The Difference between Abstract Class and Interface

Both abstract classes and interfaces are used to define contracts for classes to follow.

Abstract Classes:

Definition:

An abstract class is a type of class that cannot be instantiated on its own. It may contain abstract methods, which are methods without a body, as well as concrete methods (methods with implementations).

Purpose:

Abstract classes serve as blueprints or templates for other classes. They can contain common functionality that can be shared among subclasses while allowing flexibility for specific implementations.

Usage:

Abstract classes are used when you want to create a base class that enforces certain methods to be implemented by its subclasses. They provide a way to define common behaviour without necessarily providing complete implementations.

Interfaces:

Definition:

An interface in programming defines a contract that classes can implement. It contains method signatures, properties, and functionality that implementing classes must adhere to.

Purpose:

Interfaces are used to define a structure or shape that a class should follow. They enable multiple unrelated classes to implement the same set of methods or properties.

Usage:

Interfaces are helpful when you want to enforce certain shapes or structures for classes to implement. They provide a way to define a contract that classes must fulfil.

Key Differences:

Instantiation:

Abstract classes cannot be instantiated directly, while interfaces cannot be instantiated at all.

Methods and Properties:

Abstract classes can contain both abstract (unimplemented) methods and concrete (implemented) methods along with properties. Interfaces can contain only method signatures and properties but no implementations.

Inheritance:

Abstract classes support inheritance and can have constructors. Interfaces can't contain any implementations and can't inherit directly from other interfaces. A class can implement multiple interfaces, but it can inherit from only one abstract class.

In summary, abstract classes and interfaces serve as important tools for defining contracts and enforcing structure within object-oriented programming, each with its distinct use cases and limitations. Abstract classes provide a mix of partially implemented and unimplemented methods, while interfaces solely define method signatures and properties that classes must adhere to. Both are valuable in facilitating code organization, reuse, and ensuring adherence to defined structures and behaviours within software development projects.