1. What are the advantages of Polymorphism?

1. **Code Reusability**: Polymorphism allows objects of different classes to be treated as objects of a common superclass. This enables code reuse through inheritance, as methods defined in a superclass can be used by subclasses without the need for reimplementation.
2. **Flexibility and Extensibility**: Polymorphism enhances the flexibility and extensibility of code by enabling the addition of new classes that implement existing interfaces or override existing methods. This facilitates easy integration of new functionalities without modifying existing code.
3. **Enhanced Readability and Maintainability**: By promoting a more modular and organized code structure, polymorphism enhances readability and maintainability. It allows developers to focus on specific functionalities within classes and promotes cleaner, more understandable code.
4. **Dynamic Binding**: Polymorphism facilitates dynamic method binding, where the appropriate method implementation is determined at runtime based on the actual type of the object. This enables more flexible and adaptable behavior during program execution.
5. **Interface-based Programming**: Polymorphism is often achieved through interfaces, enabling interface-based programming. This approach emphasizes contracts between classes, promoting loose coupling and high cohesion, which in turn leads to more robust and scalable software designs.
6. **Polymorphic Behavior**: Polymorphism allows for different behaviors to be defined for objects of different classes that share a common superclass or interface. This promotes code abstraction and encapsulation, as the specific implementation details are hidden from the caller.

2. How is Inheritance useful to achieve Polymorphism in Java?

- Inheritance in Java enables method overriding, establishes subtyping relationships, facilitates dynamic method dispatch, and promotes code reuse. Together, these features allow for the creation of flexible and extensible object-oriented designs where objects of different classes can exhibit polymorphic behavior.

3. What are the differences between Polymorphism and Inheritance in Java?

	Polymorphism	Inheritance
Purpose	Refers to the ability of objects to take on multiple forms and respond differently to the same method call based on their actual type at runtime.	Establishes a hierarchical relationship between classes, allowing subclasses to inherit attributes and methods from their superclass and extend or modify them.
Implementation	Can be achieved through method overriding, interfaces, or inheritance. It enables objects of different classes to be treated as objects of a common superclass or interface.	Achieved by creating a new class (subclass) from an existing class (superclass). The subclass inherits attributes and methods from its superclass.
Flexibility	Enhances flexibility by allowing objects to exhibit different behaviors at runtime, facilitating dynamic method dispatch.	Promotes code reuse and extensibility by allowing subclasses to inherit and build upon the functionality of their superclass.
Relationship	Can be achieved independently of inheritance through interfaces and method overriding, but often utilized with inheritance to enable dynamic behavior.	Provides a mechanism for achieving polymorphism by allowing subclasses to override methods defined in their superclass.