1. What are the advantages of Polymorphism?

- Polymorphism helps in maximizing code reuse. Once a class is developed and verified, it can be used multiple times across projects, saving significant effort. Additionally, updates can be made without altering the original structure.
- Fewer lines of code lead to easier and faster debugging for developers.
- A single variable can handle different types of data. Values inherited from a parent class can be customized within a child class without impacting other classes or the original parent.

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

- Concept
    - Inheritance: Mechanism for sharing properties and behavior by passing them from a parent class to its children.
    - Polymorphism: Ability for different classes to define different behaviors while sharing a common interface.
- Goal
    - Inheritance: Encourages reuse of existing code through hierarchical relationships.
    - Polymorphism: Enhances flexibility by enabling dynamic method resolution.
- Types
    - Inheritance: Forms include single, multilevel, and hierarchical inheritance.
    - Polymorphism: Divided into compile-time (overloading) and runtime (overriding) polymorphism.
- Primary concern
    - Inheritance: Deals with establishing structural relationships between classes.
    - Polymorphism: Focuses on enabling dynamic behavior depending on the object instance.
- Connection
    - Inheritance: Makes polymorphism possible by linking classes together.
    - Polymorphism: Depends heavily on inheritance for runtime polymorphism to occur.
- Benefit Focus
    - Inheritance: Strengthens code reuse and maintenance.
    - Polymorphism: Boosts adaptability by allowing behaviors to vary with object type.

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

- Inheritance sets up a class hierarchy where a child class is seen as a specialized form of its parent, laying the groundwork for polymorphism.
- When a subclass overrides methods and is accessed via a reference of the parent class type, Java chooses the right method to execute during runtime.
- It allows a unified programming model by treating subclass objects as instances of the parent class, while still preserving their unique behaviors.
- This makes it easier to work with groups of related objects in a consistent manner.
- By coding against abstract parent classes rather than specific child classes, systems become less rigid, making them easier to maintain and extend.