Abstract class example in android

In Java Abstraction is one of the main building blocks. This is the process of hiding the internal work and showing only the necessary details. In a simple form abstraction means: Show functionality to hide the complexity of an important note: The interface is used to achieve 100% abstraction in JAVA. Here we discuss abstraction only in detail, but we recommend that you learn the topic of abstraction and interface at the same time, as both topics are similar with a small difference. Explanation of abstraction: The best explanation for abstraction is to use a common example of sending SMS. When a user sends a text message, they simply enter the address, type the message into the text box and send it. He doesn't know what's going on behind the scenes when that message is sent to another user. So the details of the implementation were hidden from the user, and only the functionality presented to the user. This is achieved in JAVA, either through abstraction or through the interface. Key Points of Learning on Abstraction: Before we share syntax you first need to understand abstract class, abstract methods, and why/how to expand an abstract class: What is an abstract class: an abstract class is announced with an abstract keyword that can contain methods without the body, with the body or the mixture of both. If a class has at least one method without a body, it should be declared abstract. Important note: A method without a body means a method without any functionality or implementation. In the example below, you can see one method of calling without a body. So here the class should be declared abstract, because it has one method without a body. Abstract Class Mobile abstract invalid call (); Abstract Invalid Messages () - System.out.println What is an abstract method: Methods without a body or methods with only signatures are called abstract methods. The abstract method has no details of implementation. It is announced using an abstract keyword in front of the method name. For example, the call method is an abstract method. Abstract empty calling Syntax: an abstract method of abstract emptiness method_name;) Important note: To use the abstract method, you first need to override this method in the subclass (i.e. in the children's class). Why and how to expand the abstract class: As you now know, an abstract class can contain an abstract method without any implementation and thus we cannot instantly abstract class. If we try to use it instantly, the object may be useless because the methods within the abstract class may have no implementation at all. So to avoid this situation, JAVA does not allow us an instantly abstract class. First we need to expand the abstract class where we will make the implementation of these abstract and then we instantly had this new class. So let's see how to expand the abstract class: Step 1: First, we define the abstract class by an abstract method: the abstract class Mobile Abstract Challenge (); Method Without a Body - Step 2: Now We Are We Abstract class and implementation of abstract methods: Samsung class expands mobile void call (); System.out.println Pro Note: An example of the abstract class abstractMap, which is part of the frame collection include TreeMap, HashMap and ConcurrentHashMap and share many methods like isEmpty ,), put (), get (), containValue (), containKey () etc. that defines AbstractMap. Abstraction syntax: The syntax of abstraction begins with an abstract keyword in front of the class name. It then contains a mixture of abstract and non-abstract methods. (with body/implementation) // If it contains atleast one abstract method, then the class should be declared abstract - Examples of abstraction: Example 1: Allows you to now understand the concept of abstraction using real life examples of different sounds created by animals. For example, Cat makes Meow and Leo do a roar. We will display different sounds using abstraction in JAVA. Step 1: First, create a new project in Eclipse and create an abstract Sound class. Below is the code of the abstract class Sound.java Sound /Non-abstract method, i.e. the method with the implementation of public invalid sound management ()) - System.out.print (Animal Sound); Abstract method, i.e. without the body/implementation of abstract empty sound (); Step 2: Now create another Cat class name that expands the Sound class. Here we will introduce an abstract sound () method for Cat. Below is cat.java code Cat expands the sound of the sound emptiness () soundmessage (); System.out.println (Кошка: Мяу); Step 3: In the same way we create another Lion class that expands the Sound class. Here also we will implement the sound () method, but for Leo. Below is the code Lion.java: The Lion class expands the sound and the empty sound ()) System.out.println (Lion: Roar); Step 4: Now create a basic AnimalSound class. Here, we will first create a Cat and Lion class object, and then call the sound method on these objects. Below is the code AnimalSound.java of the public class AnimalSound - public static emptiness core (String) - Cat cat - new cat (); cat.sound(); Lion lion and new lion lion.sound(); Exit: Now run the program and you'll see the sounds of Cat and Lion printed. Animal Sound Cat: Meow Animal Sound of the Lion: Roar Conclusion: So you can see how useful abstraction is. We simply define a mixture of abstract and non-abstract methods in an abstract classroom, and then introduce an abstract method into the children's class (i.e. subclass) as required. As a result, the same method gives a different result depending on the objects of which subclass. Remember that we can't instantly abstract class as discussed earlier. Example 2: In the second example of abstraction, we just текст AbhiAndroid, используя абстрактный метод </ClassName> </ClassName> Basic class. In this class, Child expands the base class and overrides the display method, and then prints the text, as shown in the example below. Here in this example we have provided many explanations in the code itself as comments. Below is the code of the basic.java abstract class Baza /abstract class - abstract invalid display (); abstract method - the code Child.java is given below, and the main function of the Child class expands the basic class, which expands the property of the basic class - invalid display () / / to override the method of the basic class - System.out.println (AbhiAndroid); prints AbhiAndroid text as a day off - public static emptiness of the main (String args) /core function of the class Child c'new Child (); Creation of a c.display children's class facility(); Call the method of displaying a child class with an object - OUTPUT: AbhiAndroid Conclusion: In the above example, the base is an abstract class that contains an abstract method. Its implementation is provided by the Child class. The importance of abstraction below is one of the highlights of abstraction in Java. Abstraction helps reduce complexity and also improves the convenience of maintaining the system. Abstraction gives more energy to object-oriented programming languages when used with concepts such as encapsulation and polymorphism. Abstraction is used to solve the problem of the real world. The difference between abstract class and interface: The abstract class is very similar to the interface, as both contain an abstract method, and we can't initiate both of them. But below are some key differences between the two topics: Abstract Class: Abstract Class can only expand one class at a time. Abstract class can have both abstract (method without implementation) and specific methods (implementation method). In an abstract class, the abstract keyword is used to declare the method abstract. An abstract class can have protected, public and public abstract methods.

An abstract class can have static, final or static final variables with any access hitter. Interface: The interface can expand any number of interfaces at the same time. The interface can only have abstract methods, they can't have specific methods. In interfaces, an abstract keyword doesn't necessarily declare the method abstract because all methods are abstract by default. Interfaces can only have publicly available abstract methods, i.e. by default. Interfaces can only have a static final variable, i.e. by default. A class declared using an abstract keyword is known as an abstract class. It can have abstract methods (methods without the body) as well as specific methods (ordinary with the body). A normal class (not an abstract class) cannot have abstract methods. In this guide, we learn what an abstract class is, why we use it, and what rules we should remember when working with it in Java. Abstract class can't be which means you can't create an object of it. Why? We'll discuss this later in this guide. Why abstract class? Let's say we have a class of animals that has a method of sound () and subclasses (see inheritance) of it, like a dog, a lion, a horse, a cat, etc. This is because each class of child has to override this method to give their own details of implementation, as The Lion Class will say Roar in this method and the dog class will say Woof. Therefore, when we know that all classes of children's animals will and should override this method, there is no point in implementing this method in the parent class. So making this method abstract would be a good choice because by making this method abstract, we force all subclasses to implement this method (otherwise you will get a compilation error), nor do we need to give any implementation of this method in the parent class. Because the Animal class has an abstract method, it is necessary to declare this class abstract. Now every animal must have a sound by making this method abstract, we have made it mandatory for the children's class to give details of the implementation of this method. So we ensure that every animal has a sound. Abstract Class Example /Abstract Parent Class Abstract Class Animal /abstract method of public abstract empty sound (); Dog Class expands animal class public class Dog expands Animal' public void sound ()) Exit: Woof Therefore, for these kinds of scenarios we usually declare the class abstract, and then specific classes expand these classes and override the techniques accordingly and can have their own methods as well. The Abstract Declaration class Abstract Class lays out the methods, but does not necessarily implement all the methods. Declaration using the abstract keyword of abstract class A///This abstract method of abstract emptiness myMethod(); This is a specific method with the void of the body anotherMethod ()/ Something / Rules Note 1: As we have seen in the above example, there are cases where it is difficult or often unnecessary to implement all methods in the parent class. In these cases, we can declare the parenting class abstract, making it a special class that is not complete in itself. A class derived from an abstract class must implement all the methods that are declared abstract in the parent class. Note 2: An abstract class can't be instantaneous, which means you can't create an object of it. To use this class, you need to create another class that expands this class and implements abstract methods, then you can use the object of this class of children to call no parenting methods, as well as implemented methods (those that have been in the parents, but implemented in the class of children). Note 3: If a child does not implement all abstract methods of abstract parenting, the child's class must also be declared abstract. Do you know? Because the abstract class allows for specific methods as well, it does not provide 100% abstraction. It can be said that it provides a partial abstraction. Abstraction is a process in which you show only appropriate data and hide unnecessary object details from the user. Interfaces, on the other hand, are used for 100% abstraction (More on abstraction here). You can also read this: The difference between an abstract class and an interface in Java Why can't we create an abstract class object? Because these classes are incomplete, they have abstract methods that don't have a body, so if Java allows you to create an object in that class, then if someone calls an abstract method using that object, what happens? There will be no actual implementation of the method for the call. It is also because the object is specific. The abstract class is similar to a pattern, so you have to expand it and build on it before you can use it. An example to demonstrate that an object creating an abstract class is not allowed as stated above, we cannot instantly abstract class. This program throws a compilation error. AbstractDemo's public void myMethod system.out.println (Hello); - abstract public emptiness anotherMethod (); - Public Demo Class Expands AbstractDemo Public Void anotherMethod () - System.out.print (Abstract Method); - Public static emptiness basic (String args) - /error: You can't create an object of its AbstractDemo obj - the new AbstractDemo(); obj.anotherMethod. Exit: Unsolved compilation problem: Can't instantly type AbstractDemo Note: A class that expands an abstract class, must implement all the abstract methods of it, otherwise you should declare that the class is abstract as well. An abstract class versus a particular Class A, which is not abstract, is called a specific class. In the example above, which we saw at the beginning of this guide, Animal is an abstract class, and Cat, Dog, and Lion are specific classes. Key points: The abstract class doesn't make sense until it's expanded by some other class. If you are announcing an abstract method in the classroom, then you should declare the class abstract as well. you can't have an abstract method in a particular class. This is not always true: if a class does not have any abstract method, it can also be marked as abstract. It may have an abstract method (specific) as well. I've reviewed the rules and examples of abstract methods in a separate tutorial, you can find Here: The abstract method in Java Now allows you to just see some basics and an example of an abstract method. 1) The abstract method has no body. 2) Always finish the declaration with the column (;). 3) It should be redefined. Abstract Abstract should be expanded and in the same way the abstract method should be redefined. 4) The class must be declared abstract in order to have abstract methods. Methods. how to use abstract class in android example