10. 5 Binary I/O of Class Objects

- read and write class objects in binary file

- class must be
  - » `import java.io.*`
  - » implement
  - » add **`implements Serializable`** to heading of class definition

```
public class Species implements Serializable
```

- **`methods used:`**

  to **write** object to file:
  `writeObject` method in
  `ObjectOutputStream`

  to **read** object from file:
  `readObject` method in
  `ObjectInputStream`

Marker interface

- public interface **Serializable**

- Serializability of a class is enabled by the class implementing the java.io.Serializable interface. Classes that do not implement this interface will not have any of their state serialized or deserialized. All subtypes of a serializable class are themselves serializable. The serialization interface has no methods or fields and serves only to identify the semantics of being serializable.

Marker interface

- Empty enterface
- Interface that declare no methods or constants
- Intended to mark classes as having certain properties
- Ex) Cloneable interface
  - » can be cloned

```
outputStream = new ObjectOutputStream(
        new FileOutputStream("species.records"));
...
Species oneRecord =
        new Species("Calif. Condor, 27, 0.02);
...
outputStream.writeObject(oneRecord);
```

`ClassIODemo` Excerpts

```
inputStream = new ObjectInputStream(
        new FileInputStream("species.records"));
...
Species readOne = null;
...
readOne =                          inputStream.readObject();
```

`readObject` returns a reference to type `Object` so it must be cast to `Species` before assigning to readOne

Listing 10.9 Species Class Serialized for Binary-File I/O - Species.java

```
import java.io.Serializable;
import java.io.*;

/**
 Class for data on endangered species. This class is serialized.

 */

 // except for these tow words and the import statement, this definition is
the
 //        same as the one in Display

public class Species implements
{
    private String name;
    private int population;
    private double growthRate;
```

Listing 10.10 File I/O of Class Objects - ClassIODemo.java

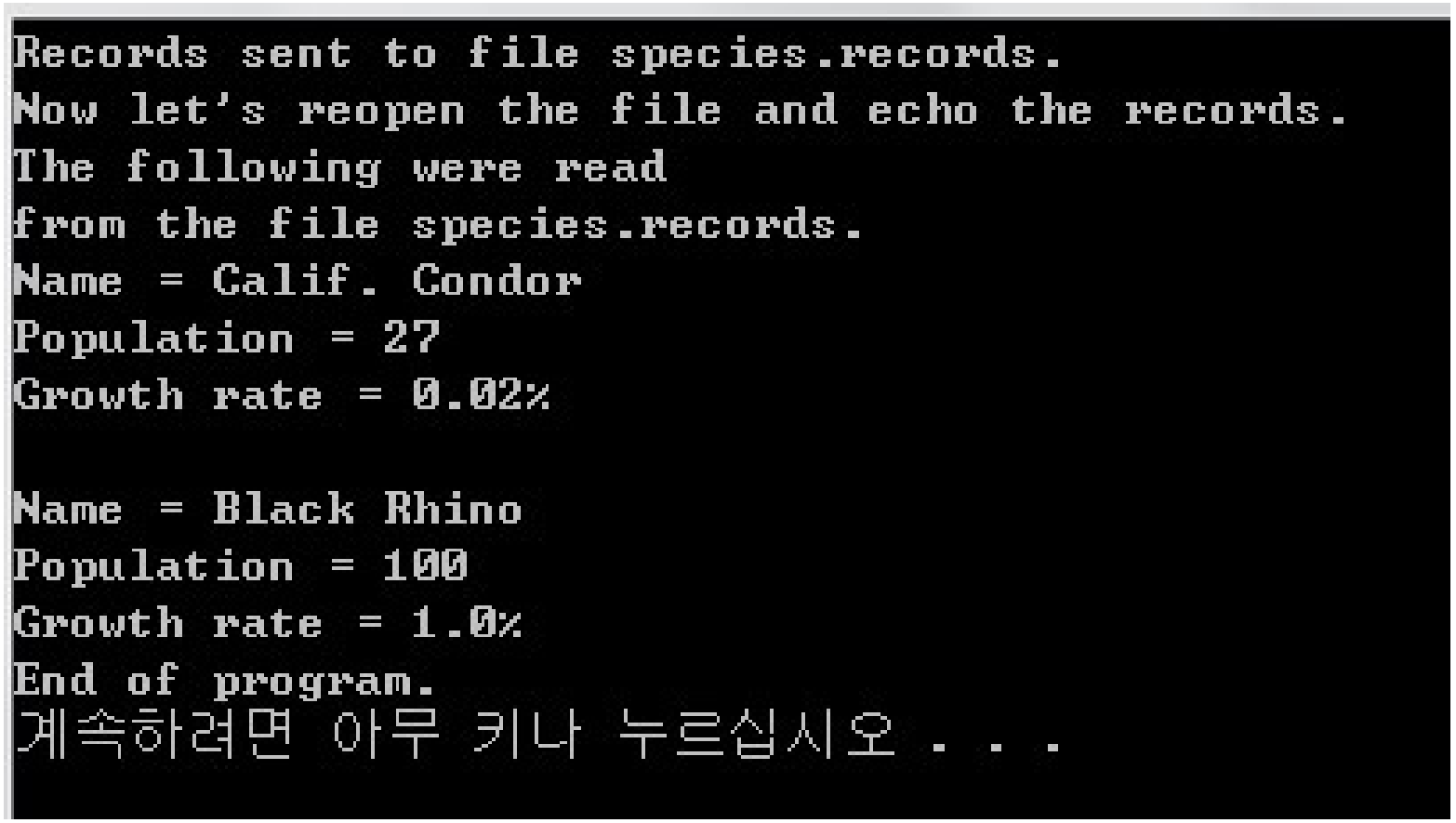
» Illustrate how class objects can be written to and read from a binary file.

```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;

public class ClassObjectIODemo
{
   public static void main(String[] args)
   {
      ObjectOutputStream outputStream = null;
      String fileName = "species.records";
```

```
try
{
    outputStream = new ObjectOutputStream(
                new FileOutputStream(fileName));
}
catch(IOException e)
{
    System.out.println("Error opening output file " +
                    fileName + ".");
    System.exit(0);
}
Species califCondor =
          new Species("Calif. Condor", 27, 0.02);
Species blackRhino =
          new Species("Black Rhino", 100, 1.0);
```

```
try
      {
         outputStream.writeObject(califCondor);
         outputStream.writeObject(blackRhino);
         outputStream.close( );
      }
      catch(IOException e)
      {
         System.out.println("Error writing to file " +
                            fileName + ".");
         System.exit(0);
      }
      System.out.println("Records sent to file " +
                         fileName + ".");
     System.out.println(
           "Now let's reopen the file and echo the records.");
      ObjectInputStream inputStream = null;
      try
      {
         inputStream = new ObjectInputStream(
                       new FileInputStream("species.records"));
      }
      catch(IOException e)
      {
         System.out.println("Error opening input file " +
                            fileName + ".");
         System.exit(0);
      }
```

```
Species readOne = null, readTwo = null;
      try
      {
         readOne = (Species)inputStream.readObject( );
         readTwo = (Species)inputStream.readObject( );
         inputStream.close( );
      }
      catch(Exception e)
      {
         System.out.println("Error reading from file " +
                            fileName + ".");
         System.exit(0);
      }
      System.out.println("The following were read\n" +
                         "from the file " + fileName + ".");
      System.out.println(readOne);
      System.out.println( );
      System.out.println(readTwo);
      System.out.println("End of program.");
   }
}
```


Records sent to file species.records.
Now let's reopen the file and echo the records.
The following were read
from the file species.records.
Name = Calif. Condor
Population = 27
Growth rate = 0.02%
Name = Black Rhino
Population = 100
Growth rate = 1.0%
End of program.
계속하려면 아무 키나 누르십시오 . . .

Array objects in Binary Files

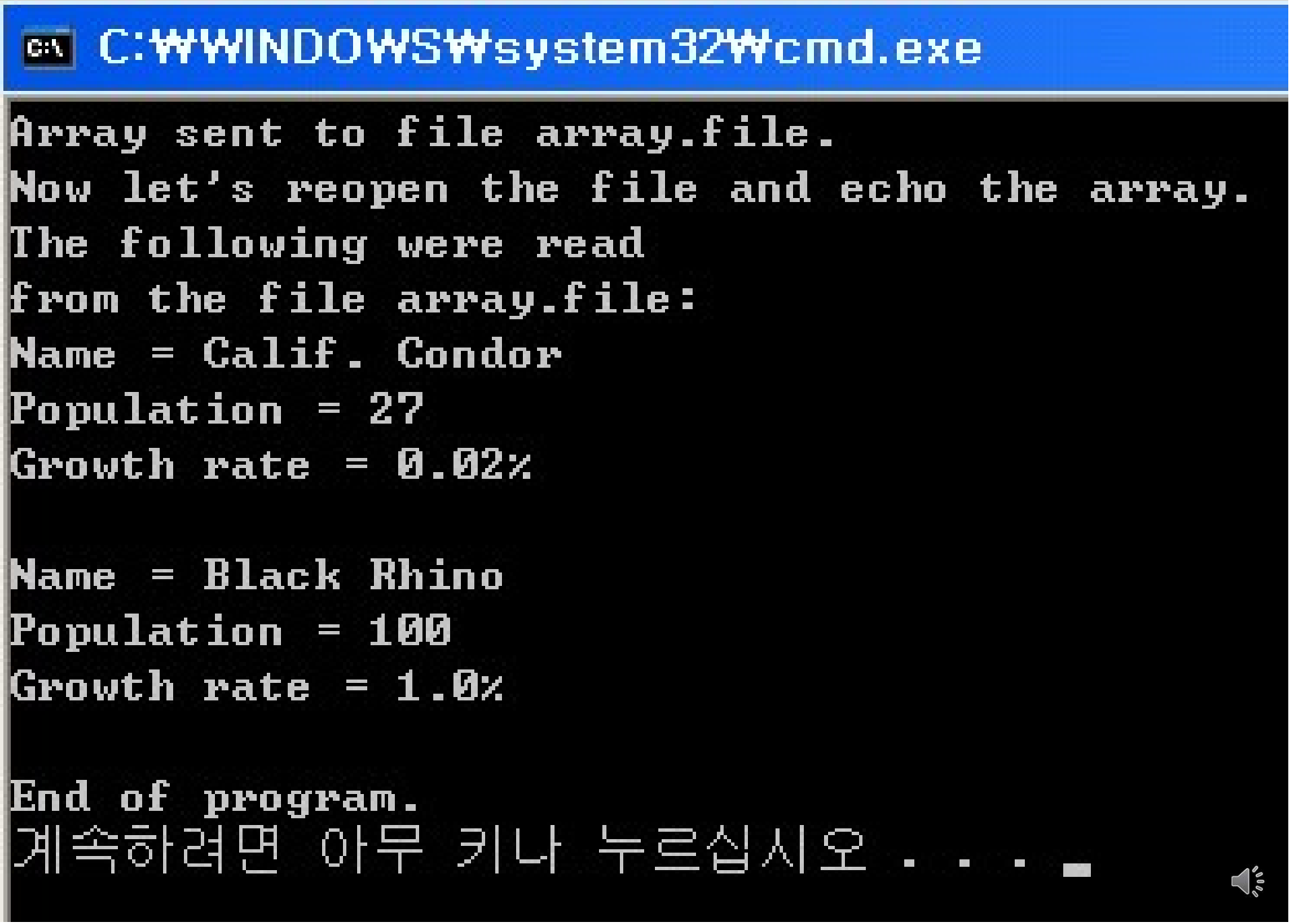
- Array objects in Binary Files
  - » Entire arrays can be saved to a binary file using writeObject
  - » and later read by using readObiect

Listing 10.11 File I/O of an Array Object - ArrayIODemo.java

```
// Listing 10.11 File I/O of an Array Object

import java.io.*;


public class ArrayIODemo
{

   public static void main(String[] args)
   {
      Species[] oneArray = new Species[2];
      oneArray[0] =
               new Species("Calif. Condor", 27, 0.02);
      oneArray[1] =
               new Species("Black Rhino", 100, 1.0);
```

```
try
      {
         ObjectOutputStream outputStream =
               new ObjectOutputStream(
                  new FileOutputStream("array.file"));
         outputStream.writeObject(oneArray);//
         outputStream.close( );
      }
      catch(IOException e)
      {
         System.out.println(
                  "Error writing to file array.file.");
         System.exit(0);
      }


       System.out.println(
                "Array sent to file array.file.");

      System.out.println(
           "Now let's reopen the file and echo the array.");
```

```
Species[] anotherArray = new Species[2];


try
{
    ObjectInputStream inputStream =
        new ObjectInputStream(
            new FileInputStream("array.file"));

   // Notice the Type Casts
   anotherArray = (Species[])inputStream.readObject( );
   inputStream.close( );
}
catch(Exception e)
{
    System.out.println(
            "Error reading file array.file.");
    System.exit(0);
}
```


C:₩WINDOWS₩system32₩cmd.exe
Array sent to file array.file.
Now let's reopen the file and echo the array.
The following were read
from the file array.file:
Name = Calif. Condor
Population = 27
Growth rate = 0.02%
Name = Black Rhino
Population = 100
Growth rate = 1.0%
End of program.
계속하려면 아무 키나 누르십시오 . . .