**Anurag Soni**

**.Net Assessment**

Guidelines:

*Use each question as an opportunity to showcase your .NET skills. Describe in the way you think can best explain the question and concept behind it. Use diagram or code snippet examples wherever possible. Don’t spend more than 45 minutes on below 10 questions.*

1. Choose the correct statement among the followings?
2. Indexers are location indicators
3. Indexers are used to access class objects
4. Indexer is a form of property and works in the same way as a property
5. All of the mentioned

**Correct Answer:** D

1. Choose the operator/operators which is/are not used to access the [] operator in indexers?
   1. get
   2. set
   3. access
   4. all of the mentioned

**Correct Answer:** C

1. What will be the output of the following C# code snippet?

**class** Program

{

**static** **void** Main(**string**[] args)

{

**int**[] nums = { 1, -2, 3, 0, -4, 5 };

**var** posNums = nums.**Where**(n => n > 0).**Select**(r => r\*2).

OrderByDescending(r=>r);

Console.Write("The positive values in nums: ");

**foreach**(**int** i **in** posNums)

Console.Write(i + " ");

}

}

1. code run successfully prints nothing
2. run time error
3. code run successfully prints multiple of 2
4. compile time error

**Correct Answer:** C

1. What is Predicate and Func ?

**Func** : When you want a delegate for a function that may or may not take parameters parameters but it will always return the result of the provided type. Func<T1,T2,T3,TResult>, here T1,T2,T3 are input parameters and TResult is the output of it.

list.Select(x => x.SomeProperty)

or filtering:list.Where(x => x.SomeValue == someOtherValue)

or key selection: list.Join(otherList, x => x.FirstKey, y => y.SecondKey, ...)

**Predicate** : Predicate is a special kind of Func. It represents a method that contains a set of criteria mostly defined inside an if condition and checks whether the passed parameter meets those criteria or not.It takes one input parameter and returns a boolean - true or false.. In simple words it is wrapper of Func<T,bool>.

1. What will be output of the following program

|  |
| --- |
| public class A  {  static A()  {  Console.WriteLine("Static A Printed");  }  public A()  {  Console.WriteLine("Public A Printed");  }  }  public class B : A  {  static B()  {  Console.WriteLine("Static B Printed");  }  public B()  {  Console.WriteLine("Public B Printed");  }  }  static void Main(string[] args)  {  B objB = new B();  } |
|  |

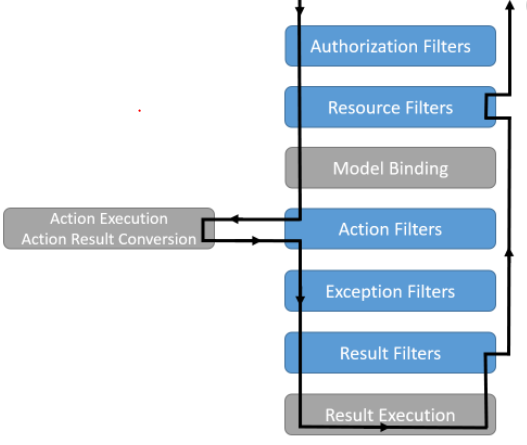
**Output :**

* Static B Printed
* Static A Printed
* Public A Printed
* Public B Printed

1. What is API versioning?

**Answer :** API versioning is the practice of transparently managing changes to your API.

1. What are filters in MVC?



ASP.NET MVC Filter is a custom class where you can write custom logic to execute before or after an action method executes. Filters can be applied to an action method or controller in a declarative or programmatic way. Declarative means by applying a filter attribute to an action method or controller class and programmatic means by implementing a corresponding interface.

|  |  |  |
| --- | --- | --- |
| Filter Type | Interface | Description |
| Authentication | IAuthenticationFilter | These are Runs, before any other filters or the action method. |
| Authorization | IAuthorizationFilter | These Runs first, before any other filters or the action method. |
| Action | IActionFilter | These Runs before and after the action method. |
| Result | IResultFilter | Runs before and after the action result are executed. |
| Exception | IExceptionFilter | Runs only if another filter, the action method, or the action resultthrows an exception. |

1. Differences between Abstract class and Interface

| Sr. No. | Key | Abstract Class | Interface |
| --- | --- | --- | --- |
| 1 | Definition | In terms of standard definition an Abstract class is, conceptually, a class that cannot be instantiated and is usually implemented as a class that has one or more pure virtual (abstract) functions. | On other hand an Interface is a description of what member functions must a class, which inherits this interface, implement. In other words, an interface describes behaviour of the class. |
| 2 | Implementation | As like of other general class design in C# Abstract class also have its own implementation along with its declaration. | On other hand an Interface can only have a signature, not the implementation. While its implementation is being provided by the class which implements it. |
| 3 | Inheritance | As per specification in C# a class can extends only one other class hence multiple inheritance is not achieved by abstract class. | On other hand in case of Interface a class can implements multiple interfaces and hence multiple inheritance is achieved by interface. |
| 4 | Constructor | Like other classes in C# for instantiation abstract class also have constructor which provide an instance of abstract class to access its non-static methods. | On other hand Interface do not have constructor so we can't instantiate an interface directly although its method could get accessed by creating instance of class which implementing it. |
| 5 | Modifiers | As abstract class is most like of other ordinary class in C# so it can contain different types of access modifiers like public, private, protected etc. | On other hand as Interface needs to be get implemented in order to provide its methods implementation by other class so can only contains public access modifier. |
| 6 | Performance | As abstract class have its method as well as their implementations also for its abstract methods implementation it have reference for its implementing class so performance is comparatively faster as compare to that of Interface. | On other hand the performance of interface is slow because it requires time to search actual method in the corresponding class. |

1. What does yield keyword do in C#

The yield keyword is use to do custom stateful iteration over a collection. The yield keyword tells the compiler that the method in which it appears is an iterator block. The yield return statement returns one element at a time. The return type of yield keyword is either IEnumerable or IEnumerator. The yield break statement is used to end the iteration.

yield return <expression>;

yield break;

1. Write a Linq query to get only Jesper Name using Linq

var students = new List<Student>() {

new Student(){ Id = 1, Name="Jesper"},

new Student(){ Id = 2, Name="Steve"},

new Student(){ Id = 3, Name="Jesper"},

new Student(){ Id = 4, Name="Abdul"}

};

Student Class: -

internal class Student

{

public int Id { get; set; }

public string Name { get; set; }

}

**Output**: var name = students.Where(x=>x.Name.ToLower() == "jesper").Select(x=>x.Name).FirstOrDefault();

**.Net Coding Assessment (attempt one of the two exercises)**

*Generic guidelines*

***Should be written in .Net***

*Use Object Oriented Programming approach to solve the problem*

*Use DRY (Don’t Repeat Yourself) Principle and Clean code practices*

*Commit your code to your Git repo.*

*Commits should be incremental with adequate and descriptive comments.*

*Don’t spend more than 1.5 hours to 2 hours of time on coding assessments.*

**Problem Statement1: Promotion Engine**

We need you to implement a simple promotion engine for Checkout process. Our Cart contains a list of single character SKU ids (A, B, C, ...) over which the promotion engine will need to run.

The promotion engine will need to calculate the total order value after applying the 2 promotion types.

Buy "n" items of SKU for fixed price (3A's for 130)

Buy SKU 2 and SKU 2 for fixed price (C+D = 30)

The promotion engine should be modular to allow more promotion types to be added at a later date (For example a future promotion could be x% of a SKU unit price). For this coding exercise you can assume that promotions will be mutually exclusive, that if one promotion is applied, the other promotions will not apply.

**Test Set Up**

Unit Price for SK IDs

A 50

B 30

C 20

D 15

**Active Promotions**

3 of A's for 130

2 of B's for 45

C & D for 30

**Scenario A**

1 \* A = 50

1 \* B = 30

1 \* C = 20

-----------

Total = 100

**Scenario B**

5\* A 130 + 2\*50

5 \* B 45+45+30

1 \* C 20

-----------

Total = 370

**Scenario C**

3 \* A 130

15\* B 45+45+1\*30

1 \* C -

1 \* D 30

-----------

Total = 280

**Problem Statement2. Business Rule Engine**

Imagine you are writing an order processing application for a large company. In the past, this company used a random mixture of manual and ad-hoc automated business practices to handle orders.

They now want to pull all these various was of handling orders together into one whole: your applications.

After a full day of workshops, you have gathered the following set of rules which need to be managed by the new system.

* If the payment is for a physical Product, generate a packing slip for shipping,
* If the payment is for a book, create a duplicate packing slips for the royalty department.
* If the payment is for a membership, activate that membership,
* If the payment is for an upgrade to a membership, apply the upgrade,
* If the payment is for a membership or upgrade, e-mail the owner and inform them of the activation/upgrade.
* If the payment is for the video "Learning to Ski", add free "First Aid" video to the packing slip (the result of a court decision in 1997),
* If the payment is for a physical product or a book, generate a commission payment to the agent.

Design a new system which can handle these rules and yet open to extension to new rules.