1. What are the advantages of Polymorphism?

- It provides reusability to the code. The classes that are written, tested and implemented can be reused multiple times. Furthermore, it
saves a lot of time for the coder. Also, the one can change the code without affecting the original code.
- A single variable can be used to store multiple data values. The value of a variable you inherit from the superclass into the subclass can be changed without changing that variable's value in the superclass; or any other subclasses.
- With lesser lines of code, it becomes easier for the programmer to debug the code.

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

- Inheritance defines a hierarchical connection between classes, where a subclass is considered a specific type of its superclass.
This relationship is key to enabling polymorphism in Java.
- When a subclass overrides a method from its superclass, and an object of the subclass is accessed through a superclass reference,
Java determines the correct method to call at runtime.
- By treating subclass objects as instances of their superclass, Java enables a consistent interface. This allows developers to write
code that operates on the superclass type while still benefiting from the subclass's behavior.
- A common benefit of polymorphism is the ability to process a collection of objects uniformly.
- By relying on abstract superclass types rather than specific subclass implementations, systems become less dependent on concrete
classes. This promotes loose coupling, improves code modularity, and makes the system easier to update or extend.

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

- Definition & Concept
    + Inheritance is about reusing code by allowing a subclass to inherit properties and methods from a superclass.
    + Polymorphism is about using a common interface to operate on different types of objects in different ways.
- Purpose
    + Inheritance establishes a hierarchical relationship ("is-a") and promotes code reuse.
    + Polymorphism promotes flexibility and dynamic behavior, allowing method calls to be resolved at runtime.
- Types
    + Inheritance types include: single, multilevel, hierarchical.
    + Polymorphism types include: compile-time (overloading) and runtime (overriding).
- Focus
    + Inheritance focuses on the structure and shared behavior of related classes.

+ Polymorphism focuses on behavior — how methods behave differently depending on the object.

- Relationship

+ Inheritance creates a connection between classes, enabling polymorphism.

+ Polymorphism often depends on inheritance to work, especially in the case of method overriding.

- Code Reusability vs. Flexibility

+ Inheritance provides reusability by passing down code.

+ Polymorphism provides flexibility by allowing method behavior to change based on object type.