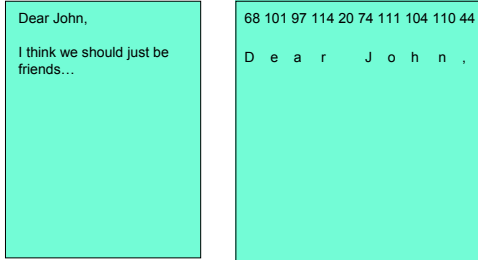
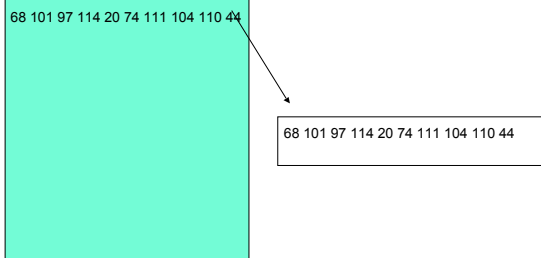


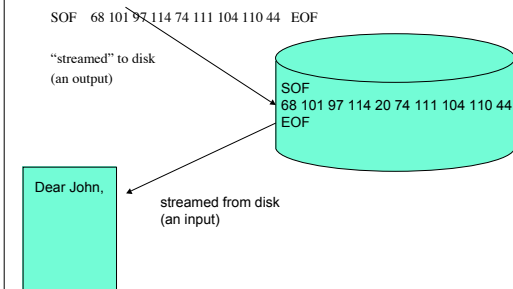
A Document



A serial data "STREAM"



Storage on disk

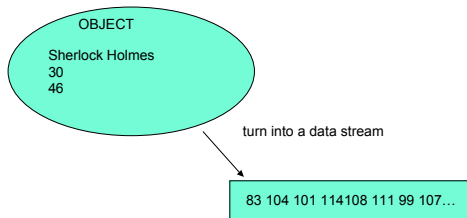


Objects can store info

- Color
- Point
- your own:

```
class infoClass
{
    String Name;
    int Age;
    int secretRealAge;
}
infoClass infoObject = new infoClass ( );
```

Serializable - means “in a row”



Serializable interface

```
class infoClass implements Serializable
{
    String Name;
    int Age;
    int secretRealAge;
}
```

- means that objects of this class can be turned into an orderly dependable data stream

the Serializable Interface

- no methods are involved, so you don't have to write anything
- called a "tagging" interface, because Java just adds variables to your object, to keep the place of *your* variables in a data stream
- Objects that are serializable can be used by "streaming classes": file I/O, network I/O

what the computer does...

```
class infoClass implements Serializable
{
    String Name;
    int placeOfName = 1;
    int Age;
    int placeOfAge = 2;
    int secretRealAge;
    int placeOfRealAge = 3;
}
```

to and from streams

- to a file:

```
FileOutputStream target = new FileOutputStream("filename.ser");
ObjectOutput out = new ObjectOutputStream( target );
out.writeObject( anyObject );
out.close();
```

- from a file:

```
FileInputStream source = new FileInputStream( "filename.ser" );
ObjectInputStream in = new ObjectInputStream( source );
gridTestSer buttons = (gridTestSer) in.readObject();
in.close();
```

using a byte array

- to a byte array

```
ByteArrayOutputStream target = new ByteArrayOutputStream();  
ObjectOutputStream out = new ObjectOutputStream( target );  
out.writeObject(anyObject);  
byte[] buf = target.toByteArray();  
out.close();
```

- from a byte array

```
ByteArrayInputStream source = new ByteArrayInputStream( buf );  
ObjectInputStream in = new ObjectInputStream( source );  
gridTestSer buttons = (gridTestSer) ois.readObject();  
in.close();
```
