

The `Polymorphism` class demonstrates Java's runtime polymorphism through method overriding. It serves as an **educational example**, not as a DTO, service, or controller. The primary focus is to illustrate how the JVM binds method calls to the actual object type at runtime.

- Dynamic Dispatch**  
Each class (`A`, `B`, `C`, `D`) provides—or inherits—a `tueEtwas()` method. Calling this method on a base-type reference resolves to the overridden implementation in the actual object.
- Method Overriding Hierarchy**
  - `A.tueEtwas()` prints `"A"`.
  - `B.tueEtwas()` overrides and prints `"B"`.
  - `C` inherits `B`'s implementation (no override).
  - `D.tueEtwas()` overrides again to print `"D"`.
- Execution Flow**  
In `main`, four references (`A a`, `B b`, `C c`, `D d`) are assigned various instances to showcase:
  - i. `A a = new D();` → prints `D`
  - ii. `B b = new C();` → prints `B`
  - iii. `C c = new C();` → prints `B` (inherited)
  - iv. `D d = new D();` → prints `D`
- Member Variables**  
There are **no** class-level fields; all state is transient and local to `main`.
- Local Variables in `main`**
  - `A a`, `B b`, `C c`, `D d`: references used to demonstrate polymorphism.
- State Mutation**  
No mutation occurs. The example purely exercises method dispatch based on object type.

Public methods are limited to the Java entry point. All other `tueEtwas()` methods are package-private helpers.

Method	Purpose	Key Parameters	Return Value
<code>public static main</code>	Entry point; demonstrates polymorphic behavior via console I/O	<code>String[] args</code>	<code>void</code>

- Missing Access Modifiers**  
Helper methods use default (package) visibility. This may expose internals unintentionally in larger packages.
- Hardcoded Output**  
String literals (`"A"`, `"B"`, `"D"`) are hard-coded. This reduces flexibility and localization support.
- No Error Handling**  
Relies on `System.out.println` without any abstraction; cannot redirect or mock easily.
- Non-extensible Hierarchy**  
Adding new subclasses requires manual overrides; there is no interface or annotation to enforce consistency.
- Thread Safety**  
Although single-threaded, reliance on `System.out` may lead to interleaved output if used in concurrent contexts.