```
// Example 9.1 SerializationDemo.java
//
//
// Copyright 2003 Bill Campbell and Ethan Bolker

import java.io.*;
import java.util.*;

// Test of Java serialization.
//
// %> java SerializationDemo
// Wrote: round blue Mon Jan 06 20:14:44 EST 2003
// Read:  round null Mon Jan 06 20:14:44 EST 2003
//
// interesting observations:
//
// %> wc -c SerializationDemo* tmp
//    1207 SerializationDemo$Circle.class
//    1611 SerializationDemo.class
//    3221 SerializationDemo.java
//     271 tmp
//
// the "strings" command finds ascii strings in a file
//
// %> strings SerializationDemo.class | wc
//    45      45     813
//
// %> strings tmp
// SerializationDemo$Circle?
// attributest
//     Ljava/util/Map;L
// selft
//      LSerializationDemo$Circle;xpsr
// java.util.HashMap
// loadFactorI
// thresholdxp?@
// datesr
// java.util.Datehj
// thisq
// namet
// roundxq
//
// %> strings tmp | wc -c
// 180

public class SerializationDemo
{
    public static void main (String[] args)
    {
        Circle circle = new Circle("round");
        write(circle,"tmp");
        System.out.println("Wrote: "+ circle);

        Circle circleCopy = (Circle)read("tmp");
        System.out.println("Read:  " + circleCopy);
    }
```



```

    public static void write (Object obj, String pathname)
    {
        try {
            FileOutputStream f = new FileOutputStream(pathname);
            ObjectOutput s = new ObjectOutputStream(f);
            s.writeObject(obj);
            s.flush();  s.close();  f.close();
        }  catch (Exception e) { e.printStackTrace(); }
    }

    public static Object read(String pathname)
    {
        try {
            Object obj;
            FileInputStream in = new FileInputStream(pathname);
            ObjectInputStream s = new ObjectInputStream(in);
            obj = s.readObject();
            s.close(); in.close();
            return(obj);
        }
        catch (Exception e) {
            e.printStackTrace();
        }
        return(null);
    }

    // To implement the Serializable interface you just _say_ so.
    // You don't have to _do_ anything, although you may choose to
    // overwrite the writeObject() and readObject() methods.

    private static class Circle implements Serializable
    {
        private Circle self;    // a circular reference
        private Map attributes; // saved with its contents

        // Don't bother saving whatever the current color
        //(user settable) happens to be:
        transient private String color;

        Circle(String name)
        {
            attributes = new HashMap();
            attributes.put("this", this); // a circular reference
            attributes.put("name", name);
            attributes.put("date", new Date());
            this.color = "blue";
            self = this;
        }

        public void setColor(String color)
        {
            this.color = color;
        }

        // NOTE: serialization does not call toString-- it calls
```

```
   113          // a smarter serialization method.
   114          public String toString()
   115          {
   116              return( (String)attributes.get("name") + " " +
   117                      color + " " + (Date)attributes.get("date") );
   118          }
   119      }
   120  }
```