**1. Định nghĩa Class, Object**

1. **Câu 1:** Trong Java, một đối tượng là gì?
   * A. Một biến chứa các giá trị
   * **B. Một instance của một class** ✅
   * C. Một hàm xử lý dữ liệu
   * D. Một kiểu dữ liệu đặc biệt

**Giải thích:** Đối tượng là một instance (phiên bản) của một class. Mỗi đối tượng có thể có các trạng thái và hành vi riêng biệt.

1. **Câu 2:** Để tạo một đối tượng trong Java, ta sử dụng cú pháp nào?
   * **A. new Class()** ✅
   * B. Class.create()
   * C. new Class;
   * D. create Class()

**Giải thích:** Để tạo một đối tượng trong Java, cú pháp đúng là new Class(). Đây là cú pháp chuẩn để khởi tạo một đối tượng từ một class.

1. **Câu 3:** Class trong Java là gì?
   * A. Một biến
   * **B. Một khuôn mẫu dùng để tạo các đối tượng, chứa các thuộc tính (fields) và phương thức (methods) mà đối tượng sẽ có** ✅
   * C. Một hàm
   * D. Một kiểu dữ liệu đơn giản

**Giải thích:** Class là một khuôn mẫu dùng để tạo các đối tượng, chứa các thuộc tính (fields) và phương thức (methods) mà đối tượng sẽ có.

1. **Câu 4:** Câu lệnh nào sau đây tạo một đối tượng từ một lớp Car trong Java?
   * **A. Car c = new Car();** ✅
   * B. Car c = Car();
   * C. new Car c();
   * D. Car c = Car.new();

**Giải thích:** Để tạo một đối tượng Car, ta phải sử dụng cú pháp Car c = new Car();.

**2. 4 Tính chất Lập trình Hướng Đối Tượng**

1. **Câu 5:** Tính chất nào trong lập trình hướng đối tượng biểu thị cho khả năng kế thừa?
   * A. Encapsulation
   * B. Polymorphism
   * **C. Inheritance** ✅
   * D. Abstraction

**Giải thích:** Kế thừa (Inheritance) cho phép một class kế thừa thuộc tính và phương thức từ một class khác, tạo thành mối quan hệ cha-con.

1. **Câu 6:** Tính chất nào trong lập trình hướng đối tượng giúp ẩn các chi tiết thực thi bên trong một đối tượng và chỉ cung cấp những phương thức cần thiết để người dùng tương tác với đối tượng đó.?
   * **A. Encapsulation** ✅
   * B. Polymorphism
   * C. Inheritance
   * D. Abstraction

**Giải thích:** Encapsulation (Đóng gói) giúp ẩn các chi tiết thực thi bên trong một đối tượng và chỉ cung cấp những phương thức cần thiết để người dùng tương tác với đối tượng đó.

1. **Câu 7:** Tính chất nào trong lập trình hướng đối tượng giúp một đối tượng có thể có nhiều hành vi khác nhau dựa trên kiểu đối tượng của nó?
   * A. Encapsulation
   * **B. Polymorphism** ✅
   * C. Inheritance
   * D. Abstraction

**Giải thích:** Polymorphism (Đa hình) cho phép một đối tượng có thể thực hiện các hành vi khác nhau tùy thuộc vào kiểu dữ liệu thực tế của đối tượng đó.

1. **Câu 8:** Tính chất nào trong lập trình hướng đối tượng cho phép ta sử dụng một giao diện các chức năng mà không cần phải biết các chi tiết bên trong của nó?
   * A. Encapsulation
   * B. Polymorphism
   * **C. Abstraction** ✅
   * D. Inheritance

**Giải thích:** Abstraction (Trừu tượng) giúp làm việc với đối tượng thông qua các phương thức hoặc thuộc tính mà không cần biết chi tiết thực thi bên trong.

**3. Overriding, Overloading**

1. **Câu 9:** Overloading là gì trong Java?
   * **A. Việc định nghĩa nhiều phương thức với cùng tên nhưng có tham số khác nhau** ✅
   * B. Việc định nghĩa nhiều phương thức với cùng tên và cùng tham số
   * C. Việc thay đổi logic của phương thức trong class con
   * D. Việc gọi phương thức trong phương thức khác

**Giải thích:** Overloading là khi ta tạo ra nhiều phương thức với cùng tên nhưng có số lượng hoặc kiểu tham số khác nhau trong một class.

1. **Câu 10:** Overriding là gì trong Java?
   * A. Việc định nghĩa nhiều phương thức với cùng tên nhưng có tham số khác nhau
   * **B. Việc thay đổi logic của phương thức trong class con** ✅
   * C. Việc thay đổi tên phương thức
   * D. Việc gọi phương thức trong phương thức khác

**Giải thích:** Overriding là việc định nghĩa lại phương thức đã được khai báo trong class cha, nhằm thay đổi hoặc mở rộng logic của phương thức đó.

**4. Access Modifiers**

1. **Câu 11:** Modifier nào trong Java cho phép truy cập phương thức từ bất kỳ đâu trong chương trình?
   * **A. public** ✅
   * B. private
   * C. protected
   * D. default

**Giải thích:** public cho phép truy cập từ bất kỳ đâu trong chương trình.

1. **Câu 12:** Modifier nào trong Java chỉ cho phép truy cập phương thức trong cùng một class?
   * A. public
   * **B. private** ✅
   * C. protected
   * D. default

**Giải thích:** private chỉ cho phép truy cập trong cùng một class.

1. **Câu 13:** Modifier nào trong Java cho phép truy cập phương thức trong cùng package hoặc class con?
   * A. public
   * **B. protected** ✅
   * C. default
   * D. private

**Giải thích:** protected cho phép truy cập từ cùng package hoặc class con (lớp kế thừa).

1. **Câu 14:** Modifier nào trong Java cho phép truy cập phương thức trong cùng package?
   * A. public
   * B. private
   * **C. default** ✅
   * D. protected

**Giải thích:** default (không có modifier) cho phép truy cập trong cùng một package.

**5. Interface, Abstract**

1. **Câu 15:** Một class có thể implement bao nhiêu interface trong Java?
   * **A. Nhiều interface** ✅
   * B. Chỉ một interface
   * C. Chỉ có thể kế thừa class cha
   * D. Không thể kế thừa interface

**Giải thích:** Java cho phép một class implement nhiều interface.

1. **Câu 16:** Một class abstract trong Java có thể có bao nhiêu phương thức abstract?
   * A. Không thể có phương thức abstract
   * B. Chỉ có thể có một phương thức abstract
   * **C. Có thể có nhiều phương thức abstract** ✅
   * D. Không thể có phương thức bình thường

**Giải thích:** Một class abstract có thể chứa nhiều phương thức abstract.

1. **Câu 17:** Một interface trong Java có thể có bao nhiêu phương thức abstract?
   * A. Không thể có phương thức abstract
   * **B. Có thể có nhiều phương thức abstract** ✅
   * C. Chỉ có thể có một phương thức abstract
   * D. Không thể có phương thức non-abstract

**Giải thích:** Một interface có thể chứa nhiều phương thức abstract, và tất cả các phương thức trong interface đều mặc định là abstract.

1. **Câu 18:** Nếu một lớp implement một interface, thì lớp đó có bắt buộc phải triển khai tất cả các phương thức của interface không?
   * A. Không, lớp có thể triển khai một số phương thức
   * **B. Có, lớp phải triển khai tất cả phương thức của interface** ✅
   * C. Không cần phải triển khai phương thức nào
   * D. Lớp không thể kế thừa interface

**Giải thích:** Khi implement một interface, lớp con bắt buộc phải triển khai tất cả các phương thức abstract của interface.

**6. Functional Interface**

1. **Câu 19:** Functional Interface trong Java là gì?
   * A. Interface chỉ có một phương thức trừu tượng
   * **B. Interface chỉ có một phương thức trừu tượng, nhưng có thể có nhiều phương thức default hoặc static** ✅
   * C. Interface có thể chứa nhiều phương thức abstract
   * D. Interface có thể không có phương thức abstract

**Giải thích:** Functional Interface là một interface chỉ có một phương thức abstract, có thể có nhiều phương thức default hoặc static.

1. **Câu 20:** Câu lệnh nào trong Java sử dụng để định nghĩa một functional interface?
   * A. @Override
   * **B. @FunctionalInterface** ✅
   * C. @Interface
   * D. @Abstract

**Giải thích:** @FunctionalInterface là annotation dùng để chỉ định rằng một interface là functional interface.

**7. Try, Catch, Finally, Throw, Throws, Checked Exception, Unchecked Exception, Error**

1. **Câu 21:** Câu lệnh nào trong Java được sử dụng để ném một exception?
   * **A. throw** ✅
   * B. throws
   * C. try
   * D. catch

**Giải thích:** throw được dùng để ném một exception trong Java.

1. **Câu 22:** Exception nào sau đây là loại checked exception?
   * A. NullPointerException
   * **B. IOException** ✅
   * C. ArithmeticException
   * D. IndexOutOfBoundsException

**Giải thích:** IOException là một checked exception, yêu cầu phải xử lý hoặc khai báo.

1. **Câu 23:** Unchecked exception trong Java là gì?
   * A. Các exception phải được xử lý thông qua try-catch
   * **B. Các exception không được kiểm tra tại thời điểm biên dịch** ✅
   * C. Các exception không yêu cầu phải xử lý tại thời điểm biên dịch
   * D. Các exception không thể bị bắt

**Giải thích:** Unchecked exceptions là các exception không được kiểm tra tại thời điểm biên dịch, chẳng hạn như RuntimeException.

1. **Câu 24:** Mục đích của khối finally trong Java là gì?
   * A. Để xử lý exception
   * B. Để kiểm tra điều kiện
   * **C. Để thực thi mã bất kể có exception hay không** ✅
   * D. Để ném exception

**Giải thích:** Khối finally luôn được thực thi sau khi khối try và catch, dùng để thực hiện các tác vụ như đóng tài nguyên.

1. **Câu 25:** Exception nào dưới đây là loại unchecked exception?
   * A. IOException
   * **B. NullPointerException** ✅
   * C. SQLException
   * D. ClassNotFoundException

**Giải thích:** NullPointerException là một unchecked exception, thuộc lớp RuntimeException.

1. **Câu 26:** Nếu bạn muốn ném một exception mà không xử lý nó, bạn sẽ dùng từ khóa nào?
   * **A. throw** ✅
   * B. throws
   * C. try
   * D. catch

**Giải thích:** Từ khóa throw dùng để ném một exception trong Java.

1. **Câu 27:** Khi một exception được ném và không được xử lý, chương trình sẽ:
   * A. Dừng và tiếp tục
   * **B. Dừng và hiển thị thông báo lỗi** ✅
   * C. Tiếp tục thực hiện
   * D. Gọi phương thức khác

**Giải thích:** Nếu một exception không được xử lý, chương trình sẽ dừng lại và hiển thị thông báo lỗi.

1. **Câu 28:** Từ khóa throws được sử dụng để làm gì?
   * A. Ném exception
   * **B. Khai báo exception có thể được ném bởi phương thức** ✅
   * C. Xử lý exception
   * D. Kiểm tra exception

**Giải thích:** Từ khóa throws được dùng để khai báo rằng một phương thức có thể ném một exception, và exception đó phải được xử lý.

**8. Collection bao gồm List, Map, Set**

1. **Câu 29:** Collection trong Java là gì?
   * **A. Collection là một tập hợp các đối tượng, cho phép lưu trữ, truy xuất và xử lý dữ liệu** ✅
   * B. Một kiểu dữ liệu đặc biệt
   * C. Một lớp chứa các phương thức
   * D. Một mảng động

**Giải thích:** Collection là một tập hợp các đối tượng, cho phép lưu trữ, truy xuất và xử lý dữ liệu.

1. **Câu 30:** Loại Collection nào trong Java không cho phép lưu trữ các phần tử trùng lặp?
   * A. List
   * **B. Set** ✅
   * C. Map
   * D. ArrayList

**Giải thích:** Set không cho phép các phần tử trùng lặp, khác với List cho phép lưu trữ các phần tử trùng lặp.

1. **Câu 31:** Loại Collection nào trong Java có thể chứa các phần tử trùng lặp và theo thứ tự chèn?
   * **A. List** ✅
   * B. Set
   * C. Map
   * D. HashSet

**Giải thích:** List cho phép lưu trữ các phần tử trùng lặp và bảo vệ thứ tự chèn.

1. **Câu 32:** Map trong Java lưu trữ các đối tượng theo cặp key-value. Câu lệnh nào sau đây là đúng để tạo một Map?
   * A. Map m = new HashMap();
   * **B. Map<String, Integer> map = new HashMap<>();** ✅
   * C. Map map = new TreeSet();
   * D. Map map = new List();

**Giải thích:** Map<String, Integer> map = new HashMap<>(); là cách khởi tạo một Map với kiểu khóa là String và giá trị là Integer.

**9. ArrayList, HashMap, Hashtable, HashSet**

1. **Câu 33:** Mảng trong Java có kích thước thay đổi được không, không phải ArrayLisst trong collection mà là array thông thường?
   * A. Có thể thay đổi kích thước khi cần
   * **B. Không thể thay đổi kích thước** ✅
   * C. Kích thước thay đổi tự động
   * D. Kích thước thay đổi nhưng phải khai báo lại

**Giải thích:** Mảng trong Java có kích thước cố định khi khởi tạo và không thể thay đổi kích thước.

1. **Câu 34:** HashMap trong Java có tính chất nào sau đây?
   * A. Lưu trữ theo thứ tự chèn
   * **B. Không cho phép các khóa(key) trùng lặp** ✅
   * C. Lưu trữ theo thứ tự tăng dần
   * D. Không lưu trữ giá trị null

**Giải thích:** HashMap không cho phép các khóa trùng lặp(key), nhưng giá trị có thể là null.

1. **Câu 35:** Hashtable trong Java khác với HashMap ở điểm nào?
   * A. Hashtable là đồng bộ (synchronized), HashMap không phải
   * **B. Tất cả đều đúng** ✅
   * C. Hashtable không cho phép null keys và values, còn HashMap thì có thể
   * D. Không có sự khác biệt

**Giải thích:** Hashtable là đồng bộ (synchronized) và không cho phép giá trị null, trong khi HashMap không phải đồng bộ và có thể chứa giá trị null.

1. **Câu 36:** HashSet trong Java có tính chất gì?
   * A. Lưu trữ các phần tử theo thứ tự
   * **B. Không cho phép các phần tử trùng lặp** ✅
   * C. Lưu trữ theo thứ tự chèn
   * D. Cho phép các phần tử null

**Giải thích:** HashSet không cho phép các phần tử trùng lặp và không bảo đảm thứ tự của các phần tử.

**10. Thread, Multiple Thread**

1. **Câu 37:** Thread trong Java là gì?
   * **A. Một đơn vị thực thi trong chương trình** ✅
   * B. Một kiểu dữ liệu
   * C. Một đối tượng
   * D. Một hàm

**Giải thích:** Thread là một đơn vị thực thi trong chương trình. Mỗi thread có thể thực thi một đoạn mã riêng biệt.

1. **Câu 38:** Câu lệnh nào sau đây dùng để tạo một thread trong Java?
   * **A. Thread t = new Thread();** ✅
   * B. Thread t = new Runnable();
   * C. Thread t = Thread();
   * D. new Runnable();

**Giải thích:** Để tạo một thread trong Java, ta sử dụng Thread t = new Thread(); để tạo một đối tượng thread.

1. **Câu 39:** Nếu bạn muốn một thread thực thi một đoạn mã cụ thể, bạn cần làm gì?
   * **A. Ghi đè phương thức run() trong class kế thừa Thread** ✅
   * B. Sử dụng phương thức start() để thực thi mã
   * C. Gọi phương thức run() trực tiếp
   * D. Cả A và B đều đúng

**Giải thích:** Để thực thi mã trong thread, bạn cần ghi đè phương thức run() trong class kế thừa Thread, sau đó gọi start().

1. **Câu 40:** Câu lệnh nào trong Java khởi tạo và chạy một thread?
   * A. start()
   * **B. Thread t = new Thread(); t.start();** ✅
   * C. Thread.run()
   * D. thread();

**Giải thích:** Để khởi tạo và bắt đầu một thread, ta sử dụng Thread t = new Thread(); t.start();.

1. **Câu 41:** Câu lệnh nào trong Java giúp thread khác thực thi sau một khoảng thời gian trong Java?
   * A. wait()
   * **B. sleep()** ✅
   * C. notify()
   * D. join()

**Giải thích:** sleep() giúp thread dừng lại trong một khoảng thời gian nhất định trước khi tiếp tục.

1. **Câu 42:** Câu lệnh nào trong Java cho phép một thread chờ đợi cho đến khi thread khác hoàn thành?
   * A. wait()
   * **B. join()** ✅
   * C. sleep()
   * D. notify()

**Giải thích:** join() giúp một thread chờ đợi cho đến khi thread khác hoàn thành.

1. **Câu 43:** Làm sao để tạo nhiều thread trong Java?
   * **A. Tạo nhiều đối tượng của class Thread và gọi start() cho từng đối tượng** ✅
   * B. Tạo một đối tượng duy nhất và gọi start() một lần
   * C. Tạo một class duy nhất và sử dụng phương thức run()
   * D. Không thể tạo nhiều thread

**Giải thích:** Để tạo nhiều thread, bạn cần tạo nhiều đối tượng Thread và gọi start() cho mỗi đối tượng.

1. **Câu 44:** Khi một thread kết thúc, trạng thái của nó là gì?
   * A. Ready
   * B. Running
   * **C. Terminated** ✅
   * D. Sleeping

**Giải thích:** Khi thread kết thúc, trạng thái của nó là Terminated.

1. **Câu 45:** Trong môi trường đa luồng, khi nào xảy ra hiện tượng race condition?
   * A. Khi nhiều thread truy cập tài nguyên đồng thời mà không có sự đồng bộ
   * **B. Khi các thread không đồng bộ khi truy cập tài nguyên chung** ✅
   * C. Khi thread không có đủ tài nguyên
   * D. Khi thread kết thúc sớm

**Giải thích:** Race condition xảy ra khi nhiều thread truy cập và thay đổi tài nguyên chung mà không có cơ chế đồng bộ.

1. **Câu 46:** Để đảm bảo rằng các thread trong Java không can thiệp vào nhau khi truy cập tài nguyên chung, ta cần sử dụng gì?
   * **A. Synchronization** ✅
   * B. Thread.sleep()
   * C. Thread.join()
   * D. Wait/Notify

**Giải thích:** Để tránh các thread can thiệp vào nhau khi truy cập tài nguyên chung, ta cần sử dụng synchronized.

1. **Câu 47:** Trong Java, phương thức nào dùng để chờ một thread hoàn thành trước khi thực thi tiếp?
   * A. wait()
   * **B. join()** ✅
   * C. sleep()
   * D. notify()

**Giải thích:** join() giúp thread hiện tại chờ cho đến khi thread khác hoàn thành trước khi tiếp tục.