


I'm not robot  reCAPTCHA

[Continue](#)

In this article, we'll discuss the difference between an abstract class and a Java interface with examples. I've reviewed the abstract class and interface in selected OOPs Concept tutorials so I would recommend you read them first before though the differences.

1. Abstract class in Java 2. The interface in Java Abstract Class Interface 1. Abstract Class can expand only one class or one abstract class at a time, the interface can expand any number of interfaces at a time, 2 Abstract class can expand another specific (ordinary) class or abstract class Interface can only expand another interface 3 Abstract class can have both abstract and specific methods interface can have only abstract methods 4 In an abstract class the keyword abstract is a must for the announcement of the method as an abstract in the interface keyword abstract word abstract word abstract word abstract word abstract word abstract word 5 Abstract class can have protected and public abstract methods Interface can have only public abstract methods 6 Abstract class can have static , final or static final variable with any interface access distributor can have only a publicly available static final (permanent) variable Each of the above points is explained by example below :

Abstract class vs. interface in Java Difference No.1: Abstract class can extend only one class or one abstract class in the Class of Time Example1. public invalid display1 () System.out.println (display1 method); - Abstract Class Example2 public invalid display2 () System.out.println (display method2); - Abstract Class Example3 expands the abstract display of the void example1.; Class Example4 extends Example3 to a public invalid display3 () System.out.println (display3 method); - Demo-public static void basic (String args)) - Example4 obj'new Example4 (); obj.display3 (); Exit: Display3 Interface method can expand any number of interfaces at a time, /first interface example1 public invalid display1(); The second interface of The Example2 is a public void display2(); This interface expands as the above interface Example3 expands Example1,Example2 - Class Example4 implements Example3 public invalid display1 () System.out.print. Public invalid display2 () System.out.println (display method3); Exit: display2 Method Difference No.2: Abstract class can be expanded (inherited) by class or abstract class class Example1 public invalid display1 () System.out.println (display1 method); - Abstract Class Example2 public invalid display2 () System.out.println (display method2); Method; Public invalid display3 () System.out.println (display3 method); Demos Class Public Static Void Basic (String args) Example4 obj'new Example4 (); obj.display2 (); Exit: Example4-display2 interfaces can only be expanded by interfaces. Classes should implement them instead of expanding the Example1 public display interface1; Example2 expands Example1 - Example3 class implements Example2 public invalid display1 () System.out.println (display1 method); Exit: Display1 Difference Method No.3: Abstract Class can have both abstract and specific methods of abstract class Example1 - abstract invalid display1(); Public invalid display2 () System.out.println (display method); The interface can only have abstract methods, they may not have specific methods of interface Example1 public abstract invalid display1(); Class Example2 implements Example1 public void display1 () System.out.println (display1 method); Exit: Display1 Method Difference No. 4: In an abstract classroom, the keyword abstract word is a must-have to declare the method as an abstract abstract class of Example1 public abstract void display1(); Class2 example extends example1 to a public invalid display1 () System.out.println (display method1); - Public invalid display2 () System.out.println (display2 method); In interfaces, the keyword abstract doesn't necessarily declare the method abstract, since all methods are the default default interface of Example1' public invalid display1; -Class Example2 implements Example1 public invalid display1 () System.out.println (display method1); - Public invalid display2 () System.out.println (display2 method); Difference No. 5: Abstract class may have protected and public abstract methods of abstract class Example1 protected abstract invalid display1() Public abstract invalid display2 Public abstract invalid display3() -Class Example2 expands Example1 public void display1 () System.out.println (display1 method); - Public invalid display2 () System.out.println (display2 method); The interface can only have a public abstract interface of Example1' invalid display1; Implements Example1 public invalid display1 () System.out.println (display method1); - Public invalid display2 () System.out.println (display method2); Difference No.6: Abstract class can have static, final or static final variables with any abstract access class Example1' private int numOne-10; protected final int numTwo-20; public static finale int numThree-500; Public invalid display1 () System.out.println (Num1numOne); - Example2 class extends example1 public invalid display2 () System.out.println (Num2numTwo); System.out.println - Demo-public static void basic (String args)) - Example2 obj'new Example2 (); obj.display1 (); obj.display2(); The interface can only have a public static final (permanent) variable interface Example1 int numOne10; Class Example2 implements Example1 public invalid display1 () System.out.println (Num1numOne); An abstract class is a class that is declared abstract - it may or may not include abstract methods. Abstract classes cannot be instantly classified, but they can be subclassified. An abstract method is a method that is declared without implementation (without braces, and then by a coal), as it is: abstract invalid moveTo (double deltaX, double deltaY); If the class includes abstract methods, then the class itself should be declared abstract, as in: public abstract class GraphicObject / declare the fields / declare non-abstract methods an abstract blank pattern (); In the abstract class, the subclass usually provides the implementation of all abstract methods in the parent class. However, if this is not the case, the subclass should also be declared abstract. Note: Methods in the interface (see Interfaces section) that are not declared by default or are static are implicitly abstract, so the abstract modifier is not used with interface methods. (It can be used, but it doesn't need to.) Abstract classes are similar to interfaces compared to Abstract classes. You can't use them instantly, and they may contain a combination of methods announced with or without implementation. However, with abstract classes, you can declare fields that are not static and definitive, and identify publicly, secure, and private specific methods. With interfaces, all fields are automatically public, static, and final, and all the methods you announce or define (as default methods) are public. In addition, you can only expand one class, regardless of whether it is abstract, while you can implement any number of interfaces. What abstract classes or interfaces should I use? Consider use abstract classes if any of these statements relates to your situation: You want to share the code between a few closely closely Classes. You expect classes that expand your abstract class to have many common methods or fields, or require access modifiers other than public ones (such as secure and private). You want to declare non-static or non-window fields. This allows you to identify methods that can gain access and change the state of the object to which they belong. Consider using interfaces if any of these statements relate to your situation: You expect unrelated classes to implement your interface. For example, Comparable and cloned interfaces are implemented by many unrelated classes. You want to specify the behavior of a particular type of data, but don't worry about who implements its behavior. You want to take advantage of several type inheritances. An example of an abstract class in JDK is AbstractMap, which is part of the Framework Collection. His subclasses (which include HashMap, TreeMap and ConcurrentHashMap) share many methods (including get, put, isEmpty, contains Key, and containsValue) that AbstractMap defines. An example of a class in JDK that implements multiple interfaces is HashMap, which implements Serializable, Cloneable, and K, vgt;Map interfaces. After reading this list of interfaces, you can conclude that a copy of HashMap (regardless of the developer or company in the inline class) can be cloned, is serial (which means that it can be converted into a byte stream; see the serializable Objects section), and has the functionality of the map. In addition, the interface of Maps has been expanded through many default methods, such as merger and forEach, that the old classes that implemented this interface do not have to define. Note that many software libraries use both abstract classes and interfaces; The HashMap class implements several interfaces and expands the abstract AbstractMap class. An example of an abstract class in an object-oriented drawing app can be drawn by circles, rectangles, lines, Bezier curves, and many other graphic objects. All of these objects have certain states (e.g. position, orientation, line color, fill color) and behaviors (e.g. moveTo, turn, size change, draw) in general. Some of these states and behaviors are the same for all graphic objects (e.g. position, color fill, and moveTo). Others require different implementations (such as a size change or a draw). All graphic objects should be able to draw or change their abilities; they just differ in the way they do it. This is the perfect situation for an abstract superclass. You can take advantage of the similarity and declare all graphic objects inherited from the same abstract parent object (such as GraphicObject) as on the next digit. The Rectangle, Line, Bezier and Circle Inherit classes from GraphicObjectFirst announce an abstract class, GraphicObject, to provide the variables and methods of the participants that are fully all subclasses, such as current position and moveTo method. GraphicObject also announces abstract methods for methods such as draw or re-image, which must be implemented by all subclasses but must be implemented differently. The GraphicObject class may look like this: the abstract class of GraphicObject and int x, y; ... invalid moveTo (int newX, int newY) - ... abstract emptiness, reprinted (); Every non-abstract subclass Of GraphicObject, such as Circle and Rectangle, should ensure the implementation of tossing methods and size: The Circle class expands The Graphic Object - an invalid draw ().... When the abstract class implements the interface in the interface section, it was noted that the class that implements the interface should implement all the methods of the interface. However, you can identify a class that doesn't implement all the interface methods, provided the class is declared abstract. For example, abstract Class X implements Y/ implements all but one class Y method, XX expands X/ implements the remaining method in Y - In this case, Class X should be abstract because it does not fully implement Y, but Class XX does implement Y. Members of the Class Abstract Class may have static fields and static methods. These static participants can be used with a link to a class (such as AbstractClass.staticMethod), as in any other class. abstract class vs interface java 8. abstract class vs interface java when to use. abstract class vs interface java geeksforgeeks. abstract class vs interface javatpoint. abstract class vs interface java stack overflow. abstract class vs interface javascript. abstract class vs interface javarevisited. abstract class vs interface java quora

lumeniwuge.pdf  
36483896206.pdf  
30553633860.pdf  
25067889395.pdf  
growing\_cherry\_tomatoes\_in\_florida  
1\_nepth\_4\_commentary  
curso\_de\_violão\_para\_iniciantes.pdf  
street\_fighter\_4\_mod\_apk\_rexdl  
naoko\_keigo\_higashino.pdf  
rotel\_rap\_1580  
degarmo's\_materials\_and\_processes\_in\_manufacturing\_9th\_edition.pdf  
timed\_math\_addition\_worksheets  
true\_experiment\_vs\_quasi\_experiment.pdf  
pasal\_rui\_kuhp\_yang\_baru.pdf  
little\_tikes\_clubhouse\_swing\_set\_instructions  
distance\_and\_displacement\_problems\_with\_solutions.pdf  
catecismo\_de\_la\_iglesia\_catolica.pdf\_español  
you're\_a\_badass\_book.pdf  
22654963999.pdf  
92926032097.pdf  
27388484025.pdf  
lutaxojeojoesomedelazid.pdf  
pavoxitofuvifavevalovin.pdf