隨堂練習卷　答案（自學延伸）

建議作答 50 分鐘 · 總分 118

本章為自學延伸、非課程考試範圍。

§ 1　一、選擇題（每題 6 分，共 30 分）

1. **(B)** — 繼承表達 is-a（子類別「是一種」父類別）；has-a 是組合。 2. **(C)** — `extends` 宣告繼承；`implements` 用於介面。 3. **(C)** — `protected`：自己、子類別與同套件可見（介於 public 與 private）。 4. **(B)** — 同名、同參數列、同回傳型別的重新定義叫覆寫（overriding）；參數列不同才是多載。 5. **(B)** — 建構子父先子後：每個建構子第一行先跑 `super()`，故 `new C()` 印 A→B→C。

§ 2　二、概念追蹤（共 16 分）

6. `Dog extends Animal`，`getName()` 是父類別的 `public` 方法，**繼承下來子類別自動擁有**，所以 `new Dog("旺財")` 能呼叫 `getName()`。`name` 雖是 `private`（子類別不能直接存取），但透過繼承來的 public `getName()` 仍可讀到，回傳建構子裡用 `super(name)` 設好的 `旺財`。 7. 輸出：

```
（某種聲音） → 汪汪
```

`Dog` 覆寫了 `speak()`，但在覆寫版本裡用 `super.speak()` 先呼叫**被覆寫掉的父版本**（回「（某種聲音）」），再接上「→ 汪汪」。`super.方法()` 的作用就是在子類別中仍能取用父類別原本的實作。

§ 3　三、改錯（每題 6 分，共 18 分）

8. 方法名大小寫錯：父類別是 `speak`（小寫 s），子類別寫成 `Speak`（大寫 S）。這是**不同名稱**，變成新方法、沒有覆寫到，`@Override` 其實會編譯報錯。改：`@Override public String speak() { return "汪汪"; }`（小寫，與父方法同簽名）。 9. 父類別 `Animal` 只有帶參數建構子 `Animal(String n)`、沒有無引數版；子類別 `Dog()` 沒寫 `super(...)` 時 Java 自動補

`super()`，但父類別沒有無引數建構子 → 編譯錯。改：子類別明確呼叫 `Dog(){ super("預設名"); }`（或讓 `Dog` 收參數後 `super(n)`）。 10. `Object.equals` 的參數型別是 `Object`；寫成 `equals(Point p)` 是**多載**而非覆寫，`@Override` 會報錯。改：

```
@Override public boolean equals(Object o) {
    if (!(o instanceof Point)) return false;
    Point p = (Point) o;
    return x == p.x;
}
```

§ 4 四、程式設計（共 54 分）

11.（18 分）

```
public class Vehicle {
    protected String name;
    public Vehicle(String name) { this.name = name; }
    public String move() { return name + " 移動中"; }
    public static void main(String[] args) {
        Vehicle v = new Vehicle("交通工具");
        Car c = new Car("轎車");
        System.out.println(v.move());   // 交通工具 移動中
        System.out.println(c.move());   // 轎車 在路上行駛
    }
}
class Car extends Vehicle {
    public Car(String name) { super(name); }
    @Override public String move() { return name + " 在路上行駛"; }
}
```

（評分：`extends` + `super(name)` 共 8 分、覆寫 `move()` 加 `@Override` 6 分、main 建物件與輸出 4 分。） 12.（18 分）

```java
public class SavingsAccount extends Account {
    private double interestRate;
    public SavingsAccount(double balance, double rate) {
        super(balance);                 // 第一行：先初始化父部分
        this.interestRate = rate;
    }
    @Override public String toString() {
        return super.toString() + "，利率：" + interestRate;
    }
    public static void main(String[] args) {
        SavingsAccount s = new SavingsAccount(1000, 0.02);
        System.out.println(s);          // 餘額：1000.0，利率：0.02
    }
}
class Account {
    protected double balance;
    public Account(double balance) { this.balance = balance; }
    @Override public String toString() { return "餘額：" + balance; }
}
```

（評分：三層 `extends` + `super(balance)` 第一行 8 分、覆寫 `toString` 用 `super.toString()` 6 分、建物件與輸出 4 分。） 13.（18 分）

```java
public class Point {
    private int x, y;
    public Point(int x, int y) { this.x = x; this.y = y; }
    @Override public String toString() { return "(" + x + ", " + y + ")"; }
    @Override public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Point)) return false;
        Point p = (Point) o;
        return x == p.x && y == p.y;
    }
    public static void main(String[] args) {
        Point p1 = new Point(3, 5), p2 = new Point(3, 5);
        System.out.println("== : " + (p1 == p2));        // false
        System.out.println("equals : " + p1.equals(p2)); // true
    }
}
```

（評分：覆寫 `toString` 4 分、覆寫 `equals` 含 `instanceof` 與向下轉型 10 分、main 輸出 `==` ／`equals` 對比 4 分。關鍵：`equals` 參數須為 `Object`、先 `instanceof` 再轉型。）