1. **What are the advantages of Polymorphism?**

- Flexibility: Polymorphism allows objects of different types to be treated as the same type, enabling dynamic method behavior and more flexible code.
- Code Reusability: Common methods can be used across different object types, reducing redundancy and simplifying the code.
- Maintainability: Changes in the superclass automatically apply to subclasses, making code easier to update and maintain.
- Extensibility: New subclasses can be added without modifying existing code, promoting scalability and reducing the risk of errors.

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

Inheritance enables polymorphism in Java by allowing subclasses to inherit and override methods from a common superclass. This lets subclasses provide their own implementations, while still being treated as instances of the superclass.

In other word, polymorphism allows methods to behave differently based on the actual object type, even when accessed through a superclass reference, enhancing flexibility and code reusability.

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

Definition:

- Inheritance is a mechanism where one class (subclass) acquires the properties and behaviors (methods) of another class (superclass). It allows code reuse and establishes a relationship between classes.
- Polymorphism allows objects of different classes to be treated as instances of a common superclass, enabling method behavior to vary based on the object's actual type.

Purpose:

- Inheritance is used to represent an "is-a" relationship between classes, promoting code reuse and establishing hierarchies.
- Polymorphism allows a single method or function to work with objects of multiple types, enhancing flexibility and enabling dynamic method dispatch.

How It's Achieved:

- Inheritance is achieved through extending a class (using the `extends` keyword) or implementing an interface.
- Polymorphism is achieved through method overriding (inherited methods) and method overloading, allowing a method to behave differently based on the object's type.

Use Case:

- Inheritance is used when a class needs to inherit common behavior from another class.
- Polymorphism is used when you want to use a unified interface to operate on different types of objects, even if they have different implementations.