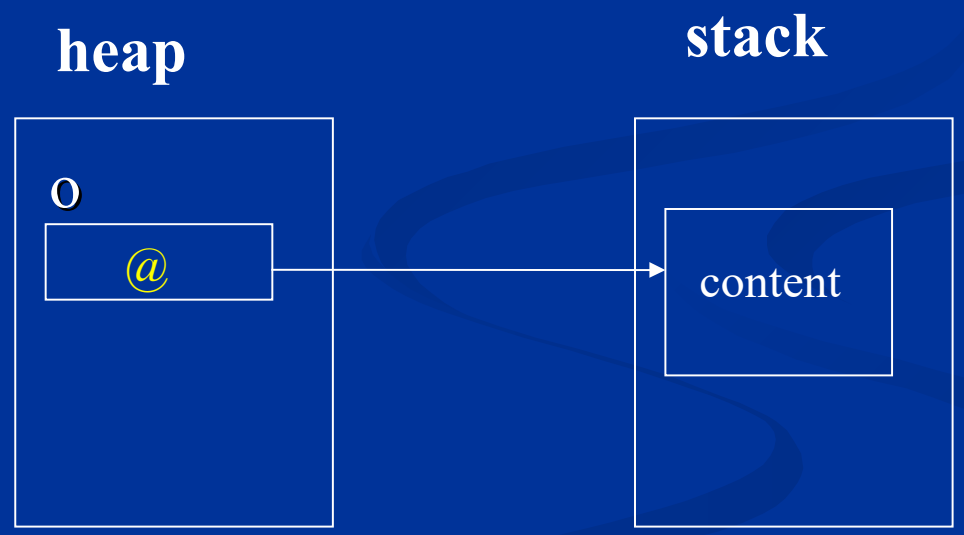
Serialization

objects created in a program reside in RAM through references

object o;



Serialization

objects can be stored / retrieved from disk with serialization/deserialization process

objects are not serializable by default

two methods :

- serialization through a specialized class
- implenting the ISerializable interface (customized serialization)

Serialization through a class

```
class myClass
{
   // attributes, properties and methods
}
```

first step : choose the class members to be serialized

Serialization through a class

properties and methods : not serialized (code)

attributes : may be serialized

add the [Serializable] attribute to the class

add the [NonSerialized] attribute to attributes

Serialization through a class

```
[Serializable]
public class animal
{
   private string _name;    // will get serialized
   private string _specie; // will get serialized

   [NonSerialized]
   private long _number;    // this one won't persist

   public override string ToString()
   {
       return _name+" "+_specie;
   }
}
```

Serialization class

```
using System.IO;
using System.Runtime.Serialization.Formatters.Soap;

SOAP :        Simple Object Access Protocol
              Service Oriented Architecture Protocol
```

"SOAP is a protocol for exchanging XML-based messages over networks, using HTTP/HTTPS" (http://en.wikipedia.org/wiki/SOAP)

Serialization class

```
class SerialTools
{
   public static void Serialize(string filename, object o)
   {}

   public static object Deserialize(string filename)
   {}
}
```

Serialization method

```
public static void Serialize(string filename, object o)
{
   Filestream fs = new Filestream(filename,
   FileMode.Create);

   SoapFormatter soapf = new SoapFormatter();

   soapf.Serialize(fs, o); // for any object

   fs.close();
}
```

Deserialization method

```
public static object Deserialize(string filename)
{
   Filestream fs = new Filestream(filename, FileMode.Open);

   SoapFormatter soapf = new SoapFormatter();

   object o = soapf.Deserialize(fs);

   fs.close();

   return o;
}
```

Using the SerialTools class

```
animal a = new animal();
animal.name = "platypus";     // where name is a
                    // property, not the _name member
animal.specie = "mammal";     // same thing
animal.number = 7;
SerialTools.Serialize("ani.soap",a);
// a : "platypus" "mammal" 7
animal n = (animal)SerialTools.Deserialize("ani.soap");
// n : "platypus" "mammal" 0
```

Serialization

- Use the BinaryFormatter class for more efficient and compact serialization
- Less verbose than XML and SOAP output !

ISerializable interface

```
[Serializable] class X : ISerializable
{
   // attributes

   // attributes

   // properties

   // methods

   public void GetObjectData(SerializationInfo info,
   StreamingContext ctxt)
   {…}
}
```

GetObjectData() is used for Serialization.

GetObjectData() method

add informations or fields into the `info` parameter.

`info` parameter acts as a hashTable (Dictionary)

```
public void GetObjectData(SerializationInfo info,
   StreamingContext ctxt)
   {
       info.AddValue("key1", attribute1);
       …
       info.AddValue("keyN",attributeN);
   }
```

How to Deserialize ?

No method signature from ISerializable interface : a new constructor is written for the class

```
public X(SerializationInfo info, StreamingContext
   ctxt)
{
   attribute1 =
   (type)info.GetValue("key1",typeof(type));
   …
   attributeN =
   (type)info.GetValue("keyN",typeof(type));
}
```

How to Deserialize ?

example : if class X stores a `private long number` attribute :

serialize it with :

```
info.AddValue("numberkey",number);
```

deserialize it with :

```
number
=(long)info.GetValue("numberkey",typeof(long));
```

Serializing objects

now, as for the first method :

open a filestream, create a formatter (Binary, Soap, Xml), and save the object using the Serialize() method.

open a filestream, create a formatter and create the object from the formatter

Serialization and composition

composition : a class contains an object from another class as an attribute.

how does serialization occur ?

a complete application code follows :

```
[Serializable] class Class1
{
  private string _name;

  public Class1(string s)
  {
      _name=s;
  }

  public string name
  {
      get {return _name;}
  }

  public override string ToString()
  {
      return "Class1 "+_name;
  }
}
```

```
[Serializable] class Class2 : ISerializable
{
   private long _number;
   private static long _count;
   private Class1 c;

   public Class2(string s)
   {
       _number = ++_count;
       c = new Class1(s);
   }

   public Class2() : this("default")
   {}

   public override string ToString()
   {
       return c.ToString+"("+_number.ToString()+")";
   }

   …
```

```
[Serializable] class Class2 : ISerializable
{
   public void GetObjectData(SerializationInfo info,
   StreamingContext ctxt)
   {
       info.AddValue("k_num",_number);
       info.AddValue("k_co",_count);
       info.AddValue("k_c1",c);
   }

   public Class2(SerializationInfo info, StreamingContext
   ctxt)
   {
       _number = (long)info.GetValue("k_num",typeof(long));
       _count = (long)info.GetValue("k_co",typeof(long));
       c = (Class1)info.GetValue("k_c1",typeof(Class1));
   }
}
```

```
class test
{
   public static void Main(string[] args)
   {
   Class2 obj1 = new Class2("first object");
   Class2 obj2 = new Class2("second object");

   FileStream fs = new filestream("data.bin",
   FileMode.Create);
   BinaryFormatter bf = new BinaryFormatter();
   bf.Serialize(fs,obj1);
   bf.Serialize(fs,obj2);
   fs.Close();

   fs = new Filestream("data.bin",FileMode.Open);

   Class2 obj3 = (Class2)bf.Deserialize(fs);
   Class2 obj4 = (Class2)bf.deserialize(fs);

   fs.Close();

   Class1 obj_c1 = obj3.get_c; // a read-only property
   }
}
```

Serializing complex objects

- Classes from the class library are often serializable
- arrays are serializable
- serialization and inheritance
- if C inherits from B, both B and C must possess the [Serializable] attribute
- a polymorph array of B (type B[]) can be serialized

last example : inheritance

```
[Serializable]class B {}
[Serializable]class C : B {}
[Serializable]class A : ISerializable
{
  private B[] myarr;

  // fill myarr with B and C objects

  public void GetObjectData(SerializationInfo info,
  StreamingContext c)
  {…}

  public A(SerializationInfo info, StreamingContext c)
  {…}
}
```