ChatGPT

Mastering Java OOP Concepts: Hidden Insights and Exam Tricks

---

1. Inheritance

Concepts:

- Single inheritance allowed, multiple inheritance via interfaces only.
- Constructor chaining: Parent constructors always execute before child constructors.
- Static block runs once on class loading; instance initializer block runs every time an object is created.
- Private members are never inherited.
- Access modifier rules: subclass methods cannot reduce visibility.

Hidden Tricks:

- If no `super()` call is explicitly made, Java implicitly calls the parent no-arg constructor.
- Final classes cannot be extended.
- Static members belong to class, not instances.

---

2. Method Overriding vs Hiding

Overriding:

- Applies only to instance methods.
- Resolved at runtime (dynamic binding).
- Signature and return type must be compatible (covariant return allowed).
- Access level cannot be more restrictive.

Hiding:

- Applies only to static methods.
- Resolved at compile-time (static binding).
- No `@Override` allowed for hiding; adding it causes compile error.
- Calls depend on reference type, not object type.

Example:

```
class Parent {
    static void m() { System.out.println("parent"); }
}
class Child extends Parent {
    static void m() { System.out.println("child"); }
}
```

```
Parent p = new Child();
p.m(); // Output: parent
```

---

3. Final Class, Method, Variable

- `final class`: no subclass allowed.
- `final method`: no overriding allowed.
- `final variable`: must be initialized once; acts as a constant.

Types:

- Instance level: initialized in constructor/block.
- Static: initialized inline or in static block.
- Blank final variables: declared without assignment, must be assigned in constructor.

Trick:

- Changing object state referenced by final reference is allowed.

---

4. Abstract Class vs Interface

Feature	Abstract Class	Interface
Inheritance	Single	Multiple
Methods	Abstract + Concrete	Abstract + Default + Static
Variables	Instance + Static allowed	Only `public static final`
Constructors	Allowed	Not allowed
Access Modifiers	Any	All methods are public by default

Trick:

- Interfaces from Java 8 onward support `default` and `static` methods.
- Variables inside interfaces are always `public static final`.

---

5. Method Overloading Priority

Order:

1. Exact match (e.g., `int` for `int`)
2. Widening primitive (e.g., `int` → `long`)
3. Autoboxing (e.g., `int` → `Integer`)
4. Varargs (e.g., `int...`)

Example:

```
void m(int x) {...}        // Preferred
void m(long x) {...}       // Widening
void m(Integer x) {...}    // Autoboxing
void m(int... x) {...}     // Varargs
```

---

6. Boxing and Unboxing

- **Boxing**: primitive → wrapper object.
- **Unboxing**: wrapper → primitive.
- Autoboxing/unboxing happens automatically in Java 5+.

Hidden Issues:

- `NullPointerException` when unboxing null.
- Caching of `Integer` values from -128 to 127.

---

7. Typecasting: Upcasting and Downcasting

- **Upcasting**: subclass → superclass (implicit, safe).
- **Downcasting**: superclass → subclass (explicit, unsafe).
- Use `instanceof` to prevent `ClassCastException`.

---

8. instanceof Keyword

- Checks object type safely before casting.
- Syntax: `if (obj instanceof ClassName)`
- Works with both inheritance and interfaces.

---

9. Shadowing vs Hiding (Variables and Static Methods)

Shadowing (variables):

```
class Parent { int x = 10; }
class Child extends Parent { int x = 20; }
Parent p = new Child();
System.out.println(p.x); // 10
```

Hiding (static methods):

```
class Parent {
    static void m() { System.out.println("Parent"); }
}
class Child extends Parent {
```

```
    static void m() { System.out.println("Child"); }
}
Parent p = new Child();
p.m(); // Parent
```

---

10. Object Class Methods

- `equals()`: default compares references; override to compare content.
- Always override `hashCode()` if `equals()` is overridden.
- Other important methods: `toString()`, `clone()`, `finalize()`, `wait()`, `notify()`.

Contracts:

- Equal objects → same hash code.
- `equals()` must be reflexive, symmetric, transitive, consistent.

---

11. Polymorphism Runtime Traps

- Static/final methods are not polymorphic.
- Overloaded methods: compile-time.
- Overridden methods: runtime.
- Avoid calling overridable methods from constructors.

---

12. Constructor & Block Order in Multi-level Inheritance

Order of Execution:

1. Static blocks in parent → child.
2. Instance blocks in parent → constructor.
3. Instance blocks in child → constructor.

Additional Hidden Tips and Tricks

- **Covariant Return Types**: overriding methods can return a subclass.
- **Anonymous Inner Classes**: can access final or effectively final variables.
- **Exception Rules**: overridden methods cannot throw broader checked exceptions.
- **Static vs Instance**: static methods can't access instance context.

Sample Practice Questions

1. What happens if a static method is "overridden"?
2. How does `final` keyword affect inheritance?
3. What's the output of:

```
Parent p = new Child();
p.staticMethod(); // ??
```

1. Explain `==` vs `.equals()` in Java.
2. When does `ClassCastException` happen in polymorphism?

How to Use This Document

- Review each topic thoroughly.
- Modify code snippets and test behaviors.
- Practice with tricky concepts like shadowing, method hiding, etc.
- Attempt mock exam questions targeting inheritance and polymorphism traps.

Let me know if you'd like this exported as a styled PDF.