(2.) Polymorphism

→ Polymorphism means "many forms". It allows objects of different classes related by inheritance to be treated as object of a common superclass, enabling one interface to represent multiple underlying forms (behaviors)

(i) Types of Polymorphism

Type	Definition	How Achieved
a) Compile-Time	Method to be called is determined during compilation	Method overloading
b) Runtime	Method to be called is decided during program execution	Method overriding

(ii) Method Overloading (Compile-Time polymorphism / Static polymorphism)

=> Multiple methods with the same but different parameter lists within the same class.

- Keypoints :-
  - varies by parameter type, number or order
  - Return type alone cannot distinguish overloads.
  - Happens at compile time.

Syntax :-

```
class Calculator {
    int add(int a, int b) { return a+b; }
    double add(double a, double b) { return a+b; }
    int add(int a, int b, int c) { return a+b+c; }
}
```

(iii) Method Overriding (Runtime Polymorphism / Dynamic Polymorphism)

⇒ A Subclass provides a specific implementation of a method already defined in its superclass with the same signature.

- Key points —:
  - Enables dynamic behavior at ~~time~~ run time.
  - Requires inheritance
  - Allows runtime method dispatch

Syntax

```
Class Animal {
void Sound() {
Sout ("Animal Sound");
    }
Class Dog extends Animal {
  @override
  void Sound(){
  Sout(" Bark "); }
  }
```

(iv) Polymorphic Reference.

⇒ A Superclass reference variable pointing to a subclass, allowing dynamic method class.

Syntax

```
Animal myAnimal = new Dog();
myAnimal.Sound();   // output → Bark.
```

(v) Dynamic Method Dispatch

→ It is the process where a superclass reference is used to call an overriden method in the subclass, and the method that gets executed is determined at runtime based on the actual object type.

Syntax:-

```
Superclass referenceVariable = new Subclass();
referenceVariable.overridenMethod();
```

⇒ JVM chooses the method version at runtime, based on the actual object (new Subclass()), not the reference types..

(vi) Static, Final & private Methods.

Modifier	Can Be overriden?	Why
Static	NO (XX)	Belongs to the class, not an instance
final	NO (XX	prevents modification & overriding
private	NO (XX)	Not visible to subclass.