24)

```
class Nevermore60Customer: GenericCustomer
{
  public Nevermore60Customer(string name, string referrerName)
      : base(name)
  {
     this.referrerName = referrerName;
  }
  private string referrerName;
  private uint highCostMinutesUsed;
```

25)

```
public Nevermore60Customer(string name) : this(name, "<None>")
{
}
```

26)

```
GenericCustomer customer = new Nevermore60Customer("Arabel Jones");
```

27)

```
public Nevermore60Customer(string Name)
    : this(Name, "<None>")
```

28)

Таблица 4.1. Модификаторы видимости

Модификатор	К чему относится	Описание
public	К любым типам или членам	Элемент виден в любом другом коде
protected	К любому члену типа, а также к любому вложенному типу	Элемент видим только любому производному типу
internal	К любым типам или членам	Элемент видим только в пределах включающей его сборки
private	К любому члену типа, а также к любому вложенному типу	Элемент видим только в пределах типа, которому он принадлежит
protected internal	К любому члену типа, а также к любому вложенному типу	Элемент видим только в пределах включающей его сборки, а также в любом коде внутри производного типа

29)

```
public class MyClass
{
    // и т.д.
```

30)

```
public class OuterClass
{
   protected class InnerClass
   {
      // и т.д.
   }
   // и т.д.
}
```

Таблица 4.2. Другие модификаторы

Модификатор	К чему относится	Описание
new	К функциям-членам	Член скрывает унаследованный член с той же сигнатурой
static	Только к классам и функциям-членам	Член не связан с конкретным экземпляром класса
virtual	К любым типам или членам	Член может быть переопределен в классах- наследниках
abstract	Только к функциям-членам	Виртуальный член, определяющий сигнатуру, но не предоставляющий реализации
override	Только к функциям-членам	Член переопределяет унаследованный виртуальный или абстрактный член базового класса
sealed	К классам, методам и свойствам	Для классов означает, что от таких классов нельзя наследовать. Для свойств и методов – член переопределяет унаследованный виртуальный член, но не может быть переопределен ни одним членом производных классов. Должен применяться в сочетании с override
extern	Только к статическим методам [Dllimport]	Член реализован внешне, на другом языке

32)

```
public interface IDisposable
{
   void Dispose() ;
}
```

33)

```
class SomeClass: IDisposable
{
   // Этот класс.ДОЛЖЕН содержать реализацию
   // метода IDisposable.Dispose (), иначе
   // возникнет ошибка компиляции.
   public void Dispose ()
   {
      // реализация метода Dispose()
   }
   // остальная часть класса
}
```

34)

```
namespace Wrox.ProCSharp
{
  public interface IBankAccount
  {
     void PayIn(decimal amount);
     bool Withdraw(decimal amount);
     decimal Balance
     {
        get;
     }
  }
}
```

35)

```
namespace Wrox.ProCSharp.VenusBank
{
  public class SaverAccount: IBankAccount
  {
```

```
private decimal balance;
    public void Payln(decimal amount)
    {
        balance += amount;
    }
    public bool Withdraw(decimal amount)
    {
        if (balance >= amount)
        {
            balance -= amount;
            return true;
        }
        Console.WriteLine("Попытка перевода денег не удалась.");
        return false;
    }
    public decimal Balance
    {
        get
        {
            return balance;
        }
    }
```

```
    public override string ToString()
    {
        return String.Format(
          "Сберегательный Банк Венеры: Баланс = (0,6:C}", balance);
    }
  }
}
```

36) public class SaverAccount: IBankAccount

37)

```
namespace Wrox.ProCSharp.JupiterBank
{
  public class GoldAccount: IBankAccount
  {
     // и т.д.
  }
}
```

38)

```
using System;
using Wrox.ProCSharp;
using Wrox.ProCSharp.VenusBank;
using Wrox.ProCSharp.JupiterBank;
```

39)

```
namespace Wrox.ProCSharp
{
  class MainEntryPoint
  {
    static void Main()
    {
        IBankAccount venusAccount = new SaverAccount();
        IBankAccount jupiterAccount = new GoldAccount();
        venusAccount.PayIn(200);
        venusAccount.Withdraw(100);
        Console.WriteLine(venusAccount.ToString());
        jupiterAccount.PayIn(500);
```

```
        jupiterAccount.Withdraw(600);
        jupiterAccount.Withdraw(100);
        Console.WriteLine(jupiterAccount.ToString());
    }
  }
}
```

40)

C:> BankAccounts

Сберегательный Банк Венеры: Баланс = £100.00

Попытка перевода денег не удалась.

Планетарный Банк Юпитера: Баланс = £400.00

41)

```
IBankAccount!] accounts = new IBankAccount[2];
accounts[0] = new SaverAccount();
accounts [1] =» new GoldAccount();
```

42)

```
accounts[1] = new SomeOtherClass();  //SomeOtherClass не реализует
                                     // IBankAccount: НЕВЕРНО!!
```

43)

Cannot implicitly convert type 'Wrox.ProCSharp.SomeOtherClass' to 'Wrox.ProCSharp.IBankAccount'

Неявное преобразование типа 'Wrox.ProCSharp.SomeOtherClass' в 'Wrox.ProCSharp.
IBankAccount' невозможно

44)

```
namespace Wrox.ProCSharp
{
  public interface ITransferBankAccount: IBankAccount
  {
     bool TransferTo(IBankAccount destination, decimal amount);
  }
}
```

45)

```
public class CurrentAccount: ITransferBankAccount
{
  private decimal balance; public void Payln(decimal amount)
  {
     balance += amount;
  }
  public bool Withdraw(decimal amount)
  {
     if (balance >= amount)
     {
        balance -= amount;
        return true;
     }
     Console.WriteLine("Попытка перевода денег не удалась.");
     return false;
  }
```

```
public decimal Balance
 {
    get
    {
       return balance;
    }
 }
 public bool TransferTo(IBankAccount destination, decimal amount)
 {
    bool result;
    result = Withdraw (amount) ;
    if (result)
    {
       destination.Payln(amount);
    }
    return result;
 }
```

```
 public override string ToString()
 {
    return String.Format(
       "Текущий счет в Банке Юпитера: Баланс = {0,6:C}", balance);
 }
}
```

46)

```
static void Main()
{
   IBankAccount venusAccount = new SaverAccount ();
   ITransferBankAccount jupiterAccount = new CurrentAccount();
   venusAccount.Payln(200); jupiterAccount.Payln(500);
   jupiterAccount.TransferTo(venusAccount, 100);
   Console.WriteLine(venusAccount.ToString0) ;
   Console.WriteLine(jupiterAccount.ToString() ) ;
}
```

47)

C:> CurrentAccount

Сберегательный Банк Венеры: Баланс = £300.00

Текущий счет в Банке Юпитера: Баланс = £400.00