1. **What are the advantages of Polymorphism?**
    - Polymorphism enhances code reusability by enabling objects from various classes to be regarded as instances of a shared class.
    - Polymorphism improves code readability and maintainability by minimizing the volume of code required and managed.
    - Polymorphism facilitates dynamic binding, ensuring the appropriate method is invoked during runtime according to the object's actual class.
    - Polymorphism enables treating objects as a unified type, simplifying the creation of generic code capable of managing objects of diverse types.
2. **How is Inheritance useful to achieve Polymorphism in Java?**
    - Inheritance allows a new class (subclass) to inherit the properties (fields) and methods of an existing class (superclass).
    - Example: I have a class `Animal` with attributes like name and methods like `eat()`. Different types of animals (`dog, cat,`...) can inherit from `Animal` and share these common traits.
    - Polymorphism literally means "many forms". In Java, it refers to the ability of objects to exhibit different behaviors at runtime.
    - By creating a hierarchy of classes, inheritance sets the stage for polymorphism. Subclasses can override inherited methods, providing their own implementation specific to that subclass.
    - Example: a `Dog` subclass might override the `eat()` method from `Animal` to specify that dogs eat kibble.
    - With inheritance, you can create a generic reference variable. This variable can then hold references to objects of any subclasses. When you call a method on this reference, the actual behavior depends on the object's type at runtime. The correct overridden method gets executed.

      That is how inheritance is useful to achieve polymorphism in Java.

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

Polymorphism	**Inheritance**
It allows the methods of a class to be defined in multiple forms.	It is one in which a new class is created (subclass) that inherits the features from the already existing class (superclass).

Types of polymorphism:
Compile-time polymorphism/ method overloading;
Run-time polymorphism/method overriding.
	Types of inheritance:
	single inheritance;
	multi-level inheritance;
	multiple inheritance;
	hybrid inheritance;
	hierarchical inheritance.

It is basically applied to functions/methods.	It is basically applied to classes.
It allows the object of the class to decide which form it has to take to work with methods and attributes of the class.	It supports the concept of reusability and reduces code duplication in object-oriented programming.