Home

PUBLIC

Stack Overflow

Tags

Users

Jobs

Overloading and overriding

Ask Question

What is the difference between overloading and overriding.

99

48

c# inheritance

edited Sep 4 '12 at 21:13

Kirill Kobelev
8,850 5 22 45

asked Mar 23 '09 at 15:00

james *

2 @james: did you at least try google?? – Mitch Wheat Mar 23 '09 at 15:02

I'm guessing you're not going to get this one. – Jon B Mar 23 '09 at 15:09

james- probably not and I feel quite silly not being able to explain a simple concept – james Mar 23 '09 at 15:26

@james my guess is you intuited these concepts without knowing the terms. I'm going to guess you're self-taught. You should probably pick up a beginner level C# book, not to learn but to reinforce what you already know. It's mind-numbing but will help immensely in tech interviews. – Michael Meadows Mar 23 '09 at 15:44

you can read more on this link : intquesans.blogspot.com/2011/05/... – asd May 29 '11 at 17:51

12 Answers

207

Overloading

Overloading is when you have multiple methods in the same scope, with the same name but different signatures.

```
//Overloading
public class test
{
    public void getStuff(int id)
    {}
    public void getStuff(string name)
    {}
}
```

Overriding

Overriding is a principle that allows you to change the functionality of a method in a child class. *

```
//Overriding
public class test
{
        public virtual void getStuff(int id)
        {
            //Get stuff default location
        }
}
```

```
public class test2 : test
{
        public override void getStuff(int id)
        {
            //base.getStuff(id);
            //or - Get stuff new location
        }
}
```

edited May 4 '16 at 20:00

user3885927
2,095 1 10 30

answered Mar 23 '09 at 15:01

cgreeno
26.7k 6 62 85

1 i love this answer @cgreeno ..Thanks – Ferrakkem Bhuiyan May 6 '14 at 19:17

1 @cgreeno i think you miss `void` for `getStuff` in `test2` – Mahdi Mar 2 '15 at 20:04

2 just to mention that the base class method must be marked as virtual to override else you'll get a compilation error. – Alex Stephens May 21 '15 at 18:38

Hey, this might be old but I still have a question about this. Does overloading make it available to have more functionality with 'one' method? Like I can do this: `getStuff(2, "Hello world!");` or I can do this `getStuff("Myname", "Mysurname", "Hello World!");` ? Anyone can confirm this is the way to do it? – Sj03rs Aug 31 '16 at 8:48

34

Simple definitions for overloading and overriding

Overloading (Compile Time Polymorphism):: Functions with same name and different parameters

```
public class A
{
    public void print(int x, int y)
    {
        Console.WriteLine("Parent Method");
    }
}

public class B : A
{
    public void child()
    {
        Console.WriteLine("Child Method");
    }

    public void print(float x, float y)
    {
        Console.WriteLine("Overload child method");
    }
}
```

Overriding (Run Time Polymorphism):: Functions in the extended class with same name and same parameters as in the base class, but with different behaviors.

```
public class A
{
    public virtual void print()
    {
        Console.WriteLine("Parent Method");
    }
}
```

```
public class B : A
{
    public void child()
    {
        Console.WriteLine("Child Method");
    }

    public override void print()
    {
        Console.WriteLine("Overriding child method");
    }
}
```

edited Sep 29 '14 at 8:06

Chamika Sandamal
19.6k 3 52 76

answered Feb 27 '13 at 6:36

Devrath
22.9k 40 132 189

Perhaps you would like to add that the overriding concept applies for the parent class-sub class relationship. – Freakyuser Feb 27 '13 at 6:56

8

I want to share an example which made a lot sense to me when I was learning:

This is just an example which does not include the virtual method or the base class. Just to give a hint regarding the main idea.

Let's say there is a Vehicle washing machine and it has a function called as "Wash" and accepts Car as a type.

Gets the Car input and washes the Car.

```
public void Wash(Car anyCar){
       //wash the car
}
```

Let's overload Wash() function

Overloading:

```
public void Wash(Truck anyTruck){
   //wash the Truck
}
```

Wash function was only washing a Car before, but now its overloaded to wash a Truck as well.

- If the provided input object is a Car, it will execute Wash(Car anyCar)
- If the provided input object is a Truck, then it will execute Wash(Truck anyTruck)

Let's override Wash() function

Overriding:

```
public override void Wash(Car anyCar){
   //check if the car has already cleaned
   if(anyCar.Clean){
       //wax the car
   }
   else{
       //wash the car
       //dry the car
       //wax the car
   }
}
```

Wash function now has a condition to check if the Car is already clean and not need to be washed again.

- If the Car is clean, then just wax it.

- If not clean, then first wash the car, then dry it and then wax it

.

So the functionality has been overridden by adding a new functionality or do something totally different.

answered Aug 16 '16 at 22:13

curiousBoy
2,640 2 26 34

1 This must be the answer. Voted+ – Willys Feb 13 '17 at 13:46

8

- Overloading = Multiple method signatures, same method name
- Overriding = Same method signature (declared virtual), implemented in sub classes

An astute interviewer would have followed up with:

What's the difference between overriding and shadowing?

edited May 23 '17 at 12:03

Community ♦
1 1 *

answered Mar 23 '09 at 15:07

Michael Meadows
23.5k 4 42 59

Never knew it was called shadowing, good to know. The only relation overloading and overriding have is overing. – Samuel Mar 23 '09 at 15:11

4

As Michael said:

- Overloading = Multiple method signatures, same method name
- Overriding = Same method signature (declared virtual), implemented in sub classes

and

- Shadowing = If treated as DerivedClass it used derived method, if as BaseClass it uses base method.

answered Mar 23 '09 at 15:10

Migol
4,512 8 39 66

3

Having more than one methods/constructors with same name but different parameters is called overloading. This is a compile time event.

```
Class Addition
{
   int add(int a, int b)
   {
     return a+b;
   }                                    *
   int add(int a, int b, int c)
   {
     return a+b+c;
   }

   public static main (String[] args)
   {
     Addition addNum = new Addition();
     System.out.println(addNum.add(1,2));
     System.out.println(addNum.add(1,2,3));
```

```
    }
}
```

O/p:

```
3
6
```

Overriding is a run time event, meaning based on your code the output changes at run time.

```
class Car
{
    public int topSpeed()
    {
        return 200;
    }
}
class Ferrari extends Car
{
    public int topSpeed()
    {
        return 400;
    }
    public static void main(String args[])
    {
        Car car = new Ferrari();
        int num= car.topSpeed();
        System.out.println("Top speed for this car is: "
    }
}
```

Notice there is a common method in both classes topSpeed(). Since we instantiated a Ferrari, we get a different result.

O/p:

```
Top speed for this car is: 400
```

answered Aug 28 '14 at 10:56

Matt

118 1 11

2

in C# there is no Java like *hidden* override, without keyword **override** on overriding method! see these C# implementations:

variant 1 without *override*: result is 200

```
class Car {
    public int topSpeed() {
        return 200;
    }
}

class Ferrari : Car {
    public int topSpeed(){
            return 400;
    }
}

static void Main(string[] args){
    Car car = new Ferrari();
    int num= car.topSpeed();
    Console.WriteLine("Top speed for this car is: "+
    Console.ReadLine();
}
```

variant 2 with *override* keyword: result is 400

```
class Car {                          *
    public virtual int topSpeed() {
        return 200;
    }
}

class Ferrari : Car {
    public override int topSpeed(){
            return 400;
    }
}
```

```
static void Main(string[] args){
    Car car = new Ferrari();
    int num= car.topSpeed();
    Console.WriteLine("Top speed for this car is: "+
    Console.ReadLine();
}
```

keyword **virtual** on Car class is opposite for **final** on Java, means *not final*, you can override, or implement if Car was abstract

edited Mar 16 '16 at 4:42

answered Feb 11 '16 at 5:01

Nurlan

196 1 7

1

Another point to add.

Overloading More than one method with Same name. Same or different return type. Different no of parameters or Different type of parameters. In Same Class or Derived class.

int Add(int num1, int num2) int Add(int num1, int num2, int num3) double Add(int num1, int num2) double Add(double num1, double num2)

*

Can be possible in same class or derived class. Generally prefers in same class. E.g. Console.WriteLine() has 19 overloaded methods.

Can overload class constructors, methods.

Can consider as Compile Time (static / Early Binding) polymorphism.

==========

Overriding cannot be possible in same class. Can Override class methods, properties, indexers, events.

Has some limitations like The overridden base method must be virtual, abstract, or override. You cannot use the new, static, or virtual modifiers to modify an override method.

Can Consider as Run Time (Dynamic / Late Binding) polymorphism.

Helps in versioning http://msdn.microsoft.com/en-us/library/6fawty39.aspx

==========

Helpful Links

http://msdn.microsoft.com/en-us/library/ms173152.aspx
Compile time polymorphism vs. run time polymorphism

edited May 23 '17 at 11:33

Community ♦
1 1

answered Aug 6 '14 at 17:29

Sai *
868 1 10 23

Good one, with the needed details. – user3885927 May 4 '16 at 19:05

1

shadowing = maintains two definitions at derived class and in order to project the base class definition it shadowes(hides)derived class definition and vice versa.

answered Feb 17 '11 at 7:10

ramya choudary
11 1

1

Overloading is a part of static polymorphism and is used to implement different method with same name but different signatures. **Overriding** is used to complete the incomplete method. In my opinion there is no comparison between these two concepts, the only thing is similar is that both come with the same vocabulary that is over.

edited Oct 4 '16 at 15:14

Athafoud
2,091 2 28 44

answered Apr 11 '16 at 0:23

Moeed
13 3

Captain here: Overloading a method gives you multiple signatures of different methods with the same name. Overriding hides an inherited member, either from classes or interfaces. *flies away* – SparK Oct 4 '16 at 14:05

1 @SparK Overriding doesn't hide a member. You can *either* override *or* hide an inherited member. The two are mutually exclusive options. – Servy Oct 4 '16 at 15:35

@Servy, wrong choice of words, I meant "substitutes". – SparK Oct 4 '16 at 17:36

0

Method overloading and Method overriding are 2 different concepts completely different. Method overloading is having the same method name but with different signatures. Method overriding is changing the default implementation of base class method in the derived class. Below you can find 2 excellent video tutorials explaining these concepts.

Method overriding Vs hiding

Method overloading

answered Jun 24 '12 at 16:34

Suresh
657 8 4

0

Overloading is the concept in which you have same signatures or methods with same name but different parameters and overriding, we have same name methods with different parameters also have inheritance is known as overriding.

edited Nov 28 '18 at 10:05

Pang
6,989 16 66 105

answered Dec 8 '13 at 15:38

waqas ur Rehman
1

*