**🔹 তুলনামূলক টেবিল**

| **বৈশিষ্ট্য / বিষয়** | **Trait** | **Inheritance (extends)** | **Interface (implements)** | **Abstract Class** |
| --- | --- | --- | --- | --- |
| **মূল উদ্দেশ্য** | Code reuse (common method/property একাধিক class–এ ব্যবহার) | Parent → Child relation (একটা class থেকে আরেকটা class সবকিছু পায়) | Contract/blueprint define করা (class কেমন হবে তা বলে দেয়) | আংশিক abstraction (কিছু method define, কিছু implementation) |
| **Multiple ব্যবহার** | ✅ এক class–এ অনেকগুলো trait use করা যায় | ❌ একটাই parent class extend করা যায় | ✅ এক class–এ একাধিক interface implement করা যায় | ❌ একটিই abstract class extend করা যায় |
| **Object তৈরি করা যায়?** | ❌ Trait থেকে object তৈরি হয় না | ✅ Class থেকে object তৈরি হয় | ❌ Interface থেকে object তৈরি হয় না | ❌ Abstract class থেকে object তৈরি হয় না |
| **Method থাকতে পারে?** | ✅ থাকতে পারে (full code সহ) | ✅ থাকতে পারে | ❌ থাকতে পারে না (শুধু signature থাকবে) | ✅ থাকতে পারে (abstract + normal method) |
| **Property থাকতে পারে?** | ✅ থাকতে পারে | ✅ থাকতে পারে | ❌ থাকতে পারে না | ✅ থাকতে পারে |
| **Constructor থাকতে পারে?** | ❌ (Trait constructor properly কাজ করে না) | ✅ থাকতে পারে | ❌ থাকতে পারে না | ✅ থাকতে পারে |
| **Use case** | একই কোড অনেক class–এ reuse করা | Parent-child সম্পর্ক model করা | Contract/Rule enforce করা (যে method অবশ্যই implement করতে হবে) | কিছু common কোড শেয়ার + subclass কে কিছু method |

## ✅ সংক্ষেপে মনে রাখার ট্রিক

* **Trait →** শুধু কোড reuse (shortcut / helper)।
* **Inheritance →** সম্পর্ক (parent → child)।
* **Interface →** নিয়ম/চুক্তি (method কেমন হবে বলে দেয়)।
* **Abstract →** আংশিক নকশা (কিছু ready, কিছু implement করতে হবে)।