УДК 004.438
T.M. Saliy, Candidate of Pedagogical Science
Innovative University of Eurasia (Pavlodar)
E-mail: toma_sal@mail.ru

Two strategies of interface implementation in the C# programming language

Annotation. *The article describes two strategies of interface implementation. The first one, additional object properties of a class are set. The class inheriting interface and implementing its methods can definitely fulfill them. The second strategy of implementation is in closing methods of interface (making them private), specified the method name by the interface name.*
Key words: *interface, programming, methods, software code, classes, object-oriented programming language.*

C# is a new software programming language from Microsoft. The goal of design is to develop component-oriented language for a new.NET framework. C# is a totally object-oriented language, where even types, integrated into language are represented by the classes [1].

Interface is a totally abstracted class, all the methods of which are abstract. Interface methods are declared with no indicating access modifiers (default public). The class, inheriting interface shall implement all interface methods. There is no complete multiple inheritance in the C# language. In order to harmonize this blank, multiple inheritance of interfaces is acceptable.

Let’s examine two strategies of interface implementation and describe a particular interface, defining additional properties of the class objects:

```
public interface IProps{
  void Prop1(string s);
  void Prop2 (string name, int val);
}
```

The class, inheriting interface and implementing its methods, can definitely implement them, declaring the relevant methods of the class opened (public).

Another strategy of realization is in making some interface methods closed (private), specified the method name by interface name:

```
public class ClainP:IProps{
  public ClainP(){ }
  void IProps.Prop1(string s)   {
    Console.WriteLine(s);
  }
  void IProps.Prop2(string name, int val)   {
    Console.WriteLine("name = {0}, val ={1}", name, val);
  }
}
```

Now let’s examine how to get an access to private methods. There are two ways to get access to private methods:

-Wrap-around. Public method being a wrap of private method is performed.

-Casting. The object of IProps interface class is created, received by transformation (casting) of ClainP object of the source class. Private interface methods are available for this object.

Example of private methods wrapping in the ClainP class:

```
public void MyProp1(string s){
  ((IProps)this).Prop1(s);
}
public void MyProp2(string s, int x){
  ((IProps)this).Prop2(s, x);
}
```

The methods are renamed and have other names, which will be known for clients of the class. Casting had to be used in the wrap for a call of private method, resulting this object to IProps interface class [2].

Let’s examine the second way of getting an access to private methods – *transformation (casting) to a class of interface.* Creating interface object in the normal way using operation `new` is impossible, but the object of interface class can be declared and be connected to the real object by upcast (casting) of inheritance object to a class of interface.

```
public void TestClainIProps(){
  Console.WriteLine("Object of ClainP class causes public methods!");
  ClainP  clain = new Clain();
  clain.Prop1("property of the method 1");
  clain.Prop2("Vladimir", 44);
  Console.WriteLine("Object of ClainP class causes public methods!");
  IProps ip = (IProps)clain;
  ip.Prop1("interface: property");
  ip.Prop2 ("interface: property",77);
}
```

With multiple inheritance of interfaces the problems also occur, but their solution is become easier. Let’s examine two main problems: name conflict and inheritance from the common ancestor.

The problem of name conflict occurs when two or more interfaces have the methods with the same names and signature. Two strategies are possible here: gluing of methods and renaming. The strategy of gluing is used when the class – descendent of interfaces – supposes that different interfaces set one and the same method, the unified implementation of which and shall be provided with the descendent. In this case the descendent creates the only public implementation.

Another strategy emanates from the various implementations of different interfaces. In this case conflicting methods should be renamed. To do so, it is enough to implement methods of different interfaces as private, and after open them renaming.

Example of two interfaces, having methods with equal signature and the class – a descendent of these interfaces, applying different strategies of implementation for the conflicting methods.

Example of the name conflict in interfaces:

```
public interface IProps{
  void Prop1(string s);
  void Prop2 (string name, int val);
  void Prop3();
}
public interface IPropsOne{
  void Prop1(string s);
  void Prop2 (int val);
  void Prop3();
}
public class ClainTwo:IProps,IPropsOne {
  // gluing of methods of two interfaces
  public void Prop1 (string s)   {  Console.WriteLine(s);   }
  // restart of methods of two interfaces
  public void Prop2(string s, int x)  {  Console.WriteLine(s + x);  }
  public void Prop2 (int x)   {    Console.WriteLine(x);   }
// private implementation and renaming of methods of two interfaces
voidIProps.Prop3()   {
Console.WriteLine("Method 3 of interface 1");
  }
voidIPropsOne.Prop3()   {
Console.WriteLine("Method 3 of interface 2");
}
  public void Prop3FromInterface1()   { ((IProps)this).Prop3();   }
  public void Prop3FromInterface2()   { ((IPropsOne)this).Prop3(); }
}
public void TestTwoInterfaces(){
  ClainTwo claintwo = new ClainTwo();
claintwo.Prop1("Gluing of a property of two methods");
claintwo.Prop2("restart.: ",99);
  claintwo.Prop2(9999);
  claintwo.Prop3FromInterface1();
  claintwo.Prop3FromInterface2();
  Console.WriteLine("Interface method requests methods of interface 1!");
  IProps ip1 = (IProps)claintwo;
  ip1.Prop1("interface IProps: property 1");
ip1.Prop3();
```

```
  Console.WriteLine("Interface object requests methods of interface 2!");
IPropsOne ip2 = (IPropsOne)claintwo;
  ip2.Prop1("interface IPropsOne: property1");
ip2.Prop3();
}
```

The second problem with multiple inheritances of interfaces is inheritance from the common ancestor. For interfaces the situation of repeated inheritance is likely, since interface, as any other class, can be a descendant of another interface. Since only signature and implementation are inherited at interfaces, the problem of doubling inheritance is resolved into the problem of doubling inheritance of the name conflict.

There is also treatment of exceptional situations. The C# language inherited profile of the C++ language exceptions, adding its amendments [3]. Let's examine the profile in details:

```
try {...}
catch (T1 e1) {...}
...
catch(Tkek) {...}
finally {...}
```

The program lines of the module, where origin of exceptional situation are possible, shall be made guarded, included into the block with the key word `try`. After *try*-block the *catch*-blocks, called processor-blocks of exceptional situations, can succeed. There can be several, but can be missing. *Finally-block*, a block of finalization, which also can be missing, completes the sequence. All this construction can be added to the structure. The structure of try-block can have another try-catch-finally construction

Throw [expression] – generates exception and creates a class object, being the descendant of *Exception class.* Usually this expression is `new`, creating the object of Exception class or its descendant [4].

The *try* block could cover the resources: files are open, some devices are engaged. *Finally-block* releases the resources, occupied by the try-block. If it is included, it is always fulfilled, exactly after completion of the *try-block* operation, no matter how it was completed.

REFERENCES

1 Троелсен Э. C# и платформа .NET. – СПб., 2005г. – 796 с.
2 Арчер Т. Основы C#. – М.: Русская редакция, 2011.
3 Лабор В.В., Си Шарп. Создание приложений для Windows. – Мн.: Харвест, 2013. – 384 с.
4 Петцольд Ч. Программирование с использованием Microsoft Windows Forms. – СПб., 2006. – 410 с.

REFERENCES

1 Troelsen A. C# i platforma. NET. - CPb., 2005 - 796 s.
2 Archer T. Osnovy C#. - M. Russkaya redaktsiya, 2011.
3 Labor V.V., Si Sharp. Sozdanie prilozhenii dlya Windows. – Mn.: Kharvest, 2013 – 384 s.
4 Pettsold Ch. Programmirovanie s ispolzovaniem Microsoft Windows Forms. – SPb., 2006. – 410 s.

ТҮЙІН

Т.М. Салий*, педагогика ғылымдарының кандидаты*
Инновациялық Еуразия университеті (Павлодар қ.)

C# бағдарламалау тілінде интерфейсті жүзеге асырудың екі стратегиясы

Мақалада интерфейсті жүзеге асырудың екі стратегиясы сипатталады. Бірінші стратегия класс объектілерінің қосымша қасиеттерін береді. Интерфейстен кейін жүретін және оның әдістерін іске асыратын класс оларды анық жүзеге асыра алады. Іске асырудың басқа стратегиясы әдістің атын интерфейстің атымен анықтап, интерфейстің кейбір әдістерін жабық (private) ету болып табылады.

Түйін сөздер: *интерфейс, бағдарламалау, әдістер, бағдарламалық код, кластар, бағдарламалаудың объекті-бағытталған тілі*

РЕЗЮМЕ

Т.М. Салий, *кандидат педагогических наук*
Инновационный Евразийский университет (Павлодар)

Две стратегии реализации интерфейса в языке программирования C#

В статье описываются две стратегии реализации интерфейса. Первая стратегия задает дополнительные свойства объектов класса. Класс, наследующий интерфейс и реализующий его методы, может реализовать их явно. Другая стратегия реализации состоит в том, чтобы некоторые методы интерфейса сделать закрытыми (private), уточнив имя метода именем интерфейса.

Ключевые слова: *интерфейс, программирование, методы, программный код, классы, объектно-ориентированный язык программирования.*

УДК 378.14

T.M. Saliy, Candidate of Pedagogical Science
Innovative University of Eurasia (Pavlodar),
I.M. Makarikhina, Candidate of Pedagogical Science
Pavlodar State University, named after S. Toraigyrov (Pavlodar)
E-mail: toma_sal@mail.ru, michmacha@mail.ru

Using new generation e-books in high school educational process, including faculty of teachers' development

Annotation. *The article describes that the effective management of the learning process new textbooks using need to create a model of teacher action. The created model explicitly takes into account the objectives, methods, learning outcomes. With its help solve the problem of the knowledge of the student. And the problem of managing the cognitive activity is settled.*

Key words: *educating, electronic textbooks, educational process, higher education, control of knowledge*

The current stage of Kazakhstan and Russian society development is characterized by the educational possess of information technologies. The transition to the education multi-level system requires higher education institutions providing such training of highly qualified staff, which could combine the ability to solve actual scientific, technical and socio-economic problems.

The most important moment in modern education today is how the knowledge-based acquire skills, to transform them and develop new knowledge in learners' professional activities. Modern Kazakh and Russian education's difficult task is to provide humanistic, technological and fundamental principles integration with modern requirements of information, humanization and fundamental education bases.

Teacher education, as an integral part of modern education system, has the vital role of staffing higher education. The need for changes in teacher education is defined as the external challenges and internal laws of its development, future needs of an individual, society and state.

You can often come across the term "new information technologies" in the scientific and popular literature. This is quite a broad term for a variety of practical applications. The adjective "new" in this case emphasizes radically different approach from the previous direction of technological development. Their introduction is an innovative instrument in the sense that it fundamentally changes the contents of various high schools activities. You can create a well-designed electronic textbook, which will carry only the information from a paper to computer-based, but the technology does not satisfy the basic principles of educational technology and will not solve the theoretical or practical problems that appeared previously in didactics.

Nowadays multimedia technologies can integrate a variety of informational media presentation: texts, static and dynamic graphics, video and audio clips into a single complex. The use of animation, sound and video greatly enhances the assimilation of educational material on the structuring of knowledge and reduces the learners' level of cognitive effort, while reducing the time required for the study of such a problem.

Accordingly, the media are being used successfully in the development of electronic textbooks. New textbook's generation provides the ability to select a desired line of development represented by the plot or situation on the user actions' analysis.

The use of e-textbooks in the classroom activities is an effective means of enhancing learners' cognitive activity, which opens up opportunities for teachers to improve training. Simulation computer programs are studied material more clearly; they can show the experiments with no equipment in high school. In addition to this, learning technology significantly saves time on the classroom activities (information retrieval, control learners' knowledge).

Since modern education is characterized by the active use of information and communication technologies (ICT) and various devices on their basis to ensure: access to the global resources of the Internet, operation of automation systems, use of electronic educational purposes, computer psycho-educational