WHAT ARE THE ADVANTAGES OF POLYMORPHISM

Flexibility and Extensibility

- New classes can be added with little to no modifications to existing code.
- Polymorphism allows systems to grow and evolve without impacting the core logic structure**.**

Code Reusability

- You can write general-purpose methods that operate on different object types using a shared interface or superclass.
- There's no need to duplicate code for every new class that fits within the same hierarchy.

Clean and Maintainable Code

- Helps eliminate repetitive code and avoids long conditionals like if-else or switch.
- Promotes a clearer structure by letting each object define its own behavior**.**

Dynamic Method Binding (Runtime Polymorphism)

- The method that gets executed is chosen based on the actual object at runtime, not at compile time.
- This supports more flexible and adaptive behavior in programs.

Supports Interface-Oriented Programming

- Encourages coding against interfaces, not concrete classes, leading to loosely coupled, easily testable systems.
- Enhances modularity and simplifies swapping or updating implementations.

How is inheritance useful to achieve polymorphism in Java

Inheritance Provides a Common Structure

Inheritance allows a subclass to inherit methods and properties from a superclass. This shared structure is what makes polymorphism possible, because different subclasses can override the same method in their own way.

Enables Method Overriding (Runtime Polymorphism)
When a subclass overrides a method from the superclass, and we use a superclass reference to point to a subclass object, Java will dynamically call the overridden method at runtime. This is the essence of runtime polymorphism.

Facilitates Code Generalization
With inheritance, you can write code that works on the superclass type but still behaves correctly for any subclass.

What are the differences between polymorphism and inheritance

---

Aspect	Inheritance	Polymorphism
Definition	A mechanism where a class inherits properties and behaviors from another class.	The ability of one interface to be used for different underlying forms (objects).
Purpose	To promote code reuse and establish a parent-child relationship.	To enable flexibility by allowing objects to behave differently based on their actual class.
Relationship Type	Represents an "is-a" relationship. (e.g., Dog is an Animal)	Represents "many forms" of a method or interface.
Usage	Used when a new class wants to use the fields and methods of an existing class.	Used when different classes implement the same method in different ways.
How it's achieved	Using extends (for classes) or implements (for interfaces) in Java.	Using method overriding (runtime) and method overloading (compile-time).
Key Concept	Reusability	Flexibility