File I/O

Part 3: More on Binary Data

Chapter 2, Core Java, Volume II &

Chapter 15, Java How to Program

Contents

Reading and Writing Primitive Types

- Storing numbers as text is less efficient than storing them in binary.
- DataInput/DataOutput interfaces have methods to read/write primitive types and String in binary format:

  readInt() / writeInt()

  readDouble()/writeDouble()

  readLine()/writeString()

  and so on.
- DataInputStream/DataOutputStream classes implement DataInput/DataOutput interfaces.
- A DataInputStream/DataOutputStream can wrap any stream:

```
DataInput in = new DataInputStream(Files.newInputStream(path));
DataOutput out = new DataOutputStream(Files.newOutputStream(path));
```

Random Access Files

- Reading/writing stream data is sequential.
- RandomAccessFile: You can jump to any file position and start reading/writing.
- RandomAccessFile implements DataInput and DataOutput interfaces.
- Open modes:
  - "r" for reading
  - "rw" for writing

```
RandomAccessFile file = new RandomAccessFile(filenameString, "rw");
```

- Methods for file positions:
  - The getFilePointer() method yields the current position (as a long).
  - The seek() method moves to a new position.
- Example: Increment an integer that you just read:

```
int value = file.readInt();
file.seek(file.getFilePointer() - 4);
file.writeInt(value + 1);
```

Example: Random Access

```
import java.io.*;
import java.util.*;
import java.time.*;
public class RandomAccessTest
{
   public static void main(String[] args) throws IOException
  {   Employee[] staff = new Employee[3];
      staff[0] = new Employee("Carl Cracker", 75000, 1987, 12, 15);
      staff[1] = new Employee("Harry Hacker", 50000, 1989, 10, 1);
      staff[2] = new Employee("Tony Tester", 40000, 1990, 3, 15);
      try (DataOutputStream out = new DataOutputStream(new FileOutputStream("employee.dat")))
     {  // save all employee records to the file employee.dat
        for (Employee e : staff)
           writeData(out, e);
     }
```

Example: Random Access

```
    try (RandomAccessFile in = new RandomAccessFile("employee.dat", "r"))
    {  // retrieve all records into a new array and compute the array size
        int n = (int)(in.length() / Employee.RECORD_SIZE);
        Employee[] newStaff = new Employee[n];
        for (int i = n - 1; i >= 0; i--) // read employees in reverse order
        {
          in.seek(i * Employee.RECORD_SIZE);
           newStaff[i] = readData(in);
        }
        // print the newly read employee records
        for (Employee e : newStaff)
           System.out.println(e);
    }
} // end of main()
```

Example: Random Access

```
/** Writes emplyee data to a data output
 * @param out the data output    * @param e the employee    */
public static void writeData(DataOutput out, Employee e) throws IOException
{
   DataIO.writeFixedString(e.getName(), Employee.NAME_SIZE, out);
   out.writeDouble(e.getSalary());
   LocalDate hireDay = e.getHireDay();
   out.writeInt(hireDay.getYear());
   out.writeInt(hireDay.getMonthValue());
   out.writeInt(hireDay.getDayOfMonth());
}
```

Example: Random Access

```
/* Reads employee data from a data input
 @param in the data input
 @return the employee*/
  public static Employee readData(DataInput in) throws IOException
  {
     String name = DataIO.readFixedString(Employee.NAME_SIZE, in);
     double salary = in.readDouble();
     int y = in.readInt();
     int m = in.readInt();
     int d = in.readInt();
     return new Employee(name, salary, y, m, d);
  }
} // end of RandomAccessTest class
```

Example: Random Access

```
import java.time.*;
public class Employee
{
   public static final int NAME_SIZE = 40;
   public static final int RECORD_SIZE = 2 * NAME_SIZE + 8 + 4 + 4 + 4;
   private String name;
   private double salary;
   private LocalDate hireDay;
   public Employee() {}
   public Employee(String n, double s, int year, int month, int day)
   {
     name = n;
     salary = s;
     hireDay = LocalDate.of(year, month, day);
   }
```

Example: Random Access

```
public String getName()
{
   return name;
}
public double getSalary()
{
   return salary;
}
public LocalDate getHireDay()
{
   return hireDay;
}
```

```
public void raiseSalary(double byPercent)
{
  double raise = salary * byPercent / 100;
  salary += raise;
}
 public String toString()
 {
    return getClass().getName()
       + "[name=" + name
       + ",salary=" + salary
       + ",hireDay=" + hireDay
       + "]";
 }
} // end of employee
```

Example: Random Access

```
import java.io.*;
public class DataIO
{
   public static String readFixedString(int size, DataInput in)    throws IOException
   {
      StringBuilder b = new StringBuilder(size);
      int i = 0;
      boolean more = true;
      while (more && i < size)
      {
         char ch = in.readChar();
         i++;
         if (ch == 0) more = false;
         else b.append(ch);
      }
      in.skipBytes(2 * (size - i));  // skip the remaining bytes for the name field
      return b.toString();
   } // end of readFixedString ()
```

Example: Random Access

```
  public static void writeFixedString(String s, int size, DataOutput out)
  throws IOException
  {
     char ch;
     for (int i = 0; i < size; i++)
     {
        ch = 0;  // fill the area that exceeds the lenghth of the string with zero
        if (i < s.length()) ch = s.charAt(i);
        out.writeChar(ch);
     }
  }
} // end of DataIO class
```