ChatGPT

SCJP-Level Master Notes: Java OOPs Hidden Concepts & Traps

1. Inheritance (Constructor Flow, Static vs Instance Block)

Key Concepts:

- Static blocks are executed **only once per class** at the time of class loading.
- Instance blocks execute **every time an object is created**, **before the constructor**.
- Constructor chaining starts from **superclass to subclass**.

Execution Order (object creation):

1. Static block (only once)
2. Instance block and variables (super → sub)
3. Constructor (super → sub)

Hidden Traps:

- If a subclass constructor doesn't explicitly call `super()`, Java automatically inserts `super()`.
- You can’t call `this()` and `super()` in the same constructor.

SCJP Example:

```
class A {
    static { System.out.println("A static"); }
    { System.out.println("A instance"); }
    A() { System.out.println("A constructor"); }
}
class B extends A {
    static { System.out.println("B static"); }
    { System.out.println("B instance"); }
    B() { System.out.println("B constructor"); }
}
public class Test {
    public static void main(String[] args) {
        new B();
        new B();
    }
}
```

Output:

```
A static
B static
A instance
A constructor
```

```
B instance
B constructor
A instance
A constructor
B instance
B constructor
```

---

2. Method Overriding vs Method Hiding

Key Concepts:

- **Overriding** applies to instance methods. Resolution is at **runtime**.
- **Hiding** applies to static methods. Resolution is at **compile time**.

Traps:

- A subclass can't override a method with **narrower access modifier**.
- Overriding method can throw **fewer or unchecked exceptions**, not broader checked ones.

SCJP Example:

```
class Parent {
    static void show() { System.out.println("Parent static"); }
    void display() { System.out.println("Parent display"); }
}
class Child extends Parent {
    static void show() { System.out.println("Child static"); }
    void display() { System.out.println("Child display"); }
}
public class Test {
    public static void main(String[] args) {
        Parent obj = new Child();
        obj.show();    // Parent static (compile-time)
        obj.display(); // Child display (runtime)
    }
}
```

---

3. Final Class, Method, Variable

Key Concepts:

- final class = cannot be inherited.
- final method = cannot be overridden.
- final variable = constant; must be initialized once.

Traps:

- A final reference can't be reassigned, but object state can change.
- Blank final variables must be initialized in all constructors.

SCJP Example:

```java
class A {
    final int x;
    A() { x = 10; } // mandatory initialization
}
```

---

4. Abstract Class vs Interface

Abstract Class:

- Can have constructors, instance variables, both abstract and non-abstract methods.

Interface (Java 8+):

- Can have static, default, and abstract methods.
- Variables are implicitly `public static final`.

Traps:

- Cannot instantiate abstract classes.
- Cannot mark interface methods as `protected`.

SCJP Example:

```java
interface I {
    void test();
}
abstract class A implements I {
    public void test() { System.out.println("A"); }
}
```

---

5. Method Overloading Priority

Resolution Order:

1. Exact match
2. Widening
3. Autoboxing
4. Varargs

Traps:

- Varargs is the last choice.
- Autoboxing can fail if widening is possible.

SCJP Example:

```
void m(int i) { System.out.println("int"); }
void m(long l) { System.out.println("long"); }
void m(Integer i) { System.out.println("Integer"); }
void m(int... i) { System.out.println("varargs"); }

m(10); // int
```

---

6. Boxing and Unboxing

Key Concepts:

- Primitive ↔ Wrapper conversion happens automatically.

Traps:

- `Integer a = 100, b = 100; a == b` → true (cached)
- `Integer a = 200, b = 200; a == b` → false (no caching)

SCJP Example:

```
Integer a = 100;
Integer b = 100;
System.out.println(a == b); // true

Integer c = 200;
Integer d = 200;
System.out.println(c == d); // false
```

---

7. Typecasting: Upcasting vs Downcasting

Key Concepts:

- Upcasting: Sub → Super (safe)
- Downcasting: Super → Sub (risky)

Trap:

- Always use `instanceof` before downcasting to avoid runtime error.

SCJP Example:

```
A a = new B(); // upcasting OK
B b = (B) a;   // downcasting OK

A x = new A();
B y = (B) x;   // Runtime error
```

---

8. `instanceof` Keyword

Key Concepts:

- Checks actual type at runtime.
- Always returns false for null.

SCJP Example:

```
Object obj = null;
System.out.println(obj instanceof String); // false
```

---

9. Shadowing vs Hiding

Variable Shadowing:

- Subclass variable hides superclass variable.

Static Method Hiding:

- Static method in subclass hides superclass static method.

Trap:

- Variables and static methods resolved by reference type.

SCJP Example:

```
class A {
    static void show() { System.out.println("A static"); }
    int x = 10;
}
class B extends A {
    static void show() { System.out.println("B static"); }
    int x = 20;
}
A obj = new B();
```

```
System.out.println(obj.x); // 10
obj.show(); // A static
```

---

10. Object Class Methods (`equals()`, `hashCode()`)

Key Concepts:

- Override `equals()` and `hashCode()` together.
- `==` checks reference; `.equals()` checks value.

Trap:

- If only `equals()` overridden, HashSet may store duplicates.

SCJP Example:

```
class Emp {
    int id;
    Emp(int id) { this.id = id; }
    public boolean equals(Object o) {
        return this.id == ((Emp)o).id;
    }
    public int hashCode() {
        return id;
    }
}
```

---

11. Polymorphism Runtime Traps

Key Concepts:

- Only instance methods are polymorphic.

Traps:

- Variables, static methods, private methods are not overridden.

SCJP Example:

```
class A {
    int x = 10;
    static void show() { System.out.println("A"); }
    void print() { System.out.println("A print"); }
}
class B extends A {
    int x = 20;
    static void show() { System.out.println("B"); }
    void print() { System.out.println("B print"); }
}
A obj = new B();
System.out.println(obj.x); // 10
obj.show(); // A
obj.print(); // B print
```

---

12. Constructor & Block Order in Multi-level Inheritance

Execution Flow:

1. Static blocks (super → sub, once)
2. Instance blocks/fields (super → sub)
3. Constructors (super → sub)

SCJP Example:

```
class G {
    { System.out.println("G instance"); }
    G() { System.out.println("G constructor"); }
}
class P extends G {
    { System.out.println("P instance"); }
    P() { System.out.println("P constructor"); }
}
class C extends P {
    { System.out.println("C instance"); }
    C() { System.out.println("C constructor"); }
}
new C();
```

Output:

```
G instance
G constructor
P instance
P constructor
C instance
C constructor
```

Want More?

Inner Classes, Enums, Serialization, Cloneable, Interface Inheritance Conflicts — Next!