1. What are the advantages of Polymorphism?

- IMPORTANT POINTS ABOUT POLYMORPHISM

A functionality can behave differently for different instances

The behavior depends on the type of data used in the operation

Polymorphism is used for implementing inheritance.

- ADVANTAGES OF POLYMORPHISM

It helps programmers reuse the code and classes once written, tested and implemented. They can be reused in many ways.

Single variable name can be used to store variables of multiple data types(Float, double, Long, Int etc).

Polymorphism helps in reducing the coupling between different functionalities.

- DISADVANTAGES OF POLYMORPHISM

One of the disadvantages of polymorphism is that developers find it difficult to implement polymorphism in codes.

Run time polymorphism can lead to the performance issue as machine needs to decide which method or variable to invoke so it basically degrades the performances as decisions are taken at run time.

Polymorphism reduces the readability of the program. One needs to identify the runtime behavior of the program to identify actual execution time.

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

Inheritance facilities Polymorphism in 2 ways:

- Method Overriding: Inheritance allows subclasses to provide their own implementation of methods that are already defined in the superclass. When a method is invoked on an object of the superclass, Java's dynamic method dispatch mechanism ensures that the appropriate overridden method in the subclass is called at runtime. This enables polymorphic behavior, where the same method call produces different results depending on the actual type of the object.
- Polymorphic References: In Java, you can declare a reference variable of a superclass type and use it to refer to objects of either the superclass or any of its subclasses. This allows you to treat objects of different subclasses uniformly through a common interface provided by the superclass. At runtime, Java determines the actual type of the object being referred to and invokes the appropriate methods accordingly.
- Inheritance Hierarchies: Polymorphism can be further extended through inheritance hierarchies, where classes are organized in a hierarchical structure. This allows for more complex polymorphic behavior, where subclasses inherit and override methods from their direct superclass, and can also be treated polymorphically through references of more distant superclasses.

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

- Inheritance:

Purpose: Inheritance is a mechanism that allows a class (subclass or child class) to inherit properties and behaviors (methods and fields) from another class (superclass or parent class). It facilitates code reuse and establishes an "is-a" relationship between classes.

Syntax: In Java, inheritance is achieved using the extends keyword. Subclasses inherit members (methods and fields) of the superclass unless they are marked as private or final.

Usage: Inheritance promotes code reuse by allowing subclasses to leverage the functionality defined in their superclass. It enables hierarchical organization of classes and establishes a parent-child relationship between them.

- Polymorphism:

Purpose: Polymorphism is the ability of objects of different classes to be treated as objects of a common superclass. It allows methods to behave differently based on the actual type of the object that they are invoked on.

Types: Polymorphism in Java can be achieved through method overriding (runtime polymorphism) and method overloading (compile-time polymorphism).

Usage: Polymorphism allows for more flexible and dynamic code. It enables the use of a superclass reference to refer to objects of its subclasses, facilitating code abstraction and simplifying interactions between objects.