

We do not currently allow content pasted from ChatGPT on Stack Overflow; read our policy [here](#).

How does abstraction help in hiding the implementation details in Java?

Asked 5 years, 3 months ago Modified 1 year ago Viewed 8k times



8



Abstraction is a process of hiding the implementation details and showing only functionality to the user.

Another way, it shows only important things to the user and hides the internal details. So below is an example where an abstract class is made and abstract methods are overridden.

But the thing i didn't understand is how it is hiding the implementation details?

```
abstract class Bank
{
    abstract int getRateOfInterest();
}

class SBI extends Bank
{
    int getRateOfInterest()
    {
        return 7;
    }
}

class PNB extends Bank
{
    int getRateOfInterest()
    {
        return 8;
    }
}

class TestBank{
public static void main(String args[])
{
    Bank b;
    b=new SBI();
    System.out.println("Rate of Interest is: "+b.getRateOfInterest()+" %");
    b=new PNB();
    System.out.println("Rate of Interest is: "+b.getRateOfInterest()+" %");
}
}
```

[java](#) [abstract-class](#) [abstraction](#)

Share Edit Follow

edited Jun 26, 2021 at 12:24



[syntagma](#)

22.7k 14 76 131

asked Sep 4, 2017 at 16:36



[Tarunjeet Singh Salh](#)

184 2 12

Possible duplicate of [Abstraction VS Information Hiding VS Encapsulation](#) – RafatMunshi Sep 6, 2017 at 18:15

2 Answers

Sorted by:

Highest score (default)



The **abstraction** in your code is the abstract class itself:

18

```
abstract class Bank {  
    abstract int getRateOfInterest();  
}
```



and the rest is the **implementation** (and the implementation details), specifically: classes PNB and SBI



But the thing i didn't understand is how it is hiding the implementation details?

Imagine you have a bank comparison engine, which is represented by the BankComparisonEngine class. It will just take a Bank (**abstract** class) as an argument, then get its interest rate and save it to its internal database, like so:

```
class BankComparisonEngine {  
    public void saveInterestRateOf(Bank bank) {  
        int rate = bank.getRateOfInterest();  
        // Save it somewhere to reuse later  
    }  
}
```

How are the implementation details hidden exactly? Well, BankComparisonEngine does not know how getRateOfInterest() method of a concrete implementation of Bank works (that is: PNB.getRateOfInterest() OR SBI.getRateOfInterest() is implemented). It only knows it is a method that returns an int (and that it should return an interest rate). The **implementation details** are hidden inside the concrete classes that extend abstract class Bank .

Share Edit Follow

answered Sep 4, 2017 at 16:48




syntagma

22.7k 14 76 131

Also, when we use the imported libraries objects and their methods, we dont worry about the implementation of those methods, we just got the functionality to use. Hence by OOP we got the important things without the internal details. Thats abstraction! – RafatMunshi Sep 6, 2017 at 17:57

- 1 So it's like the user is aware that there is a functionality called getRateOfInterest() but it's implementation is unknown to him. But I have a doubt here. Even if a normal class say PNB Bank had been seperately declared without deriving an abstract class but having same functionality getRateOfInterest() defined in it and when a user is using that method by creating an object of the

class, it will be just enough to hide the function's implementation. So why then do we need an abstract class / abstraction? – [Mithra](#) Sep 6, 2021 at 11:45 

Isn't that polymorphism though? – [Michael](#) Apr 1 at 21:31

Mithra that is exactly the question i have. People say "In java, abstraction is achieved by abstract classes and interfaces". But normal methods in normal classes also hide implementation and just offer the public services. Then people go ahead and give an explanation like @syntagma's explanation. But they end up just describing polymorphism. – [Michael](#) Apr 1 at 21:35



1



When a client has to use your object, the client need not import your class or have your class definition in his jar, he/she can just import the abstract class or interface and accept your object as an argument, like explained in bank example. So client will still be able to use your object with parent reference but can't actually see (or need) the functional implementation of the class. Hope this cleared your doubt.

Share Edit Follow

answered Oct 26, 2021 at 16:22



[deepak s](#)

11 1