The Official C#.NET Alpha-Geek® Quiz

From the Official C#.NET Alpha-Geek® Intelligence Training Series

Rules and Regulations for Taking the Quiz:

1. **TAKE THIS TEST ONLY** if you consider yourself a superior, truly certified C#.NET Alpha Geek. Programmers lacking deep C#.NET expertise or those with a weak self image should not take the quiz. Only those those with TRULY SUPERIOR technical skills should consider taking this exam.

2. **DO NOT TAKE THE TEST ALONE.** This quiz is designed to highlight, showcase, showboat, parade and trumpet the superior skill of the true Alpha-Geek. There is only one Alpha Geek per organization. This quiz is intended to be taken by that superior, #1 technical programmer in your organization. The ideal way to take the quiz is to gather up to 12 other programmers around you as you take this exam. This way as you get all the right answers, you will reinforce your Alpha Geek status in your group.

3. **THE QUIZ IS TRULY SIMPLE TO PASS** if you are truly the best C#.NET programmer on your team. However, even ONE wrong answer means you need to attend a C#.NET training program right away to correct your C#.NET knowledge deficiencies.

---

1. Assuming you have created a Class in your system. Which is an appropriate way of notifying the class that the object is going out of scope?

- ☐ **A.** No notification mechanism is available in .NET. .NET does it's own memory management and therefore is unable to tell us when an object has gone out of scope.
- ☐ **B.** You would need to implement the IDisposable interface in the class and wrap the instance of the object with the "using" block.
- ☐ **C.** You would need to implement the Dispose method and call it manually right before the object went out of scope.
- ☐ **D.** Either B or C would work.

2. What does the Using statement do?

- ☐ **A.** Tells the compiler what namespaces should be used in the current file.
- ☐ **B.** Similar to the "with" statement in VB, allows the programmer to code the properties of an object without having to specify the object in each line.
- ☐ **C.** Allows the dispose method on an IDisposable object to be called automatically.

D. None of the above.

3. Which of the following does the destructor of a class [~classname()] implement in the MSIL?

- **A.** There is no such syntax in C#. A destructor is a C++ construct.
- **B.** It implements the Dispose method off of the IDisposable interface.
- **C.** It implements Finalize() method.
- **D.** None of the answers are correct.

4. What is wrong with the following code snippet?

```
public void test()
{
   FileStream fs =
   new FileStream("c:\\soap.xml",FileMode.CreateNew);
   fs.WriteByte(5);
   fs.Close();
}
```

- **A.** The Dispose method is never called on the FileStream object.
- **B.** The first letter of the method name "test" should be capitalized.
- **C.** WriteByte() is not a method of the FileStream object.
- **D.** None of the above

5. Given the following code:

```
class Person {
   public string FirstName;
   public String LastName;
   Person()
   {
      FirstName="";
      LastName=""
   }
}
class Employee : Person{
   public int ID;
   Employee(){ID=3;}
   }

public void Main()
{
   Person p = new Person();
```

```
    Employee e = (Employee)p;
}
```

What is the value of e.ID?

- **A.** e.ID is 3 (three)
- **B.** e.ID is 0 (zero)
- **C.** e.ID is undefined it will have whatever happens to be in memory at the time.
- **D.** None of the above.

6. Given the following code:

```
public class BaseClass {
    public BaseClass(){ }

    public void foo(){
        System.Windows.Forms.MessageBox.Show("Base foo");
    }

    public void callFoo() {
        foo();
    }
}

public class ChildClass : BaseClass {
    public ChildClass(){ }

    public new void foo() {
        System.Windows.Forms.MessageBox.Show("Child foo");
    }
}
// the following code called in another function.
ChildClass cc = new ChildClass();
cc.callFoo();
```

...What will be displayed in a message box?

- **A.** The code will not compile.
- **B.** "Child foo"
- **C.** "Base foo"
- **D.** None of the above

7. Given the following code:

```
public class BaseClass {
    public BaseClass(){ }
```

```
    public void virtual foo(){
        System.Windows.Forms.MessageBox.Show("Base foo");
    }

    public void callFoo() {
        foo();
    }
}
public class ChildClass : BaseClass {
    public ChildClass(){ }

    public void override foo() {
        System.Windows.Forms.MessageBox.Show("Child foo");
    }
}
// the following code called in another function.
ChildClass cc = new ChildClass();
cc.callFoo();
```

...What will be displayed in a message box?

- **A.** The code will not compile.
- **B.** "Child foo"
- **C.** "Base foo"
- **D.** None of the above

8. Given two interfaces, IMetric and IEnglish, with a Size property defined:
How would you implement these two interfaces in a class in such a way that you could call Size from an IMetric type and get a metric size and call Size from an IEnglish type and get the English size?

- **A.** The interfaces only specify that the Size property will be available in the class, they do not give you control over which implementation will be used. Since there can only be one Size property in the class, you can't return a metric or English specific value.
- **B.**You would have to implement two methods in the class using the following notation:

```
public Size Size : IMetric.Size
{
    get{return m_MetricSize;}
}

public Size Size : IEnglish.Size
{
```

```
    get{return m_EnglishSize;}
}
```

- [ ] **C.** You would have to implement the two methods in the class using the following notation:

```
public Size IMetric.Size
{
    get{return m_MetricSize;}
}
public Size IEnglish.Size
{
    get{return m_EnglishSize;}
}
```

- [ ] **D.** None of the above.

9. What is wrong with the following code?

```
public struct A : IDisposable {
    public int i;
    public int j;
    public Size Size
    {
        return new Size(i,j);
    }
    public A(int I,int J)
    {
        i = I;
        j = J;
    }
}
```

- [ ] **A.** Structures can not inherit from anything.
- [ ] **B.** Structures can not have properties
- [ ] **C.** Structures can not have constructors.
- [ ] **D.** The variable a can not be assigned to be without first initializing it's member variables.

10. What code would we need to implement to have an integer "cast" to a string?

- [ ] **A.**

```
public static implicit operator String(int i)
{
    return i.ToString();
}
```

B.

```
public static explicit operator String(int i)
{
   return i.ToString();
}
```

C.

```
public static operator String(int i)
{
   return i.ToString();
}
```

D. None of the above

Click To Get Answers