Лабораторная работа № 6
Создание и использование классов
В лабораторной работе используется Microsft Visual Studio 2008

Упражнение 1 – Создание классов

В этом упражнении необходимо преобразовать структуру, описывающую банковский счет в класс. В этом упражнении изучаются механизмы инкапсуляции.

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, *FileDetails*.
4. Изучите программу в файле BankAccount.cs. BankAccount имеет тип *struct*.
5. Откомпилируйте программу и запустите ее. Программа предложит ввести номер счета и баланс, затем еще раз тоже самое для другого счета.
6. Модифицируйте программу BankAccount.cs, превратив структуру в класс. Откомпилируйте программу. Вы получите ошибку.
7. Откройте файл CreateAccount.cs и посмотрите определение класса *CreateAccount*. Класс выглядит так:

```
class CreateAccount
{
...
static BankAccount NewBankAccount( )
{
BankAccount created;
...
created.accNo = number; // Error here
...
}
...
}
```

Назначение created.accNo компилируется без ошибок, пока *BankAccount* был структурой. Надо изменить объявление переменной *created* (в структуре переменная хранилась в стеке и имела тип *value*, а теперь переменную надо объявить как ссылочный тип).

8. Измените объявление переменной *created* и создайте новый объект *BankAccount*.
9. Откомпилируйте программу. Откорректируйте ошибки и сохраните результат. Убедитесь, что данные вводятся корректно.
10. Выполните инкапсуляцию класса *BankAccount*. Все члены класса *BankAccount* имеют модификатор *public*. Поменяйте его на *private*.
11. Откомпилируйте программу. Вы получите ошибку.
12. Вам нужно написать нестатический *public* метод в *BankAccount*. Назовите его *Populate*. Это метод должен быть *void* и иметь два параметра: *number* и *balance*. Первый должен обозначать номер счета и иметь тип *long*, а второй – баланс и иметь тип *decimal*. В теле метода необходимо назначить параметр с типом *long* полю *accNo*, а параметр с типом *decimal* – полю *accBal*. Полю *accType* надо установить *AccountType.Checking*.
13. В файле BankAccount.cs закомментируйте назначения переменной *created* в методе *NewBankAccount*. Добавьте выражение, которое вызывает метод *Populate* и передайте *number* и *balance* как аргументы.

14. Откомпилируйте программу. Вы получите ошибки. Это происходит из-за того, что метод *Write* пытается непосредственно обратиться к полям *BankAccount*, объявленным с модификаторами *private*.
15. Добавьте в *BankAccount* три метода как *public*. Эти методы должны возвращать значения трех полей. Назовите методы *Number, Balance* и *Type*. Методы должны быть объявлены как *public*, не иметь параметров и возвращать данные типа *long*, *decimal* и *string* соответственно.
16. Замените в методе *Write* в *CreateAccount* три выражения, которые пытаются непосредственно обращаться к полям *BankAccount*, объявленным private. Вместо этого надо вызвать три метода, которые Вы создали в предыдущем пункте.
17. Откомпилируйте программу. Откорректируйте ошибки и сохраните результат. Убедитесь, что данные вводятся корректно. Покажите результат преподавателю. Полученный листинг упражнения 1 сохраните для отчета.

Упражнение 2 – Использование методов со ссылочными параметрами

В этом упражнении необходимо внести изменения в программу, в которую Вы писали в предыдущем упражнении.

1. Откройте в Visual Studio 2008 тот проект, с которым Вы работали в предыдущем упражнении.
2. В классе *BankAccount* добавьте *nextAccNo* как *private static long*.
3. Добавьте метод *NextNumber* как public static в класс *BankAccount*. Метод не имеет параметров и должен возвращать тип *long*. Метод должен возвращать значение поля nextAccNo, увеличенное на 1.
4. Закомментируйте в файле *CreateAccount* строчки

```
//Console.Write("Enter the account number   : ");
    //long number = long.Parse(Console.ReadLine());
```

5. Инициализируйте *number* как результат работы метода *NextNumber()*.
6. Откомпилируйте программу. Откорректируйте ошибки и сохраните результат. Убедитесь, что данные вводятся корректно.
7. Явно инициализируйте значение поля *nextAccNo* равным 123.
8. Откомпилируйте программу. Откорректируйте ошибки и сохраните результат. Убедитесь, что данные вводятся корректно. Проверьте, что два созданных счета имеют номера 123и 124. Покажите результат преподавателю.
9. Выполните дальнейшую инкапсуляцию класса *BankAccount*. Измените метод P*opulate*. Оставьте только один параметр *decimal balance*. Внутри метода сделайте назначение полю *accNo* с помощью статического метода *NextNumber*.
10. Измените модификатор метода *NextNumber* на *private*.
11. Закомментируйте объявление и инициализацию *number* в методе *NewBankAccount*.
12. Откомпилируйте программу. Откорректируйте ошибки и сохраните результат. Убедитесь, что данные вводятся корректно. Покажите результат преподавателю. Полученный листинг упражнения 2 сохраните для отчета.

Упражнение 3 – Добавление методов Withdraw и Deposit

В этом упражнении необходимо добавить два метода *Withdraw* и *Deposit*.

1. Откройте в Visual Studio 2008 тот проект, с которым Вы работали в предыдущем упражнении.

2. Добавьте метод Deposit как public в класс *BankAccount*. Метод должен возвращать значение типа decimal и иметь параметр amount тоже типа decimal. Этот параметр добавляется к балансу счета (переменная *accBal*).

3. Добавьте метод *TestDeposit* как *public static void* в класс *CreateAccount*. Метод должен иметь параметр типа *BankAccount*. Назовите его *acc*. Метод должен включать подсказку пользователю, чтобы он ввел «Enter amount to deposit: ». Введенное пользователем значение должно быть прочитано и преобразовано в десятичное значение, которое присваивается переменной *amount*. Затем должен вызываться метод *Deposit* с параметром *amount* в качестве аргумента.

4. Добавьте в метод *Main* вызов метода *TestDeposit* с параметром *berts*. Добавьте метод *Write* с параметром *berts*, чтобы отобразить счет после размещения депозита. Напишите аналогичный код для параметра *freds*.

5. Откомпилируйте программу. Откорректируйте ошибки и сохраните результат. Убедитесь, что данные вводятся корректно.

6. Добавьте метод *Withdraw* как public в класс *BankAccount*. Метод должен возвращать значение типа *bool* и иметь параметр *amount* типа *decimal*.

7. Добавьте в метод *Main* вызов метода *Test Withdraw*. Используйте метод *Write* для отображения информации о счете.

8. Откомпилируйте программу. Откорректируйте ошибки и сохраните результат. Убедитесь, что данные вводятся корректно. Покажите результат преподавателю. Полученный листинг упражнения 3 сохраните для отчета.