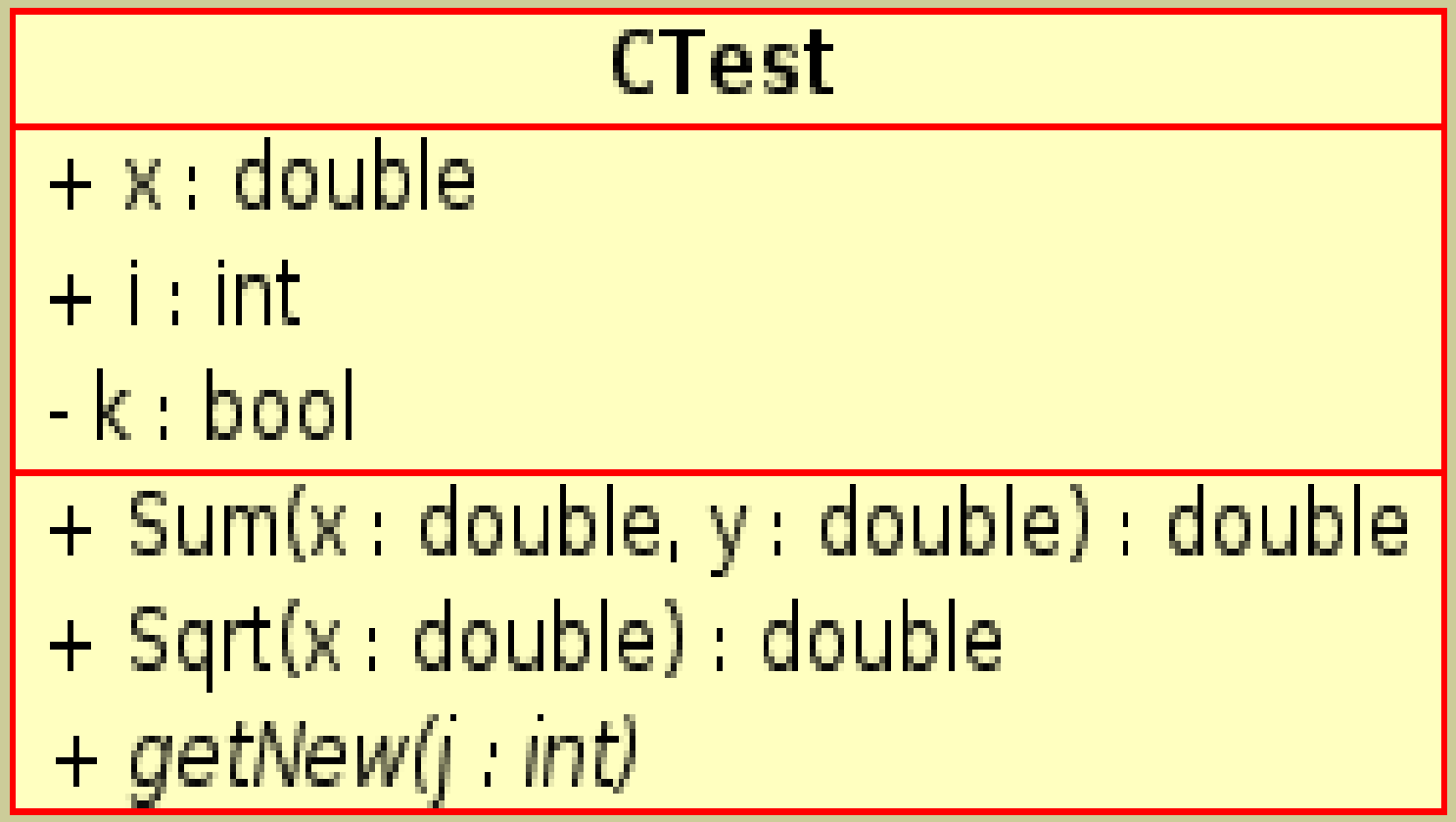
Объектно-ориентированное программирование

Парадигма ООП родилась из потребностей коллективной разработки, повтоного использования кода и сокрытия кода.

Центральное место в ООП занимает понятие *пользовательского типа данных* называемого *классом*, объединяющего под общим именем не только данные, но и функции.

Описание класса в соответствие с *UML:*



Переменные соответствующего типа называются *объектами* **(или *экземплярами*) данного класса. Переменные - члены класса, называются его** *свойствами*, **а *функции* - члены класса, называются его** *методами*.

В дополнение к классическому представлению типов данных, основанных на структурах данных, ООП вносит в него принципиально новые черты, которые можно разделить на следующие группы:

- *инкапсуляция;*
- *наследование;*
- *полиморфизм.*

Инкапсуляция:

Свойства и методы класса могут быть открытыми или закрытыми для пользователя (программиста, который использует этот класс). В конкретных реализациях языков ООП это достигается использованием модификаторов доступа *public*, *private* и (иногда) *protected*. Абстракция данных. Интерфейс.

```
class Test{
public:  int a;
private: int f(int a, int b){
 return a+b;
}
public: double getData(){ return c};
private: double c;
};
```

```
Test A;
int d;
d=A.a;  //OK
d=A.f(2,6); //Error
double d1;
d1=A.c;  //Error
d1=A.getData(); //OK
```

Наследование:

можно расширить возможности класса посредством наследования.

```
class TestA{
public: TestA():c(3.1415){c*=2; } //Конструктор
        ~TestA(){}      //Деструктор
private: double c;
public: double getData({ return c}
         double  x;
protected: int q;
};
class TestB: public TestA{
  int s;
  public: int f(int a){return a*q}
};
```

```
TestB  B;
B.s; //OK
B.x; //OK
B.c; //Error
B.q; //OK
```

Полиморфизм: **поведение, зависящее от контекста.**

Перегрузка (overloading):

```
class Test2{
 ..................
 double f(int a){return a*2.0}
 double f(double a){return a/2.0; }
 ....................................
};
```

```
double c;
c=f(3); // c=6;
c=f(3.0);// c=1.5
```

Переопределение(overriding):

```
class Test3{
public:
   virtual int g(int a){return a*a;}
   virtual int g1(int a)=0;
};
```

```
class Test4:public Test3{
public: virtual int g(int a)
{ return a+4; }
};
```

```
Test3 D3;
Test4 D4;
int c;
c=D3.g(3) // 9
c=D4.g(3) //7
c=((Test3)D4).g(3) //9
```

Модель составных компонентов - *COM-технология*, реализует ООП на двоичном уровне. В настоящее время *COM-объекты* называются *объектами ActiveX*

Пример, *объектная модель Microsoft Office*:

Represents the Microsoft Office Word application.
The Application type exposes the following **members.**

Methods

……

Quit Quits Microsoft Word and optionally saves the open documents. (Inherited from _Application.)

Properties

……

Documents Returns a Documents collection that represents all the open documents.(Inherited from _Application.)

……

Visible Determines if the specified object is visible. This property returns True if the specified object is visible, and False if not. (Inherited from _Application.)

……

Documents Members

Add Returns a Document object that represents a new, empty document added to the collection of open documents

Document Members

SaveAs Saves the specified document with a new name or format. Some of the arguments for this method correspond to the options in the **Save As** dialog box (**File** menu). (Inherited from _Document.)

Document Properties

Content *Returns* a Range object that represents the main document story. (Inherited from _Document.)

Range Properties

Bold Determines if the font or range is formatted as bold.

……..

Font Returns or sets a Font object that represents the character formatting of the specified object.

……

Text Returns or sets the text in the specified range.

……

test1.js

```
var Word, Doc;

Word=new ActiveXObject("Word.Application");

Word.Visible=false;
Doc=Word.Documents.Add();
Doc.Content.Bold=1;
Doc.Content.Font.Size=32;
Doc.Content.Text="Попробуйте автоматизировать работу с
Microsoft Word";
Doc.SaveAs("C:\\temp-work\\Test.doc");
Word.Application.Quit();
```

Windows Scripting Host (***WSH***) – объектно-ориентированный интерфейс автоматизации MS Windows

Фрагмент сценария WSH:

```
var Word;
Word=new ActiveXObject("Word.Application");
……..
var fso;
fso=new ActiveXObject("Scripting.FileSystemObject");
……
 if(fso.FileExists("C:\\dummy.ddd"))  str+="The file has
appeared\n";
  else str+="The file is absent\n";
  Doc.Content.Text=str;
……..
```

Визуальное программирование: