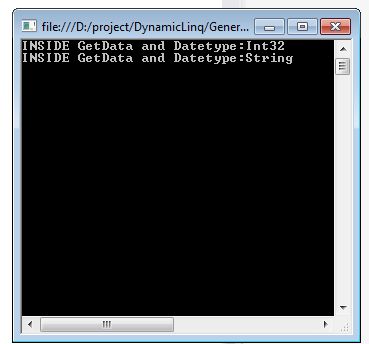
**Introduction**  
  
I would like to introduce generic method overloading.  
  
It would be good for you to already understand generics in C# but if not then please visit the following link.  
  
[Basics of Generic Classes in C#](http://www.c-sharpcorner.com/UploadFile/deveshomar/basic-of-generic-class-in-C-Sharp/)  
  
First we will understand simple overloading then move towards generic overloading.  
  
**Steps and code**  
Add the following class to the project:

1. **public** **class** SimpleClass
2. {
3. **public** **void** GetData(**int** x)
4. {
5. Console.WriteLine("INSIDE GetData and Datetype:" + x.GetType().Name);
6. }
8. **public** **void** GetData(**string** xStr)
9. {
10. Console.WriteLine("INSIDE GetData and Datetype:" + xStr.GetType().Name);
11. }
12. }

**Program.cs**

1. **class** Program
2. {
3. **static** **void** Main(**string**[] args)
4. {
5. SimpleClass o = **new** SimpleClass();
6. o.GetData(345);
7. o.GetData("Devesh is testing the code");
8. Console.ReadKey();
9. }
10. }

**Running the Code**  
  
The following will be the output:  
  
  
 **Understanding the code**  
  
We have created the GetData function with overloaded functions having integer and string parameters.  
  
When we passed an integer to the method first GetData (int x ) is called else if we passed a string then GetData(string xStr) is called.  
  
  
  
Now we have a basic knowledge of method overloading.  
   
Let us move towards **Generic Method Overloading**.  
  
Now we are updating the GetData() method as a generic method.

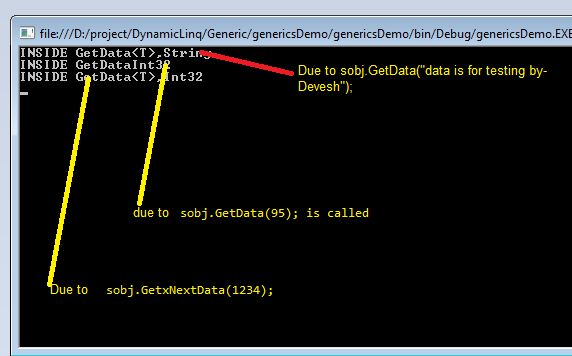
1. **public** **void** GetData<T>(T obj)
2. {
3. Console.WriteLine("INSIDE GetData<T>,"+ obj.GetType().Name);
4. }

The following would be the complete code:

1. **public** **class** SimpleDemoClass
2. {
3. **public** **void** GetData<T>(T obj)
4. {
5. Console.WriteLine("INSIDE GetData<T>,"+ obj.GetType().Name);
6. }
7. **public** **void** GetData(**int** x)
8. {
9. Console.WriteLine("INSIDE GetData" + x.GetType().Name);
10. }
11. **public** **void** GetxNextData<T>(T obj)
12. {
13. GetData(obj);
14. }
15. }

**Program.cs**

1. **class** Program
2. {
3. **Static** **void** Main(**string**[] args)
4. {
5. SimpleDemoClass sobj = **new** SimpleDemoClass();
6. sobj.GetData("data is for testing by-Devesh");
7. sobj.GetData(95);
8. sobj.GetxNextData(1234);
9. Console.ReadKey();
10. }
11. }

**Running code**  
  
  
  
Understanding the code.  
  


* When we called sobj.GetData("data is for testing by-Devesh");
* Getdata<T>(T obj) is called. Because we are passing a string and this is implicitly an object
* When we called sobj.GetData(95);, GetData(int x ) is called because we are passing an integer to the GetData method.
* During runtime, the compiler decides to invoke the best sutiable method to be invoked

**Conclusion**  
Here we learned the basics of generic method overloading.