

# Generalization and Specialization in C#

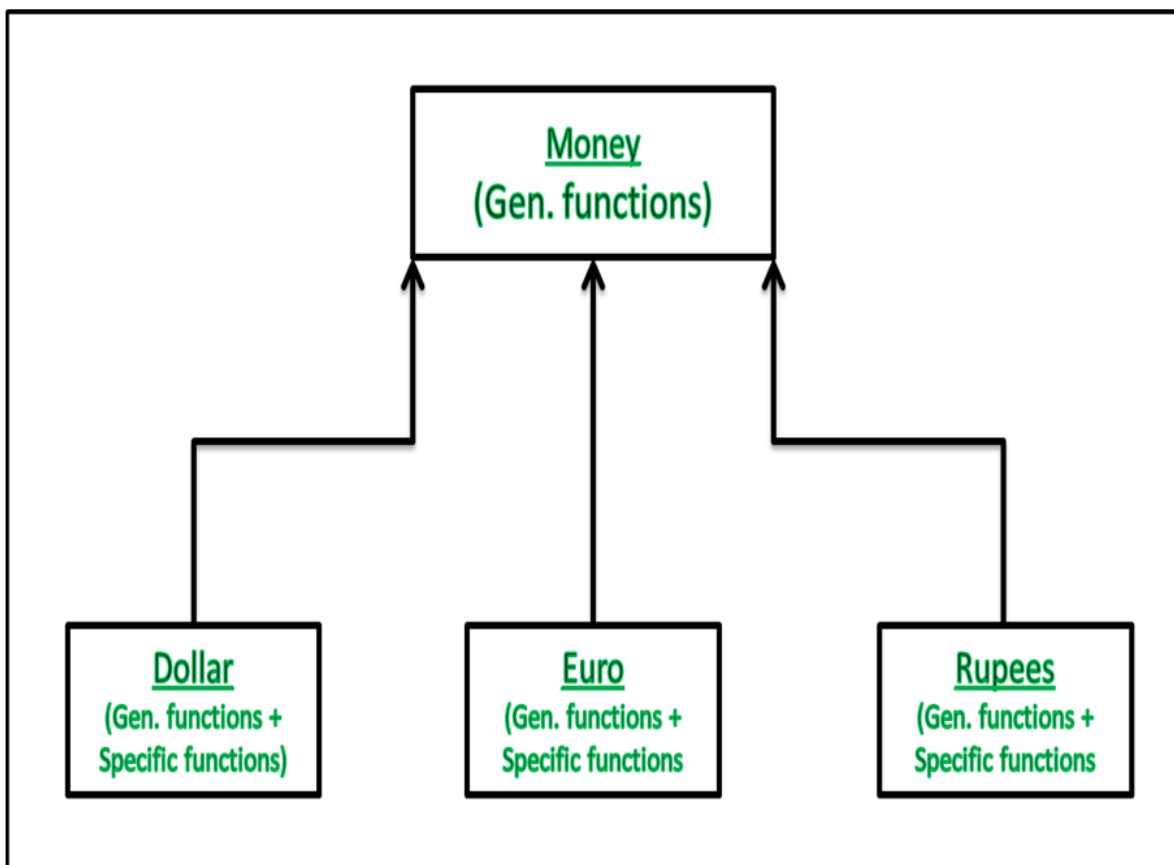
## General class?

Loosely speaking, a class which tells the main features but not the specific details. The classes situated at the top of the inheritance hierarchy can be said as General.

## Specific class?

A class which is very particular and states the specific details. The classes situated at the bottom of the inheritance hierarchy can be said as Specific.

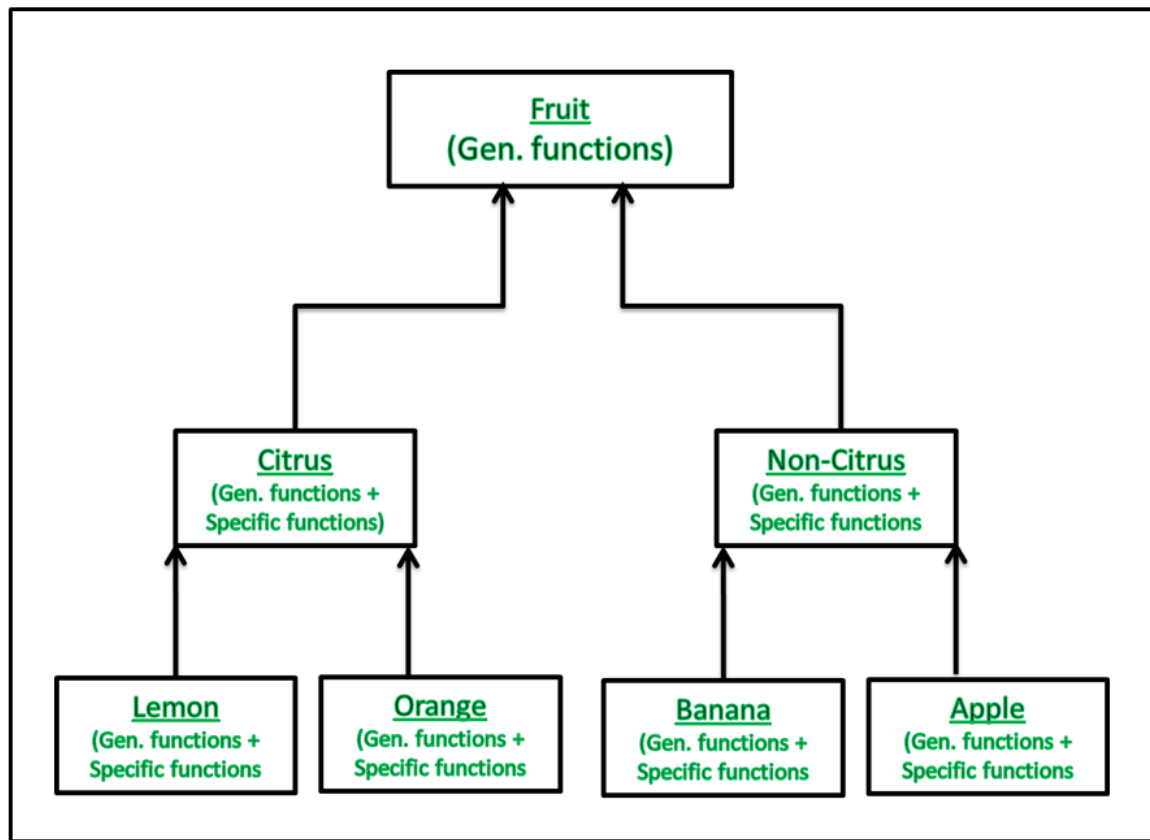
### Example 1:



Relatively General Class: Money

Relatively Specific Class: Dollar, Euro, Rupees

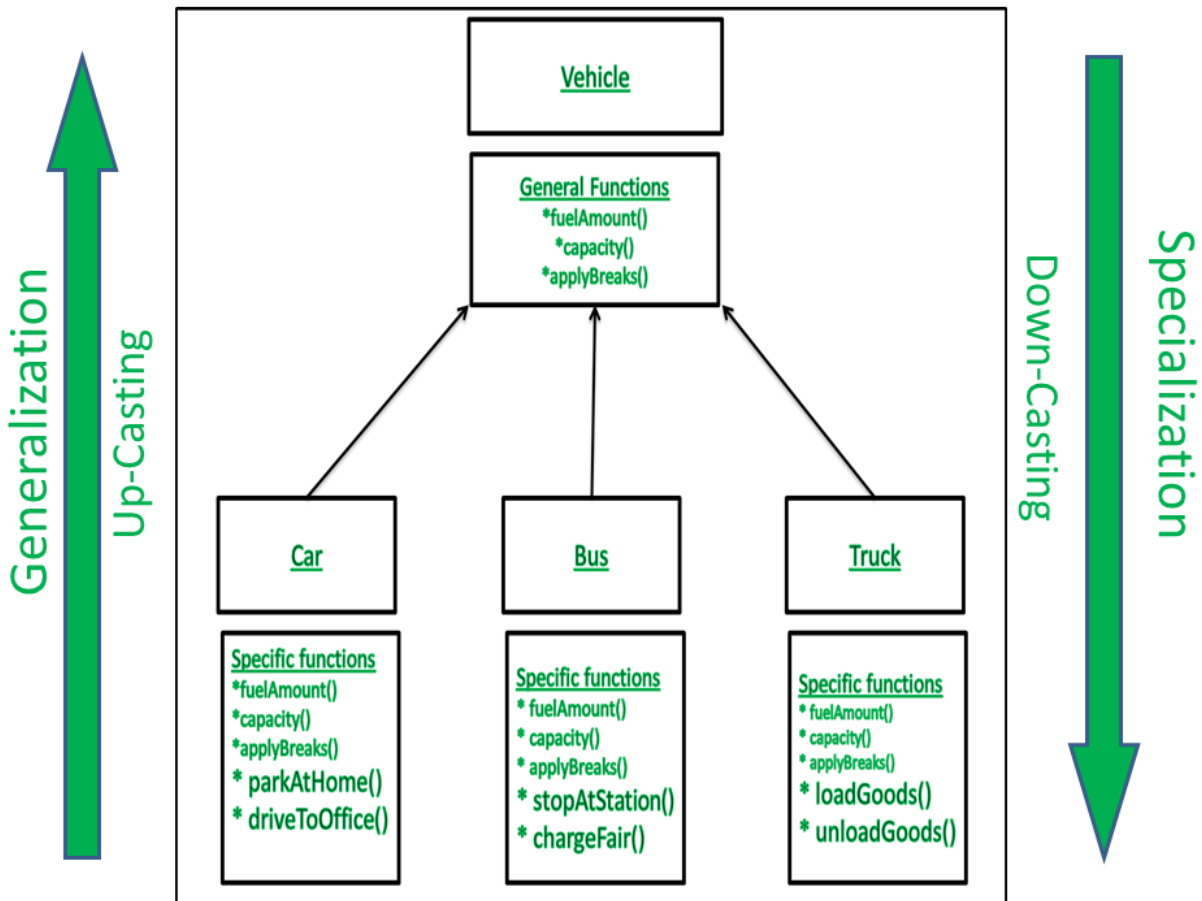
### Example 2:



Lemon, Orange are more Specific than Citrus  
Banana, Apple are more Specific than Non-Citrus  
Citrus, Non-Citrus are more Specific than Fruit  
Fruit is most general class

#### **Conversion of one class type to another class type?**

We can convert references to one class type to another class type in C#. But for the conversion to happen the classes should be related with each other by the way of inheritance.



Therefore,

- References for Vehicle and Bus **can** be type-casted to each other.
- References for Vehicle and Car **can** be type-casted to each other.
- References for Vehicle and Truck **can** be type-casted to each other.
- References for Bus, Car and Truck **can't** be type-casted to each other.

### Generalization

Converting a subclass type into a base class type is called '**Generalization**' because we are making the subclass to become more general and its scope is widening. This is also called **widening or up casting**. Widening is safe because the classes will become more general.

For example, if we say Car is a Vehicle, there will be no objection. Thus C# compiler will not ask for cast operator in generalization.

### **Example:**

```

class Father
{
    public void work()
    {

```

```

        Console.WriteLine("Earning Father");
    }
}

class Son : Father
{
    public void play()
    {
        Console.WriteLine("Enjoying son");
    }
}

class Main
{
    static void Main(String[] args)
    {
        // father is a base class reference
        Father father;

        // new operator returns a subclass reference
        father = (Father) new Son();

        // which is widened using casting
        // and stored in father variable
        // Though casting is done but it is not needed
        father.work();

        // Uncomment next lone to see the error
        // father.play();
    }
}

```

### Output:

Earning Father

So, in widening or Generalization, **we can access all the base class methods, but not the subclass methods.**

**Example:** Now Suppose we override the base class methods in sub class

```

class Father
{
    public void work()
    {
        Console.WriteLine("Earning Father");
    }
}

class Son : Father {

    public override void work()
    {
        Console.WriteLine("Earning Son");
    }
}

```

```

class Main
{
    static void Main(String[] args)
    {

        // father is the base class reference
        Father father;

        // new operator returns a subclass reference
        father = (Father) new Son();

        // which is widened using casting
        // and stored in father variable
        // Though casting is done but it is not needed

        // subclass method is invoked
        father.work();
    }
}

```

### Output:

```
Earning Son
```

### Specialization

Converting a base class type into a sub class type is called '**Specialization**'. Here, we are coming down from more general form to a specific form and hence the scope is narrowed. Hence, this is called **narrowing** or **down-casting**.

Narrowing is **not safe** because the classes will become more and more specific thus giving rise to more and more doubts. For example if we say Vehicle is a Car we need a proof. Thus, In this case, C# compiler specifically asks for the casting. This is called **explicit casting**.

**Example:** To show when Narrowing is not allowed

```

class Father
{
    public void work()
    {
        Console.WriteLine("Earning Father");
    }
}

class Son : Father {
    public void play()
    {
        Console.WriteLine("Enjoying son");
    }
}

class Main
{
    static void Main(String[] args)
    {

```

```

try {
    // son is a sub class reference
    Son son;

    // new operator returns a base class reference
    // which is narrowed using casting
    // and stored in son variable

    // This will throw exception
    son = (Son) new Father();

    // Through a narrowed reference of the base class
    // we can neither access base class method
    // and nor the subclass methods

    // Below lines will show
    // an error when uncommented
    // son.work();
    // son.play();
}
catch (Exception e) {
    Console.WriteLine(e);
}
}

```

### Output:

```

System.InvalidCastException : Unable to cast object of type
'ProjectName.Father' to type ' ProjectName.Son'.

```

### Example:

```

class Father
{
    public void work()
    {
        Console.WriteLine("Earning Father");
    }
}

class Son : Father {
    public void play()
    {
        Console.WriteLine("Enjoying son");
    }
}

class Main
{
    static void Main(String[] args)
    {
        // son is a subclass reference
    }
}

```

```
Father father;

// new operator returns a subclass reference
// which is stored in the father variable
// father stores a Father class reference
// because of implicit casting
father = new Son();

// father is narrowed
Son son = (Son)father;

son.work(); // works well
son.play(); // works well
}
```

**Output:**

Earning Father

Enjoying son

**Conclusion:**

1. When a base class reference (referring to base class object) is narrowed, then using that reference we can access neither methods of subclass nor methods of base class.
2. When a subclass reference (referring to subclass object) is widened and then again narrowed, then using that reference we can access all the methods of the subclass as well as the base class. This is the same as simple base class reference referring to base class object where base class methods have got inherited.