1)

```
class УнаследованныйКласс: БазовыйКласс
{
   // Данные-члены и функции-члены
}
```

2)

```
public class MyDerivedClass: MyBaseClass, IInterface1, IInterface2
{
   // и т.д.
}
```

3)

```
public struct MyDerivedStruct: IInterfacel, IInterface2
{
   // и т.д.
}
```

4)

```
class MyClass: Object // наследуется от System.Object
{
   // и т.д.
}
class MyClass // наследуется от System.Object
{
   // и т.д.
}
```

5)

```
class MyClass: object // наследуется от System.Object
{
   // и т.д.
}
```

6)

```
class MyBaseClass
{
  public virtual string VirtualMethod()
  {
     return "Это - виртуальный метод, определенный в
MyBaseClass";
  }
}
```

7)

```
public virtual string ForeName
{
   get { return foreName; }
   set { foreName = value; }
}
private string foreName;
```

8)

```
class MyDerivedClass: MyBaseClass
{
  public override string VirtualMethod ()
  {
     return "Этот переопределенный метод объявлен в
MyDerivedClass";
  }
}
```

9)

```
class HisBaseClass
{
   // разнообразные члены
}
```

10)

```
class MyDerivedClass: HisBaseClass
{
  public int MyGroovyMethod()
  {
     // некая превосходная реализация
     return 0;
  }
}
```

11)

```
class MyDerivedClass: HisBaseClass
{
  public new int MyGroovyMethod()
  {
     // некая превосходная реализация
     return 0;
  }
}
```

12)

```
class CustomerAccount
{
  public virtual decimal CalculatePrice ()
  {
     // реализация
     return 0.0M;
  }
}
class GoldAccount: CustomerAccount
{
  public override decimal CalculatePrice()
  {
     return base.CalculatePrice() * 0.9M;
  }
}
```

13)

```
abstract class Building
{
  public abstract decimal CalculateHeatingCost(); //абстрактный метод
}
```

14)

```
sealed class FinalClass
{
   // и т.д. ,
)
class DerivedClass: FinalClass    //Неверно.Ошибка компиляции.
{
   // и т.д.
}
```

15)

```
class MyClass: MyClassBase
{
  public sealed override void FinalMethod ()
  {
     // и т.д.
  }
}
class DerivedClass: MyClass
{
  public override void FinalMethod() //Неверно. Ошибка
компиляции.
  {
  }
}
```

16)

```
abstract class GenericCustomer
{
    private string name; // прочие методы
}
```

```
class Nevermore60Customer: GenericCustomer
{
   private uint highCostMinutesUsed;
   // прочие методы
}
```

17) GenericCustomer customer = new Nevermore60Customer();

18)

```
public abstract class GenericCustomer
{
   private string name;
   public GenericCustomer()
    :based //эту строку можно пропустить без влияния
          //на скомпилированный код
   {
      name = "<no name>";
   }
}
```

19)

```
public GenericCustomer()
{
   name = "<no name>";
}
```

20)

```
private GenericCustomer()
{
   name = "<no name>";
}
```

21)

‘Wrox.ProCSharp.GenericCustomer.GenericCustomer()’ is inaccessible due to its protection level

'Wrox.ProCSharp.GenericCustomer.GenericCustomer()' не доступен из-за его уровня защиты

22)

```
abstract class GenericCustomer
{
   private string name;
   public GenericCustomer(string name)
   {
      this.name = name;
   }
```

23)

```
class Nevermore60Customer:GenericCustomer
{
   private uint highCostMinutesUsed;
   public Nevermore60Customer(string name):base(name)
   {
   }
```