1. What are the advantages of Polymorphism?

Polymorphism brings many benefits in object-oriented programming. First, it enhances code reusability because you can write generalized code that works with a superclass, and it will automatically support any subclass. This helps developers avoid writing duplicate code. Second, it provides flexibility and makes the program easier to maintain, since you can change or extend specific parts of the codebase without affecting the rest. Third, polymorphism makes it easier to scale a system by allowing new subclasses to be added without modifying existing logic. Fourth, it enables dynamic behavior at runtime, where the appropriate method is chosen based on the actual object's type. Lastly, it leads to cleaner and more readable code by removing the need for complex conditional structures like if-else or switch statements based on object types.

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

Inheritance is essential for achieving polymorphism in Java. It allows a subclass to inherit fields and methods from a superclass. The subclass can then override certain methods to provide its own implementation. When a superclass reference is used to point to a subclass object, Java applies dynamic method dispatch to determine the correct method to call at runtime. This mechanism is what enables runtime polymorphism. Without inheritance, objects would not share a common structure or interface, making it impossible to treat them uniformly.

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

Polymorphism and inheritance are closely related concepts in Java, but they serve different purposes and function in different ways. Inheritance is a structural concept that allows a class to inherit properties and behavior from another class. Its main purpose is to promote code reuse and establish relationships between classes. Polymorphism, on the other hand, is a behavioral concept that allows objects of different classes to be treated as instances of a common parent class or interface. It promotes flexibility and enables dynamic

behavior during program execution. While inheritance can exist without polymorphism, polymorphism cannot function without inheritance. The relationship formed through inheritance is typically described as an “is-a” relationship—for example, a Dog is an Animal. In contrast, polymorphism is about how objects can behave or be used interchangeably, depending on their shared interface or superclass. This means an object "behaves like" or "can be used as" a parent type.