CSIT121
Object Oriented Design and Programming

Lesson 6

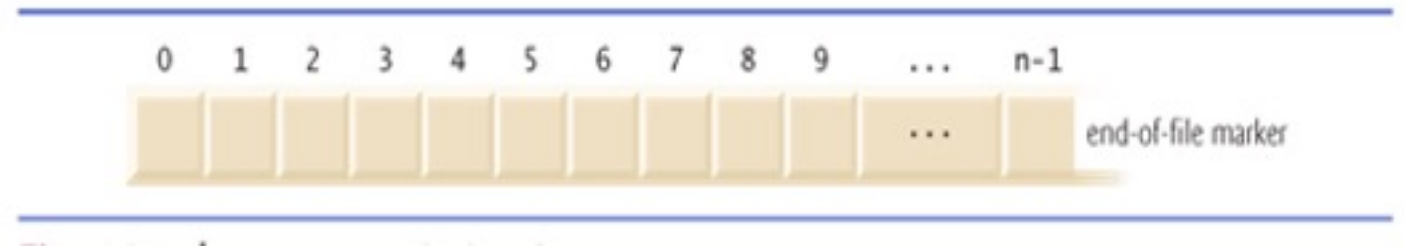
Files I/O and Object Serialization

Introduction

- Data stored in variables and arrays is temporary
  - It's lost when a local variable goes out of scope or when the program terminates
- For long-term retention of data, computers use **files.**
- Computers store files on **secondary storage devices**
  - hard disks, flash drives, DVDs and more.
- Data maintained in files is **persistent data** because it exists beyond the duration of program execution.

Files and Streams

- Java views each file as a sequential **stream of bytes.**
- Every operating system provides a mechanism to determine the end of a file, such as an **end-of-file marker** or a count of the total bytes in the file that is recorded in a system-maintained administrative data structure.
- A Java program simply receives an indication from the operating system when it reaches the end of the stream



Files and Streams

- File streams can be used to input and output data as bytes or characters.
  - **Byte-based streams** output and input data in its ***binary*** format—a `char` is two bytes, an `int` is four bytes, a `double` is eight bytes, etc.
  - **Character-based streams** output and input data as a ***sequence of characters*** in which every character is two bytes—the number of bytes for a given value depends on the number of characters in that value.
- Files created using **byte-based streams** are referred to as **binary files.**
- Files created using **character-based streams** are referred to as **text files.** Text files can be read by text editors.
  - Each character is represented using Unicode.
- Binary files are read by programs that understand the specific content of the file and the ordering of that content.

Writing to a binary file

```
import java.io.*;

public class WriteToBinFile {

    public static void main(String[] args) {
        char[] characters = { 'a', 'b', 'c' };
        int[] ints = {1,2,3};

        String outputFile = "test.bin";

        try {
            OutputStream outputStream = new FileOutputStream(outputFile);
            for (char c : characters) {
                outputStream.write(c); //write the unicode in
            }

            for (int i : ints) {
                outputStream.write(i);
            }

            outputStream.close();

        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }
}
```

Read from binary file

```
import java.io.*;
import java.nio.charset.Charset;

public class ReadFromBinFile {

    public static void main(String[] args) {

        String inputFile = "test.bin";
        try {
            InputStream inputStream = new FileInputStream(inputFile);

            boolean carryOnReading = true;
            int counter = 0;
            while (carryOnReading) {

                int byteRead = inputStream.read();
                counter++;

                if (byteRead == -1) {
                    carryOnReading = false;
                } else {

                    System.out.println(byteRead);

                    if(counter<=3) {
                        byte[] bytes = {(byte)byteRead};
                        String str = new String(bytes, Charset.forName("UTF-8"));
                        System.out.println(str);
                    }
                }
            }

        } catch (IOException ex) {
            System.out.println("IO Exception");
        }
    }

}
```

*Binary files are read by programs that understand the specific content of the file and the ordering of that content.

Write to text file

```
import java.io.*;

public class WriteToTextFile {

    public static void main(String[] args) {

        String outputFileName = "fruits.txt";

        try {
            PrintWriter pw = new PrintWriter(outputFileName);

            pw.write("Apple\n");
            pw.write("Orange\n");
            pw.write("Pear");

            pw.close();


        }catch(FileNotFoundException ex) {
            System.out.println("Unable to open file for writing");
        }
    }
}
```

```
1 Apple
2 Orange
3 Pear
```

Read from text file

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

public class ReadFromTextFile {

    public static void main(String[] args) {

        String inputFileName = "fruits.txt";
        try {

            File file = new File(inputFileName);
            Scanner reader = new Scanner(file);

            while(reader.hasNextLine()) {
                String line = reader.nextLine();
                System.out.println(line);
            }
            reader.close();

        }catch(FileNotFoundException ex) {
            System.out.println("Unable find file for reading");
        }
    }
}
```

CSV file

- Comma-separated values (CSV) file
  - A **text file** with delimited text using comma as a separate value
  - Stores tabular data (numbers and text) as plain text.
  - Each line/row of a file is a data record.
  - Each line/row of a file consists of one or more fields, separated by commas.
  - Example:

```
1 student_number,first_name,last_name,score
2 S111,John,Smith,80
3 S222,Peter,Lee,70
4 S333,Bob,Tan,85
```

Reading from CSV file

- Reading a csv is the same as reading a text file.
- Extract each data item by using String split().

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

public class ReadFromCSVFile {

    public static void main(String[] args) {
        String inputFileName = "results.csv";
        try {
            File file = new File(inputFileName);
            Scanner reader = new Scanner(file);
            String header = reader.nextLine(); //read in the header

            while(reader.hasNextLine()) {

                //trim() is to remove extra beginning and ending white spaces
                String line = reader.nextLine().trim();
                if(line.equals("")) {//skip empty line
                    continue;//go back to while()
                }

                String[] data = line.split(",");
                String studentNumber = data[0].trim();
                String firstName = data[1].trim();
                String lastName = data[2].trim();
                double score = Double.parseDouble(data[3].trim());

                System.out.println(studentNumber);
                System.out.println(firstName);
                System.out.println(lastName);
                System.out.println(score);
                System.out.println();

            }
            reader.close();

        }catch(FileNotFoundException ex) {
            System.out.println("Unable find file for reading");
        }
    }
}
```

Mapping records in CSV file to objects

- Each record in the student.csv file can be mapped to one student object

```
public class Student {

    private String studentNumber;
    private String firstName;
    private String lastName;
    private double score;

    public Student(String studentNumber, String firstName, String lastName, double score) {
        this.studentNumber = studentNumber;
        this.firstName = firstName;
        this.lastName = lastName;
        this.score = score;
    }

    public void display() {
        System.out.println("Student number:"+this.studentNumber);
        System.out.println("First name:"+this.firstName);
        System.out.println("Last name:"+this.lastName);
        System.out.println("Score:"+this.score);
        System.out.println("Grade:"+computeGrade(score));
    }

    public static String computeGrade(double score) {
        String grade = "F";
        if(score>=85) {
            grade = "HD";
        }else if(score>=75) {
            grade = "D";
        }else if(score>=65) {
            grade = "C";
        }else if(score>=50) {
            grade = "P";
        }
        return grade;
    }
}
```

Mapping records in CSV file to objects

- Each record in the student.csv file can be mapped to one student object

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

public class MappingCSVToObject {

    public static void main(String[] args) {
        String inputFileName = "results.csv";
        ArrayList<Student> students = new ArrayList<Student>();
        try {
            File file = new File(inputFileName);
            Scanner reader = new Scanner(file);
            String header = reader.nextLine(); //read in the header

            while(reader.hasNextLine()) {
                //trim() is to remove extra beginning and ending white spaces
                String line = reader.nextLine().trim();
                if(line.equals("")) {//skip empty line
                    continue;//go back to while()
                }

                String[] data = line.split(",");
                String studentNumber = data[0].trim();
                String firstName = data[1].trim();
                String lastName = data[2].trim();
                double score = Double.parseDouble(data[3].trim());

                Student student = new Student(studentNumber,firstName,lastName,score);
                students.add(student);
            }
            reader.close();

        }catch(FileNotFoundException ex) {
            System.out.println("Unable find file for reading");
        }

        for(Student student:students) {
            student.display();
            System.out.println();
        }
    }
}
```

```
Student number:S111
First name:John
Last name:Smith
Score:80.0
Grade:D

Student number:S222
First name:Peter
Last name:Lee
Score:70.0
Grade:C

Student number:S333
First name:Bob
Last name:Tan
Score:85.0
Grade:HD
```

Writing to CSV file

- Writing to a csv file is the same as writing to a text file.
- This example read the result from results.csv, map it to student object and write the grade to grade.csv.

```java
public class Student {

    private String studentNumber;
    private String firstName;
    private String lastName;
    private double score;

    public Student(String studentNumber, String firstName,
                   String lastName, double score) {
        this.studentNumber = studentNumber;
        this.firstName = firstName;
        this.lastName = lastName;
        this.score = score;
    }

    public void display() {
        System.out.println("Student number:"+this.studentNumber);
        System.out.println("First name:"+this.firstName);
        System.out.println("Last name:"+this.lastName);
        System.out.println("Score:"+this.score);
        System.out.println("Grade:"+computeGrade(score));
    }

    public String toCSVString() {
        return this.studentNumber+","+this.computeGrade(score);
    }

    public static String computeGrade(double score) {
        String grade = "F";
        if(score>=85) {
            grade = "HD";
        }else if(score>=75) {
            grade = "D";
        }else if(score>=65) {
            grade = "C";
        }else if(score>=50) {
            grade = "P";
        }
        return grade;
    }
}
```

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

public class WriteToCSVFile {

    public static void main(String[] args) {
        String inputFileName = "results.csv";
        String outputFileName = "grades.csv";
        ArrayList<Student> students = new ArrayList<Student>();
        try {
            PrintWriter pw = new PrintWriter(outputFileName);
            File file = new File(inputFileName);
            Scanner reader = new Scanner(file);
            String header = reader.nextLine(); //read in the header

            while(reader.hasNextLine()) {
                //trim() is to remove extra beginning and ending white spaces
                String line = reader.nextLine().trim();
                if(line.equals("")) {//skip empty line
                    continue;//go back to while()
                }

                String[] data = line.split(",");
                String studentNumber = data[0].trim();
                String firstName = data[1].trim();
                String lastName = data[2].trim();
                double score = Double.parseDouble(data[3].trim());

                Student student = new Student(studentNumber,firstName,lastName,score);
                students.add(student);
                System.out.println();

            }
            reader.close();

            for(Student student:students) {
                pw.write(student.toCSVString()+"\n");
            }
            pw.close();

        }catch(FileNotFoundException ex) {
            System.out.println("Unable find file for reading");
        }
    }
}
```

```
1 S111,D
2 S222,C
3 S333,HD
4
```

Object Serialization

- To read an entire object from or write an entire object to a file, Java provides **object serialization.**
- After a serialized object has been written into a file, it can be read from the file and **deserialized** to recreate the object in memory.
- A class must implement `Serializable` so that it can be serialized.
- `Serializable` is an **interface** (It does not contain methods.).

```
import java.io.Serializable;

public class Foo implements Serializable {
    private int x;
    private String s;

    public Foo() {
        this(100, "hello");
    }

    public Foo(int x, String s) {
        this.x = x;
        this.s = s;
    }

    public String toString() {
        return s + " " + x;
    }
}
```

Object Serialization

- Use `ObjectOutputStream` to write object to a file.

```
import java.io.ObjectOutputStream;

public class SaveFoos {

    public static void main(String[] args) {
        try {
            ObjectOutputStream o = new ObjectOutputStream(new FileOutputStream("foo.bin"));
            Foo foo1 = new Foo();
            Foo foo2 = new Foo(99, "Javaloons");
            Foo foo3 = new Foo(88, "Javaian");
            o.writeObject(foo1);
            o.writeObject(foo2);
            o.writeObject(foo3);
            o.close();
        } catch (IOException e) {
            System.out.println(e);
        }
    }

}
```

Object Serialization

- Use `ObjectInputStream` to read object back from a file.

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

public class ReadFoos {

    public static void main(String[] args) {
        ArrayList<Foo> datas = new ArrayList<Foo>();
        try {
            ObjectInputStream o = new ObjectInputStream(new FileInputStream("foo.bin"));


            boolean endOfFile = false;
            while(!endOfFile) {
                try {
                    // Note the typecasts to Foo
                    Foo foo = (Foo) o.readObject();
                    datas.add(foo);
                }catch(EOFException eof) {
                    endOfFile=true;
                }
            }
            o.close();
        } catch (ClassNotFoundException e) {
            System.out.println(e);
        } catch (FileNotFoundException e) {
            System.out.println(e);
        } catch (IOException e) {
            System.out.println(e);
        }

        for(Foo foo: datas) {
            System.out.println(foo);
        }
    }
}
```

```
hello 100
Javaloons 99
Javaian 88
```

Object Serialization

- Method `readObject` throws an **`EOFException`** if an attempt is made to read beyond the end of the file.

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

public class ReadFoos {

    public static void main(String[] args) {
        ArrayList<Foo> datas = new ArrayList<Foo>();
        try {
            ObjectInputStream o = new ObjectInputStream(new FileInputStream("foo.bin"));


            boolean endOfFile = false;
            while(!endOfFile) {
                try {
                    // Note the typecasts to Foo
                    Foo foo = (Foo) o.readObject();
                    datas.add(foo);
                }catch(EOFException eof) {
                    endOfFile=true;
                }
            }
            o.close();
        } catch (ClassNotFoundException e) {
            System.out.println(e);
        } catch (FileNotFoundException e) {
            System.out.println(e);
        } catch (IOException e) {
            System.out.println(e);
        }

        for(Foo foo: datas) {
            System.out.println(foo);
        }
    }
}
```

Object Serialization

- Method `readObject` throws a `ClassNotFoundException` if the class for the object being read cannot be located.

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

public class ReadFoos {

    public static void main(String[] args) {
        ArrayList<Foo> datas = new ArrayList<Foo>();
        try {
            ObjectInputStream o = new ObjectInputStream(new FileInputStream("foo.bin"));


            boolean endOfFile = false;
            while(!endOfFile) {
                try {
                    // Note the typecasts to Foo
                    Foo foo = (Foo) o.readObject();
                    datas.add(foo);
                }catch(EOFException eof) {
                    endOfFile=true;
                }
            }
            o.close();
        } catch (ClassNotFoundException e) {
            System.out.println(e);
        } catch (FileNotFoundException e) {
            System.out.println(e);
        } catch (IOException e) {
            System.out.println(e);
        }

        for(Foo foo: datas) {
            System.out.println(foo);
        }
    }
}
```