Java Method Overloading vs Overriding - Exam Cheat Sheet

🔷 Java Method Overloading vs Overriding

```
          COMPILE-TIME POLYMORPHISM
                (Overloading)
------------------------------------------------
Class: Calculator
+ add(int a, int b)
+ add(int a, int b, int c)
+ add(double a, double b)

Compiler decides which method to call based on:
- Number of parameters
- Type of parameters
- Order of parameters

Example:
calc.add(2,3)       -> add(int,int)
calc.add(2,3,4)     -> add(int,int,int)
calc.add(1.5,2.5)   -> add(double,double)

Note: JVM does not decide, all fixed at compile time

================================================
           RUNTIME POLYMORPHISM
                (Overriding)
------------------------------------------------
Superclass: Vehicle
+ start()

Subclasses: Car, Bike
+ Car: start() -> "Car runs"
+ Bike: start() -> "Bike runs"

Superclass reference = new Subclass();

Vehicle v = new Car();    // reference type = Vehicle, object type = Car
v.start();                // JVM decides at runtime -> Car's start()

v = new Bike();
```

```
v.start();                // JVM decides at runtime -> Bike's start()

Note: Reference type = compile-time check, Object type = runtime decides
```

🔹 Quick Comparison Table

Feature	Overloading	Overriding
Method Name	Same	Same
Parameters	Different (number/type)	Same
Return Type	Can differ	Same / Covariant
Class Relation	Same class	Superclass + Subclass
Polymorphism	Compile-time	Runtime
Annotation	❌optional	✅ `@Override` recommended

⬜ **Exam Tip:**
- Overloading → Compiler decides
- Overriding → JVM decides at runtime
- Runtime polymorphism = Superclass reference + Subclass object