DZone

When to Use Abstract Class and Interface

by Akash Deep · Jun. 04, 16 · Java Zone · Tutorial

Abstract Class

An abstract class is a class that is declared **abstract**—it may or may not include abstract methods. Abstract classes cannot be instantiated, but they can be subclassed. An abstract class may have static fields and static methods. When an abstract class is subclassed, the subclass usually provides implementations for all of the abstract methods in its parent class. However, if it does not, then the subclass must also be declared abstract.

An abstract method is a method that is declared without an implementation (without braces and followed by a semicolon), like this:

```
abstract void sum(int a, int b);
```

Consider using abstract classes if any of these statements apply to your situation:

1. You want to share code among several closely related classes.
2. You expect that classes that extend your abstract class have many common methods or fields or require access modifiers other than public (such as protected and private).
3. You want to declare non-static or non-final fields. This enables you to define methods that can access and modify the state of the object to which they belong.

Interface

Interface

An interface is just the declaration of methods of an Object, it's not the implementation. In an interface, we define what kind of operation an object can perform. These operations are defined by the classes that implement the interface. Interfaces form a contract between the class and the outside world, and this contract is enforced at build time by the compiler.

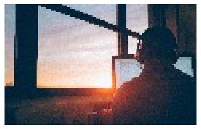
```
interface Vehical {
            // declaration
            void changeGear(int newValue);
             void speedUp(int increment);
             void applyBrakes(int decrement);
}
class Car implements Vehical {
         int speed = 0;
         int gear = 1;
         // implementation
         void changeGear(int newValue) {
                 gear = newValue;
          }
          void speedUp(int increment) {
                 speed = speed + increment;
          }
           void applyBrakes(int decrement) {
                 speed = speed - decrement;
           }
            void printStates() {
                  System.out.println(" speed:" + speed + " gear:" + gear);
            }
}
```

Consider using interfaces if any of these statements apply to your situation:

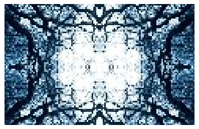
1. You expect that unrelated classes would implement your interface. For example, the interfaces Comparable and Cloneable are implemented by many unrelated classes.
2. You want to specify the behavior of a particular data type, but not concerned about who implements its behavior.

3. You want to take advantage of multiple inheritances.

Like This Article? Read More From DZone



Abstract Classes and OOP Extras in PHP



OOP in F# - Define and Implement Classes, Abstract Classes, and Interfaces

What Is an Object?

Free DZone Refcard
Java Containerization

Topics: JAVA , OBJECT ORIENTED PROGRAMMING , OOP , ABSTRACT CLASS , TUTORIAL