Exact difference between Inheritance and Abstraction

See more: C#2.0 C#3.0 C# C#4.0

Rate this: ☆☆☆☆☆

Can somebody tell me what is the exact difference between Inheritance and Abstraction. i.e. In which scenario I have to use only inheritance and in which scenario i have to use only Abstract classes.

Posted 4-May-12 18:47pm
Rakesh N Bhavsar

Add a Solution

3 solutions

Top Rated Most Recent

Solution 1

Rate this:

Hi,

Of course Abstraction is interested with Inheritance. But there are some important points.
First of all when you mark a class as abstract you cannot create an instance. This is very important. Abstract classes are used to be inherited only.

But when you inherited from a non abstract class you can create instance from both of derived and base classes.

So we can say abstract classes are used to generate new types,to be inherited not to create instance !

And of course abstraction is interested with Polymorphism. Abstraction forces us to Polymorphism.

For example you have a base Employee class like below:

Hide Copy Code

```
public class Employee
    {
        public string Name { get; set; }
        public double Salary { get; set; }

        public void IncreaseSalary(double rate)
        {
            this.Salary += (this.Salary * rate / 100);
        }
    }
```

Now when we create SalesRepresentative class like below we should inherit it from Employee because SalesRepresentative is an Employee.

Hide Copy Code

```
public class SalesRepresentative : Employee
    {
```

Also Read

Public, Private, and Hybrid Cloud: What's the difference?

Related Questions

Abstract classes, inherit, overwrite etc.

Differences between interface and abstract class

Inheritance of abstract class

abstraction and abstract

abstract and interface difference

multiple inheritance in c#.net - abstract classes

Does not implement inherited abstract member

does not implement inherited abstract class

Abstract class ....

What is the difference between an abstract class and normal public class?

```
        public double AnnualSalesAmount { get; set; }
    }
```

Now SalesRepresentative object has IncreaseSalary method because it is inherited from Employee. But in generally Sales Representatives' and Employee's Salaries are increased by different ways for example according to their AnnualSalesAmount.

In this case you should be able to change method code of IncreaseSalary from SalesRepresentative but you can't. Actually now you are across with Polymorphism

Now Let's come to abstraction. If you want to change the default code of IncreaseSalary from inherited class there are 2 choices. First marking the method as Virtual. And the second one is marking it as abstract.

The difference is If you mark it as virtual. You don't have to implement it in SalesRepresentative but If you mark it as abstract you have to implement it and you should'nt forget an abstract member can only be in abstract classes. Examine the example below

Hide Copy Code

```
public abstract class Employee
   {
       public string Name { get; set; }
       public double Salary { get; set; }

       public abstract void IncreaseSalary(double rate);

   }
```

Hide Copy Code

```
public class SalesRepresentative : Employee
    {
        public double AnnualSalesAmount { get; set; }

        public override void IncreaseSalary(double rate)
        {
            if (this.AnnualSalesAmount > 100000)
            {
                this.Salary += (this.Salary * (rate + 5) / 100);

            }
            else
            {
                this.Salary += (this.Salary * rate / 100);
            }
        }
    }
```

Posted 4-May-12 20:05pm
thursunamy

Permalink

Comments

Nima.naqipoor 5-May-12 1:54am
good explain! :)

Solution 2

Rate this: ★★★★★

`Abstraction` and `Inheritance` are the important features having wide implications in `Object Oriented Programming`. The full features may not be covered exhaustively in a short answer, and these are dealt elaborately in various books and articles. However, to get a quick idea they can described as below:

`Abstraction` from Object Oriented Programming perspective is extracting the core features of an object to deal without being specific about the implementation details. For example if we take a vehicle then it can be abstracted like a an object having `Wheels, Seats, Gears` among other things, which can be treated as the members of a `Vehicle` class describing its composition. To add behaviour to the Vehicle there can be methods like `GiveHorn, Drive, ChangeGear, ApplyBrake` etc.

Then specific classes can be derived from this base class using `Inheritance`. Say `a Car, a Bus`. So `a Car` is a vehicle with 4 wheels, 4 gears and say 4 seats. The `Drive, Change Gear, Apply brake` can be implemented within this derived class either afresh or they can used as provided in the base class or with a combination as per the requirement. Similarly a Bus is derived from the Vehicle class using inheritance and it has 6 wheels, 20 seats, 5 gears. The methods can be implemented as explained above.

In this connection the following articles may be helpful.
Abstraction (computer science)[^]
Inheritance (object-oriented programming)[^]

The abstraction and inheritance can be implemented either with `Abstract` classes or `Interfaces` and the following references may be helpful in this regard.

Abstract Class versus Interface[^]
Abstract Class Vs Interface[^]

The answers given to this question may also be helpful.
Purpose of Interfaces[^]

Posted 4-May-12 20:11pm
VJ Reddy

Solution 3

Rate this: ☆☆☆☆☆

Read this article:

Introduction to Object Oriented Programming Concepts (OOP) and More[^]

hope it helps :)

Posted 4-May-12 20:36pm
Uday P.Singh

Add your solution here

B *I* U ~~S~~ small BIG code var < > & link [^] encode untab case indent outdent

Preview

...

Existing Members

Sign in to your account

Your Email

Password

Forgot your password?

...or Join us

Download, Vote, Comment, Publish.

Your Email

Optional Password

✓ I have read and agree to the Terms of Service and Privacy Policy

✓ Please subscribe me to the CodeProject newsletters

Submit your solution!

When answering a question please:

1. Read the question carefully.
2. Understand that English isn't everyone's first language so be lenient of bad spelling and grammar.

3. If a question is poorly phrased then either ask for clarification, ignore it, or **edit the question** and fix the problem. Insults are not welcome.
4. Don't tell someone to read the manual. Chances are they have and don't get it. Provide an answer or move on to the next question.

Let's work to help developers, not make them feel stupid.