Anglia Ruskin University

Classes and Objects

Primitive types - int, float, etc

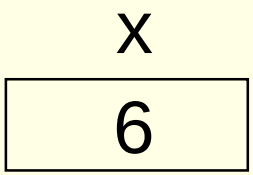
Complex types - classes

A *string* is a collection of alphanumeric characters eg **“Hello world”**

In C# a string can be a character array **char[ ]** but more usually is represented using an *object* of the class **string**

Anglia Ruskin University

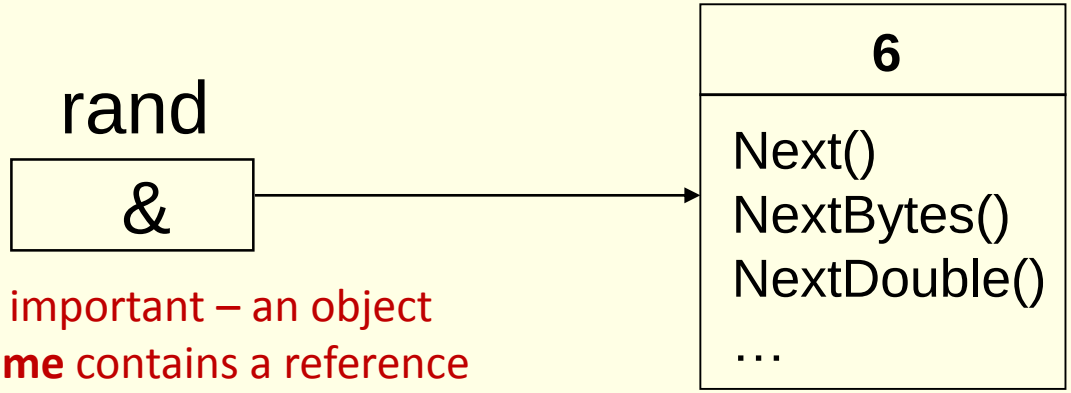
An object is created using the keyword new




```
int   x   =   6;
```

type *variable* *value*

```
Random  rand = new Random( 6 ) ;
```



NB this is important – an object **variable name** contains a reference (address) to the actual object

Anglia Ruskin University

shorthand *just* for creating a string object is>

string fname = “DAVID”;

ConsoleWriteLine(“David in lowercase is “ +
fname.ToLower());

DAVID in lowercase is david

When *classname* is used method is **static**
When *objectname* is used method is **non-static**

Refer to: **class** methods and
object (instance) methods

Anglia Ruskin University

Objects from user-defined classes

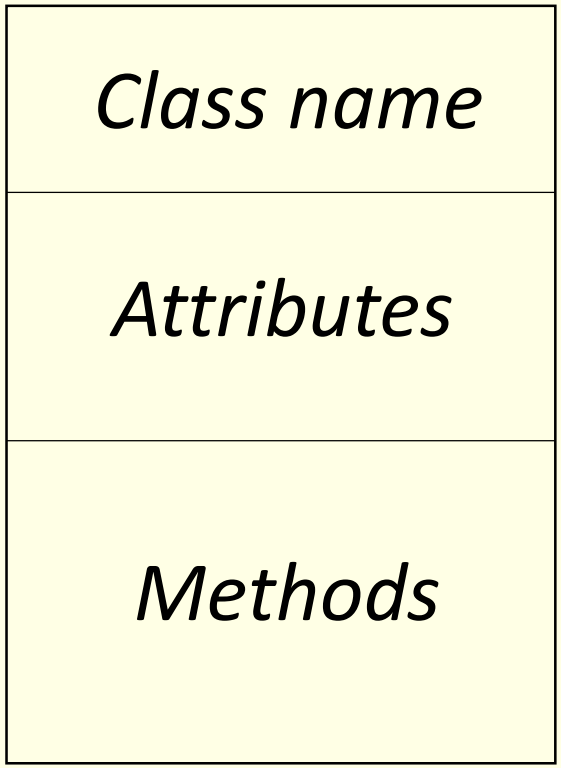
EG: Bank account class

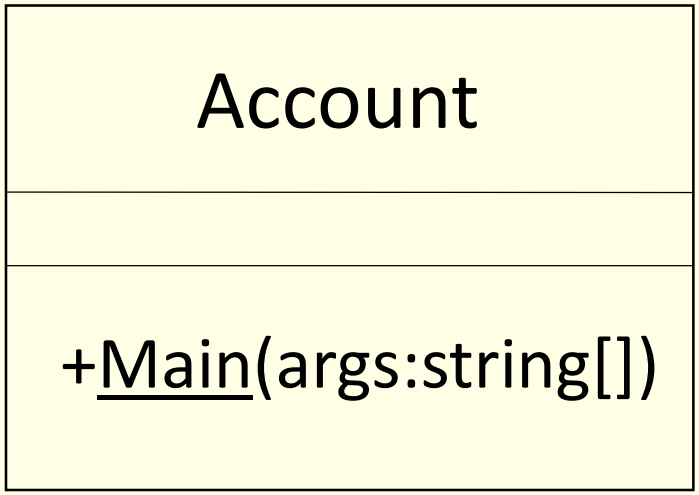
```
class Account
{
   public static void Main( string[ ] args)
   {
      double balance = 400.0;
      Console.WriteLine("Balance is "
                                + balance);
   }
}
```

Balance is 400.0

Anglia Ruskin University

The Class Diagram





Anglia Ruskin University

How can we create *many* bank accounts?

We *could:* Create more classes (Account2.cs, Account3.cs)

But: Managing them would be *difficult* and *wasteful* of resources

Instead we create *objects* of the type/class>

Account acc1 = new Account() ;

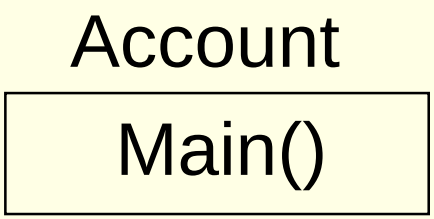
type

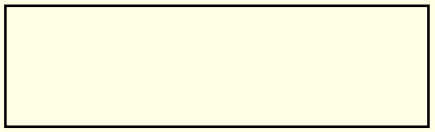
variable

reserves memory for object

Constructor – a special method - same name as class

Anglia Ruskin University





acc1 object has virtually no properties and can't do anything

```
class Account
{
   public static void Main( string[ ] args)
   {
      double balance = 400.0;
      System.out.println("Balance is "
                                  + balance);
      Account acc1 = new Account();
   }
}
```

Anglia Ruskin University

To give object data and allow different start balances - make **balance** non-static and overload **Account()** constructor

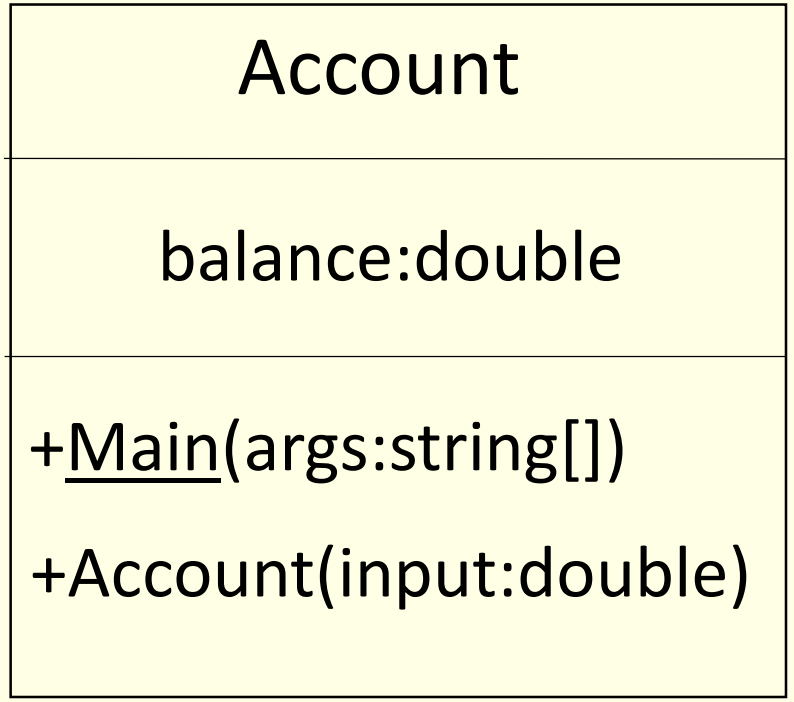
```
class Account
{
  double balance;  // This is an attribute

  public static void Main( string[ ] args )
  {
    Account acc1 = new Account( 400.00  );
    Console.WriteLine("acc1 balance is " + acc1.balance);
  }

  public Account( double input )
  {
    balance = input;
  }
}
```

Anglia Ruskin University

So now can make lots of objects of the one class, each with a unique balance




```
Account acc1 = new Account(400.0);
Account acc2 = new Account(250.0);
Account acc3 = new Account(1100.0);

Console.WriteLine("Balance of acc1 is "+ acc1.balance);
Console.WriteLine("Balance of acc2 is "+ acc2.balance);
Console.WriteLine("Balance of acc3 is "+ acc3.balance);
```

Anglia Ruskin University

Summary

- Classes are types, objects are variables
- Classes occur in memory once and can have their own attributes and methods (static)
- Objects are created using a special method called a constructor
- Objects can occur in memory many times each with their own attributes and methods (non-static)
- Alternative static/non-static terms are :

static or *class*
non-static or *object* or *instance*