Labs

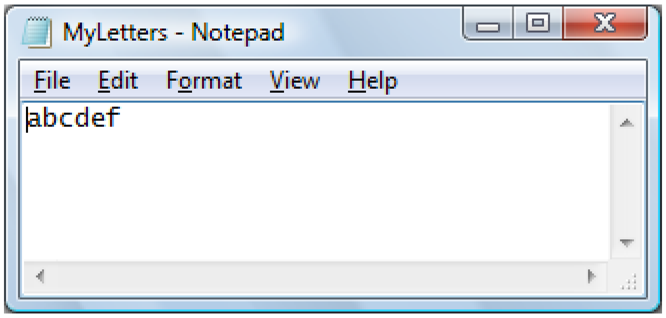
Lab 2.1: FileArray Class

What is the purpose?

In this lab, you will design a class that has a static method named writeArray(). The method should take two arguments, the name of a file and a reference to a char array. The file should be opened, the contents of the array should be written to the file, and then the file should be closed.

Write a second method in the class named readArray(). The method should take two arguments, the name of a file and a reference to a char array. The file should be opened, characters should be read from the file and stored in the array, and then the file should be closed. Demonstrate both the methods in a program.

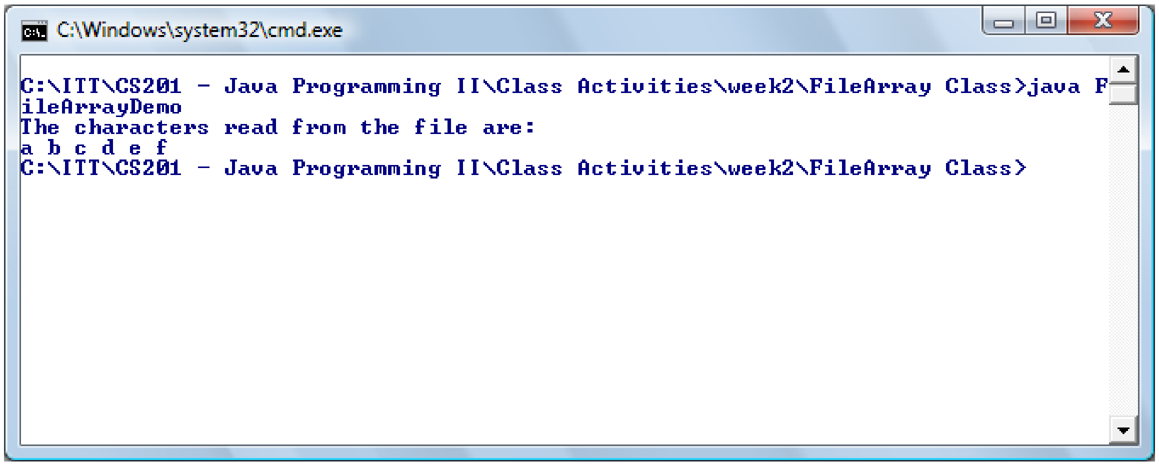
Here is a sample of the data file:



```
abcdef
```

Figure 2-1-1

Here is a sample of the output:



```
C:\ITT\CS201 - Java Programming II\Class Activities\week2\FileArray Class>java F
ileArrayDemo
The characters read from the file are:
a b c d e f
C:\ITT\CS201 - Java Programming II\Class Activities\week2\FileArray Class>
```

Figure 2-1-2

What are the steps?

- **Task 1**

  Procedure:
  1. Create a Java file named FileArray.java.
  2. Create a method named writeArray() to write a char array to a file and throw the I/O exception when an I/O error occurs.
  3. Create a method named readArray() to read characters from a file into a char array and throw the I/O exception when an I/O error occurs.

  Here is a program skeleton:

```
/**
 FileArray Class
*/

import java.io.*;

public class FileArray
{
 /**
  The writeArray method writes a char array to a file.
  @param filename The name of the file.
  @param array The array of characters to write.
  @exception IOException When an IO error occurs.
 */

 public static void writeArray(String filename, char[] array) throws
IOException
 {

 }

 /**
  The readArray method reads characters from a file
  into a char array.
  @param filename The name of the file.
  @param array The array to hold the characters.
  @exception IOException When an IO error occurs.
 */

 public static void readArray(String filename, char[] array) throws
IOException
 {

 }
}
```

Figure 2-1-3: Sample Program Skeleton 1

4. Create a demo program to implement the FileArray class. The demo program should write an array of characters to a text file and read the contents of the text file into a test array.

Here is a program skeleton:

```
/**
 Demo Program for the FileArray Class
*/

import java.io.*;

public class FileArrayDemo
{
 public static void main(String[] args)
 {
  char[] letters = {'a', 'b', 'c', 'd', 'e', 'f' };
  char[] test = new char[6];

  try
  {
   // Write the array to the file MyLetters.txt.

   // Read the contents of the file into the test array.

   System.out.println("The characters read from the file are:");
   // Display the characters.

  }
  catch (IOException e)
  {
   System.out.println("Error = " + e.getMessage());
  }
 }
}
```

Figure 2-1-4: Sample Program Skeleton 2

5. Submit a Word document with your Java code and some screen shots of the output or demonstrate the program to your instructor.

Did it work?

Were you able to:

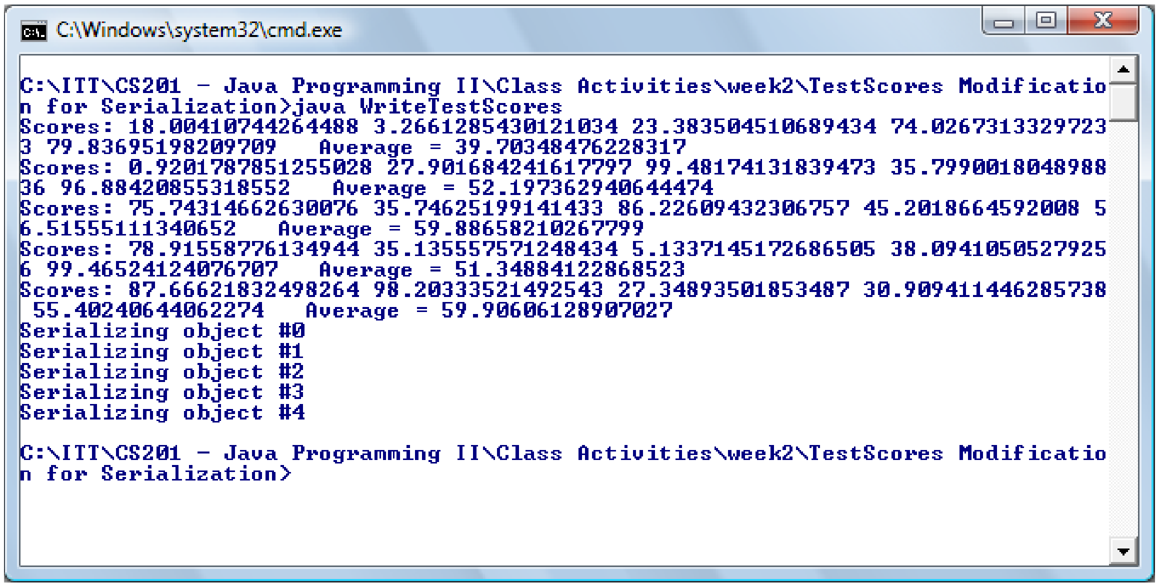
- Write an array of characters to the data file by using writeArray() method?
- Read an array of characters from the data file by using readArray() method?
- Print out the data from the data file?

Lab 2.2: TestScores Modification for Serialization

What is the purpose?

In this lab, you will modify the TestScores class that you created for Lab 1.1 to make it serializable. Write a program that creates an array of at least five TestScore objects and serializes these. Write another program that deserializes the objects from the file.

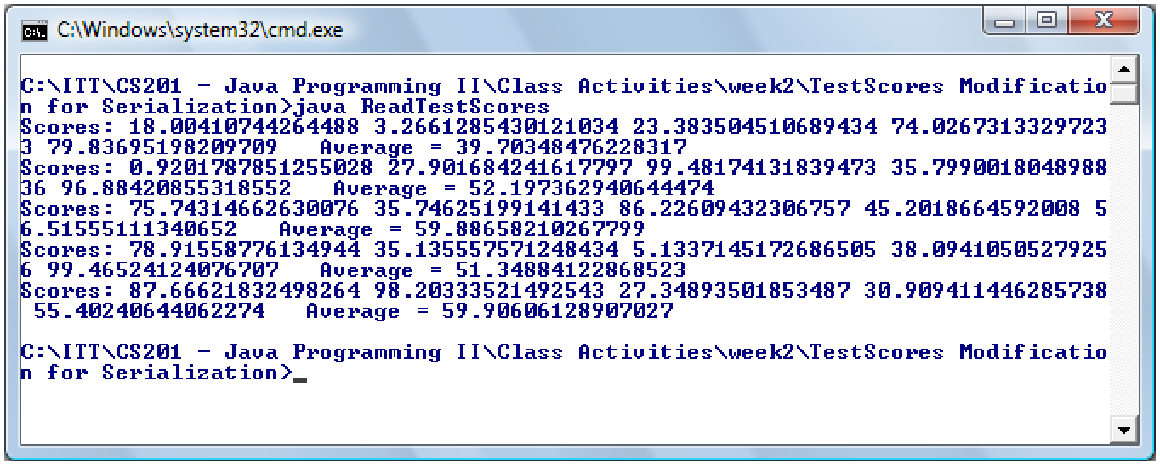
Here are samples of the output:



```
C:\ITT\CS201 - Java Programming II\Class Activities\week2\TestScores Modificatio
n for Serialization>java WriteTestScores
Scores: 18.00410744264488 3.2661285430121034 23.383504510689434 74.0267313329723
3 79.83695198209709   Average = 39.70348476228317
Scores: 0.9201787851255028 27.901684241617797 99.48174131839473 35.7990018048988
36 96.88420855318552   Average = 52.19736294064474
Scores: 75.74314662630076 35.74625199141433 86.22609432306757 45.2018664592008 5
6.51555111340652   Average = 59.88658210267799
Scores: 78.91558776134944 35.135557571248434 5.133714517268650 5 38.0941050527925
6 99.46524124076707   Average = 51.34884122868523
Scores: 87.66621832498264 98.20333521492543 27.34893501853487 30.909411446285738
 55.40240644062274   Average = 59.90606128907027
Serializing object #0
Serializing object #1
Serializing object #2
Serializing object #3
Serializing object #4

C:\ITT\CS201 - Java Programming II\Class Activities\week2\TestScores Modificatio
n for Serialization>
```

Figure 2-2-1: Sample Output 1



```
C:\ITT\CS201 - Java Programming II\Class Activities\week2\TestScores Modificatio
n for Serialization>java ReadTestScores
Scores: 18.00410744264488 3.2661285430121034 23.383504510689434 74.0267313329723
3 79.83695198209709   Average = 39.70348476228317
Scores: 0.9201787851255028 27.901684241617797 99.48174131839473 35.7990018048988
36 96.88420855318552   Average = 52.19736294064474
Scores: 75.74314662630076 35.74625199141433 86.22609432306757 45.2018664592008 5
6.51555111340652   Average = 59.88658210267799
Scores: 78.91558776134944 35.135557571248434 5.133714517268650 5 38.0941050527925
6 99.46524124076707   Average = 51.34884122868523
Scores: 87.66621832498264 98.20333521492543 27.34893501853487 30.909411446285738
 55.40240644062274   Average = 59.90606128907027

C:\ITT\CS201 - Java Programming II\Class Activities\week2\TestScores Modificatio
n for Serialization>_
```

Figure 2-2-2: Sample Output 2

What are the steps?

- **Task 1**

  Procedure:

1. Create a Java file named TestScores.java that implements the Serializable class.
2. Create a constructor that initializes an object with an array of scores. If the array contains an invalid value (less than 0 or greater than 100), an IllegalArgumentException is thrown.
3. Create a method named getAverage() that returns the average test scores of the object.

Here is a program skeleton:

```
/**
   TestScores Class
*/

import java.io.Serializable;

public class TestScores implements Serializable
{
   // Variable to reference an array of test scores
   private double[] scores;

   /**
      The constructor initializes an object with an array of scores.
      If the array contains an invalid value (less than 0 or greater
      than 100) an exception is thrown.
      @param s The array of test scores.
      @exception IllegalArgumentException When the
      argument array contains an invalid value.
   */

   public TestScores(double[] s) throws IllegalArgumentException
   {
      // Create an array to hold the scores passed
      // as an argument.
      scores = new double[s.length];

      // Copy the scores passed as an argument into
      // the new array. Check for illegal values as
      // they are copied.

   }

   /**
      The getAverage method returns the average
      of the object's test scores.
      @return The average of the object's test scores.
   */

   public double getAverage()
   {
      double total = 0.0;  // Accumulator

      // Accumulate the sum of the scores.
      // return the average.
      return (total / scores.length);
   }

   /**
      toString method
      @return A string representation of an object.
   */

   public String toString()
   {
      String str = "Scores: ";

      str += "  Average = " + getAverage();
      return str;
   }
}
```

Figure 2-2-3: Sample Program Skeleton 1

4. Create a demo program named WriteTestScores.java to implement the TestScores class and create an array of at least five TestScore objects and serialize these.
5. Create a method named getRandomScores() that returns an array with random numbers stored in it.

Here is a program skeleton:

```
/**
 Demo program for the TestScores Class
*/

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

public class WriteTestScores
{
 public static void main(String[] args)
 {
  double[] randomScores;      // Random test scores
  TestScores[] ts = new TestScores[5]; // Array to hold TestScore
objects

  try
  {
   // Create objects, store them in the array, and display them.

   // Serialize the objects.
   FileOutputStream outStream = new FileOutputStream(“Objects.dat”);
   ObjectOutputStream objectFile = new ObjectOutputStream(outStream);


   // Close the file.
   objectFile.close();
  }
  catch (IllegalArgumentException e)
  {
   System.out.println(“Invalid score found.\n” + e.getMessage());
  }
  catch (IOException e)
  {
   System.out.println(“Error – “ + e.getMessage());
  }
 }

 /**
  The getRandomScores method returns an array with
  random numbers stored in it.
  @return An array holding random test scores.
 */

 public static double[] getRandomScores()
 {
  // Create a Random object for random number generation.
  Random rand = new Random();

  double[] array = new double[5];

  return array;
 }
}
```

Figure 2-2-4: Sample Program Skeleton 2

6. Create a demo program named ReadTestScores.java to implement the TestScores class and deserialize the objects from the file.

Here is a program skeleton:

```
/**
 Demo program for the TestScores Class
*/

import java.io.*;

public class ReadTestScores
{
 public static void main(String[] args)
 {
  TestScores[] ts = new TestScores[5];

  try
  {
   // Create the file objects.
   FileInputStream inStream = new FileInputStream(“Objects.dat”);
   ObjectInputStream objectFile = new ObjectInputStream(inStream);

   // Deserialize the objects and display them.
   // Close the file.
   objectFile.close();
  }
  catch (Exception e)
  {
   System.out.println(“Error – “ + e.getMessage());
  }
 }
}
```

Figure 2-2-5: Sample Program Skeleton 3

7. Submit a Word document with your Java code and some screen shots of the output or demonstrate the program to your instructor.

Did it work?

Were you able to:

- Serialize the TestScores class?
- Create a Java class that serializes an array of five TestScore objects?
- Create a Java class that deserializes the objects from a file?