1. Advantages of Polymorphism:

- Flexibility: Polymorphism enhances code flexibility by enabling objects to alter their behavior at runtime.
- Code Reusability: It promotes code reuse as methods implemented in subclasses can be utilized without rewriting the code.
- Readability: Source code becomes more readable and understandable with the use of polymorphism, aligning closely with the underlying model.

2. Inheritance's Role in Achieving Polymorphism in Java:

In Java, the combination of inheritance and polymorphism is crucial in object-oriented programming. Inheritance allows subclasses to inherit attributes and behaviors, fostering code reusability and establishing a hierarchical structure. The pivotal aspect for achieving polymorphism lies in method overriding, a feature facilitated by inheritance. Subclasses can provide specific implementations for methods in their superclass, enabling diverse behaviors within a unified interface. Polymorphism permits a reference variable of a superclass type to dynamically bind to objects of its subclasses, offering flexibility in code design. This dynamic binding enhances adaptability and readability, creating robust, extensible, and maintainable Java code.

3. What are the differences between Polymorphism and Inheritance in Java?
   a) In the realm of Java programming, polymorphism and inheritance serve distinct purposes:
      - Polymorphism is primarily concerned with establishing a unified interface for diverse types. This is achieved through method overriding, allowing a reference variable of a superclass type to dynamically refer to objects of its subclasses. The resulting flexibility is evident in the ability to use a common interface, fostering dynamic binding during runtime.
      - On the other hand, inheritance focuses on creating a hierarchical structure wherein subclasses inherit attributes and behaviors from their superclasses. This involves the creation of subclasses based on existing superclasses, with the option to override or extend inherited functionality. The flexibility inherent in inheritance lies in the

capacity to craft specialized classes that inherit and extend the functionality of more general classes.

b) In terms of relationships, polymorphism establishes a single interface for multiple types, often realized through interfaces, abstract classes, or method overriding.
   - In contrast, inheritance defines an "is-a" relationship, forming a hierarchy between subclasses and their respective superclasses.

c) Both polymorphism and inheritance contribute significantly to code reusability.
   - Polymorphism achieves this by enhancing reusability through a single method name applied to various implementations in subclasses. –
   - Inheritance promotes code reuse by enabling subclasses to inherit attributes and behaviors from their superclasses.

⇨ In summary, while polymorphism provides a unified interface for different types via method overriding, inheritance creates a hierarchical structure that fosters code reuse and establishes meaningful relationships between classes. The harmonious integration of these concepts forms a powerful foundation for crafting flexible, reusable, and efficient object-oriented programs in the Java language.