của biến là byte, short hoặc char.

Giá trị của biểu thức (được biết tại thời điểm biên dịch, vì nó là một biểu thức hằng) có thể biểu diễn trong kiểu của biến.

Lưu ý rằng chuyển đổi thu hẹp hoàn toàn (tức là chuyển đổi không có diễn viên rõ ràng) không áp dụng cho float, long hoặc double.

Ví dụ, char ch = 30L; sẽ không biên dịch được mặc dù 30 là đủ nhỏ để vừa với một ký tự.

1. Cố gắng ghi đè một phương thức tĩnh bằng một phương thức không tĩnh (và ngược lại) trong một lớp sẽ dẫn đến lỗi biên dịch. Ngay cả trong trường hợp giao diện, một giao diện con không thể ghi đè một phương thức mặc định bằng một phương thức tĩnh. Tuy nhiên, bạn có thể có một phương thức mặc định trong một giao diện con có cùng chữ ký với một phương thức tĩnh của siêu giao diện bởi vì một phương thức tĩnh của một giao diện chỉ có thể được gọi bằng tên của giao diện đó.