Abstraction: it is the process when we hide implementation details and show only essential information to the user.

We can achieve it by using abstract class or interface.

Important rules for abstract methods are mentioned below:

- Any class that contains one or more abstract methods must also be declared abstract.

- If a class contains an abstract method it needs to be abstract. (But abstract class can have not abstract methods)

- If a non-abstract class extends an abstract class, then the class must implement all the abstract methods of the abstract class else the concrete class has to be declared as abstract as well.

**Зачем вообще нужны абстрактные классы?**

Когда у нас родительским классом является нечто что не существует в реальности, это некий набор данных (свойства/поля) и поведения (методы) который общий для целого ряда девайсов(сущностей), а конкретная имплементация это классы наследники.

Абстрактные классы в Java используются для представления общих концепций или типов. Они служат как база для создания более конкретных классов. Допустим, у вас есть абстрактный класс «Животное». Вы не можете создать объект типа «Животное», потому что «Животное» — это слишком общее понятие. Но вы можете создать классы «Кошка» и «Собака», которые наследуют от абстрактного класса «Животное».

Но, что нам мешает сделать класс животное конкретным(т.е. не абстрактным), почему мы должны выбрать именно абстрактную версию класса? Java - язык относящийся к Объектно Ориентированному подходу в программировании. Это значит что любой объект из реального мира мы можем создать в программе, предварительно описав как он должен выглядеть и что делать, т.е. создав класс. Если какого то объекта в природе не существует то и в программе не должна существовать такая сущность, это основа основ объектного подхода. Здесь нужно руководствоваться логикой. В реальной жизни мы не можем встретить животное как животное, это не логично, соответственно и JAVA придерживается этого правила, нельзя создовать обьекты от обстрактного класса. Да но зачам нам абстрактный класс? Для некой архитектуры (либо спосаба создания классов) наличие которой позволяет понять как построена программа человеку который её не создавал. Да но давайте зделаем сразу класс Кошка, класс Собака…

1)В этом случае у нас повторяется код в каждом из классов

2)Мы не можем объекты к примеру кошка и собака хранить в какойто коллекции.

3)-Java реализует ООП, соответственно обьект животное нельзя создавать, это как бы противоречит подходу.

4)-Наверно как-то мешает полиморфизму(не уверен)

Abstract class

--------------

-Provides partial abstraction (it can have abstract methods as well as regular methods)

-Abstract method is declared without implementation

-Abstract class cannot be directly instantiated (создан экземпляр)

-Sub class can access Abstract class using extends keyword (since we cannot create object, only inheritance should be used to access the abstract class)

-Sub class must implement all abstract methods i.e. Overriding is compulsory (принудительный)

-Abstract class can have parameterized constructor

--------------

Advantage

Reduces complexity by hiding implementation

Better viewing

Avoids code duplication and promotes reusability

Increases security by providing only important details to the user

--------------

E.g. ← example

Shapes: Abstract

Triangle: SubClass1

Square: SubClass2