Đề thi Đánh giá Năng lực Java & OOP

Thời gian: 15 phút
Số lượng câu hỏi: 5 câu

Phần câu hỏi

1. Trong Lập trình hướng đối tượng (OOP), cơ chế nào cho phép ẩn chi tiết triển khai nội bộ của đối tượng và chỉ lộ ra các phương thức cần thiết để tương tác với bên ngoài?

   (a) Tính kế thừa (Inheritance)

   (b) Tính đa hình (Polymorphism)

   (c) Tính đóng gói (Encapsulation)

   (d) Tính trừu tượng (Abstraction)

2. Cho đoạn mã sau. Điều gì sẽ xảy ra khi biên dịch?

```java
class Animal {
    public void makeSound() { System.out.println("Sound"
        ); }
}
class Dog extends Animal {
    @Override
    public void makeSound() { System.out.println("Bark")
        ; }
}
```

   (a) Lỗi biên dịch vì thiếu phương thức `main`.

   (b) Lớp `Dog` kế thừa lớp `Animal` và ghi đè (override) phương thức `makeSound`.

   (c) Lỗi biên dịch vì phương thức trong lớp cha không phải là `abstract`.

   (d) Lớp `Dog` nạp chồng (overload) phương thức `makeSound`.

3. Để đảm bảo nguyên lý đóng gói (Encapsulation) chặt chẽ nhất trong Java, bạn nên sử dụng *access modifier* nào cho các thuộc tính (fields) của lớp?

   (a) `public`

   (b) `protected`

   (c) `default` (không khai báo gì)

   (d) `private`

4. Hãy xác định output của chương trình sau:

```java
class A {
    void print() { System.out.print("A"); }
}
class B extends A {
    void print() { System.out.print("B"); }
}
public class Main {
    public static void main(String[] args) {
        A obj = new B();
        obj.print();
    }
}
```

(a) In ra: A

(b) In ra: B

(c) Lỗi biên dịch tại dòng `A obj = new B();`

(d) Lỗi Runtime: `ClassCastException`

5. Điều gì xảy ra khi chạy đoạn mã dưới đây?

```java
class Parent {
    static void show() { System.out.println("Parent
        Static"); }
    void display() { System.out.println("Parent Instance
        "); }
}
class Child extends Parent {
    static void show() { System.out.println("Child
        Static"); }
    @Override
    void display() { System.out.println("Child Instance"
        ); }
}
public class Test {
    public static void main(String[] args) {
        Parent p = new Child();
        p.show();
        p.display();
    }
}
```

(a) Parent Static / Parent Instance

(b) Child Static / Child Instance

(c) Parent Static / Child Instance

(d) Child Static / Parent Instance

Câu hỏi Tự luận (60 phút)

Chủ đề: Thiết kế hệ thống quản lý nhân sự sử dụng Java và OOP.

Đề bài:

Một công ty phần mềm cần xây dựng chương trình quản lý tiền lương cho hai loại nhân viên: **Nhân viên Full-time** và **Nhân viên Part-time**.

Hãy thiết kế và cài đặt chương trình bằng ngôn ngữ Java thỏa mãn các yêu cầu sau:

1. **Thiết kế lớp cha (Base Class):** Tạo một lớp trừu tượng (abstract class) hoặc Interface tên là `NhanVien` gồm:
   - Các thuộc tính chung: mã nhân viên, tên nhân viên.
   - Phương thức trừu tượng `tinhLuong()` trả về số thực (double).
2. **Thiết kế lớp con (Subclasses):**
   - Lớp `FullTime`: Kế thừa từ `NhanVien`. Lương được tính bằng công thức:
     $Luong = LuongCoBan \times HeSoLuong$.
   - Lớp `PartTime`: Kế thừa từ `NhanVien`. Lương được tính bằng công thức:
     $Luong = SoGioLamViec \times LuongMoiGio$.
3. **Yêu cầu kỹ thuật bắt buộc:**
   - **Tính đóng gói (Encapsulation):** Tất cả các thuộc tính của lớp phải để ở chế độ `private`. Dữ liệu chỉ được truy xuất hoặc thay đổi thông qua các phương thức Getter/Setter hoặc Constructor.
   - **Tính đa hình (Polymorphism):** Trong hàm `main`, tạo một danh sách (List hoặc Array) chứa lẫn lộn cả nhân viên Full-time và Part-time. Sử dụng vòng lặp để in ra tên và lương của từng người mà không cần kiểm tra kiểu dữ liệu cụ thể (không dùng `instanceof`).

Yêu cầu trả lời:

1. Viết mã nguồn Java hoàn chỉnh cho bài toán trên.
2. Viết một đoạn văn ngắn (3-5 dòng) giải thích rõ: Bạn đã áp dụng **Tính đa hình** ở dòng code nào và lợi ích của nó trong bài toán này là gì?