Лабораторная работа № 8
Применение конструкторов
В лабораторной работе используется Microsft Visual Studio 2008

Упражнение 1 – Добавление методов с параметрами в класс

В этом упражнении необходимо внести изменения в программу, которую Вы писали в предыдущей лабораторной работе (Откройте файлы BankAccount.cs, AccountType.cs и CreateAccount.cs). В класс BankAccount надо добавить конструктор по умолчанию и еще три конструктора с различной комбинацией параметров.

1. Откройте Visual Studio 2008.
2. В меню File выберите New, затем выберите Project.
3. Из списка шаблонов выберите Console Application. Впишите название проекта, например, Constructors.
4. Изучите программу в файле BankAccount.cs. В файле дано описание класса BankAccount.
5. Отредактируйте класс BankAccount.
6. Удалите метод Populate.
7. Добавьте конструктор по умолчанию. Назовите его BankAccount. Он должен иметь модификатор public, не иметь параметров и не возвращать тип. Тело конструктора должно генерировать номер счета с помощью метода NextNumber, применить AccountType.Checking для установления типа счета accType и инициализацировать баланс счета accBal в 0. Перейдите к созданию оставшихся конструкторов.
8. Создайте еще один конструктор BankAccount. У этого конструктор должен быть один параметр aType, определенный как AccountType. Этот конструктор должен генерировать номер счета с помощью метода NextNumber (как в предыдущем конструкторе), установить accType в aType и инициализацией баланса счета accBal в 0 (как в предыдущем конструкторе).
9. Создайте следующий конструктор BankAccount. У этого конструктора должен быть один параметр aBal, имеющий тип decimal. Этот конструктор должен генерировать номер счета с помощью метода NextNumber, применить AccountType.Checking для установления типа счета accType и установить accBal в aBal.
10. Создайте последний конструктор BankAccount. У этого конструктора должно быть два параметра aType определенный как AccountType и aBal, имеющий тип decimal. Этот конструктор должен генерировать номер счета с помощью метода NextNumber, установить accType в aType и установить accBal в aBal.
11. Проверьте работу конструкторов. В методе Main класса CreateAccount для BankAccount определите 4 переменных. Назовите их acc1, acc2, acc3, acc4.
12. Инициализируйте acc1 с помощью конструктора по умолчанию.
13. Присвойте значение acc2 с помощью конструктора, который имеет только один параметр, определенный как AccountType. Установите тип acc2 как AccountType.Deposit.
14. Присвойте значение acc3 с помощью конструктора, который имеет один параметр баланс типа decimal. Установите значение равным 100.
15. Присвойте значение acc4 с помощью конструктора, который имеет два параметра. Установите тип acc4 как AccountType.Deposit и значение для баланса 500.
16. Используйте метод Write 4 раза для отображения значения каждого счета.

17. Откомпилируйте программу. Откорректируйте ошибки и сохраните результат. Покажите результат преподавателю. Полученный листинг упражнения 1 сохраните для отчета.

Упражнение 2 – Преобразование символов файла в верхний регистр

В этом упражнении необходимо добавить новый класс BankTransaction.

1. Откройте Visual Studio 2008, если она не была открыта.
2. В проект Constructors добавьте новый класс BankTransaction.
3. Удалите команду namespace вместе с открывающейся фигурной скобкой. Удалите закрывающую фигурную скобку в конце модуля.
4. Добавьте переменную amount и определите ее как private readonly decimal.
5. Аналогично добавьте переменную when и определите ее как private readonly DateTime.
6. Добавьте два метода доступа Amount и When. Эти два метода должны возвращать значения экземпляров переменных amount и when соответственно.
7. Определите конструктор для класса BankTransaction как public. У него должен быть параметр tranAmount, определенный как decimal. Этот параметр используется для передачи значения переменной amount. В конструкторе инициализируйте when как DateTime.Now.
8. Откомпилируйте программу. Откорректируйте ошибки.
9. Перейдите к созданию транзакции. Транзакция должна создаваться в классе BankAccount и сохраняться в очереди всякий раз, когда вызываются методы Deposit и Withdraw. В модуле BankAccount добавьте using System.Collections.
10. Добавьте в классе BankAccount переменную tranQueue, объявив ее как
    private Queue tranQueue = new Queue();
11. В методе Deposit перед тем как возвратить значение accBal создайте транзакцию, используя amount в качестве параметра и добавьте ее в очередь с помощью метода Enqueue.
12. В методе Withdraw создайте транзакцию и добавьте ее в tranQueue с помощью метода Enqueue.
13. Проверьте работу программы. Для этого добавьте в класс BankAccount метод Transactions как public. Метод должен иметь тип Queue и возвращать tranQueue.
14. В классе CreateAccount модифицируйте метод Write так, чтобы отображалась информация о транзакциях для каждого счета. Очереди применяют интерфейс IEnumerable, что означает возможность применения конструкции foreach. В теле цикла foreach добавьте вывод даты и времени, а также amount для каждой транзакции, используя методы Amount и When.
15. Добавьте в метод Main выражения, которые кладут деньги на счет и снимают с него. Сделайте это для всех четырех счетов acc1, acc2, acc3, acc4.
16. Проверьте работу программы. Откомпилируйте программу. Откорректируйте ошибки и сохраните результат. Покажите результат преподавателю. Полученный листинг упражнения 2 сохраните для отчета.

Упражнение 3 – Создание деструктора

В этом упражнении необходимо добавить в программу деструктор.

1. Откройте Visual Studio 2008, если она не была открыта.
2. Откройте проект, который Вы создали в предыдуих упражнениях.
3. Добавьте в классе BankTransaction команду using System.IO;.
4. В класс BankTransaction добавьте деструктор ~BankTransaction().

5. В теле деструктора ~BankTransaction() определите несколько выражений. Код должен выглядеть так:

```
~BankTransaction()
        {
                StreamWriter swFile = File.AppendText("Transactions.Dat");
                swFile.WriteLine("Date/Time: {0}\tAmount: {1}", when, amount);
                swFile.Close();
                GC.SuppressFinalize(this);
        }
```

6. Откомпилируйте программу. Откорректируйте ошибки.
7. Проверьте работу деструктора. Для этого внесите изменения в класс CreateAccount. Удалите (закомментируйте) определения переменных acc1, acc2, acc3, acc4 и оставшийся от предыдущего упражнения код. Определите переменную acc1 как BankAccount. Примените методы Deposit и Withdraw с разными параметрами. Выведите значение acc1 с помощью метода Write на консоль.
8. Откомпилируйте программу. Откорректируйте ошибки и сохраните результат.
9. проверьте, что в папке bin\debug создается файл Transactions.Dat. Откройте файл с помоью блокнота. Полученный листинг упражнения 3 сохраните для отчета.