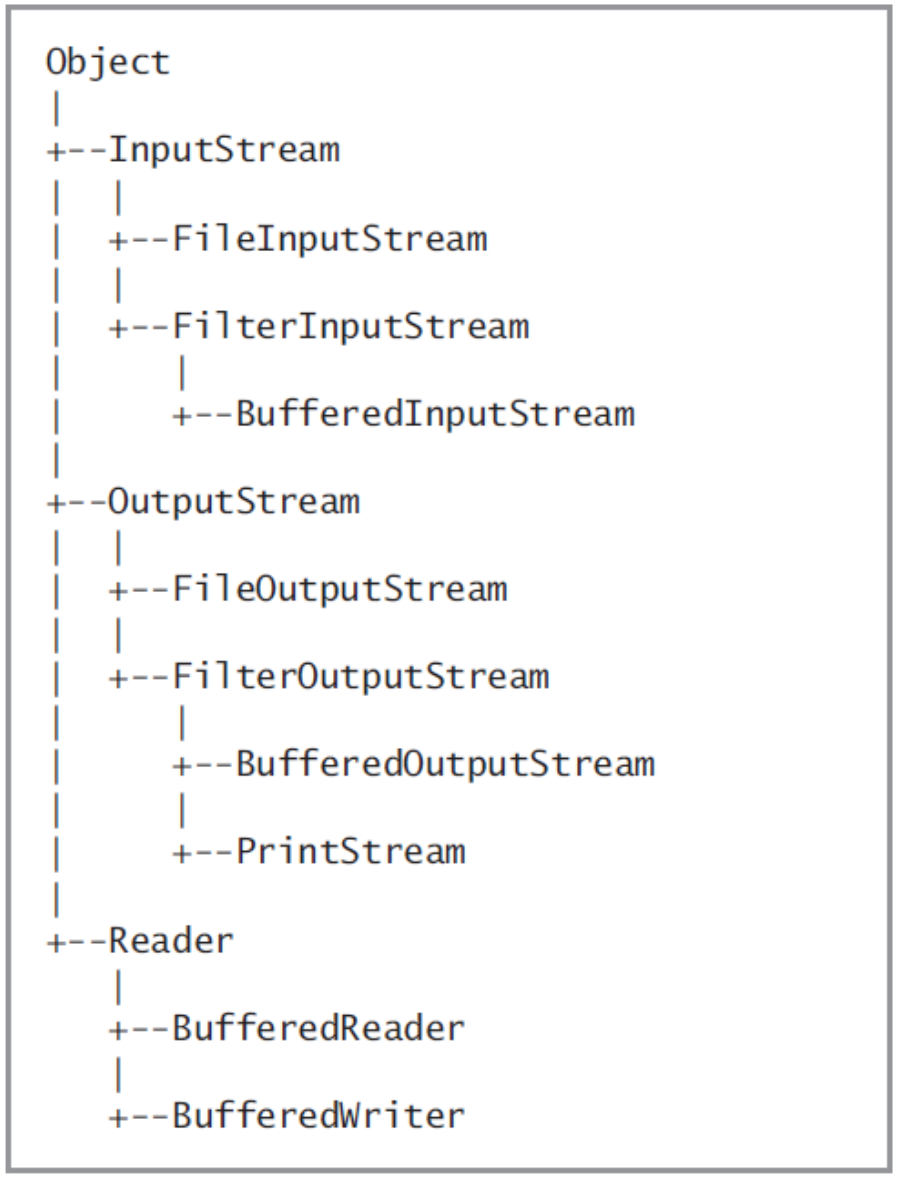


Figure 13-14 Relationship of selected IO classes

```
import java.nio.file.*;
import java.io.*;
public class ReadFile
{
   public static void main(String[] args)
   {
      Path file = Paths.get("C:\\Java\\Chapter.13\\Grades.txt");
      InputStream input = null;
      try
      {
         input = file.newInputStream();
         BufferedReader reader = new
           BufferedReader(new InputStreamReader(input));
         String s = null;
         s = reader.readLine();
         System.out.println(s);
         input.close();
      }
      catch (IOException e)
      {
         System.out.println(e);
      }
   }
}
```

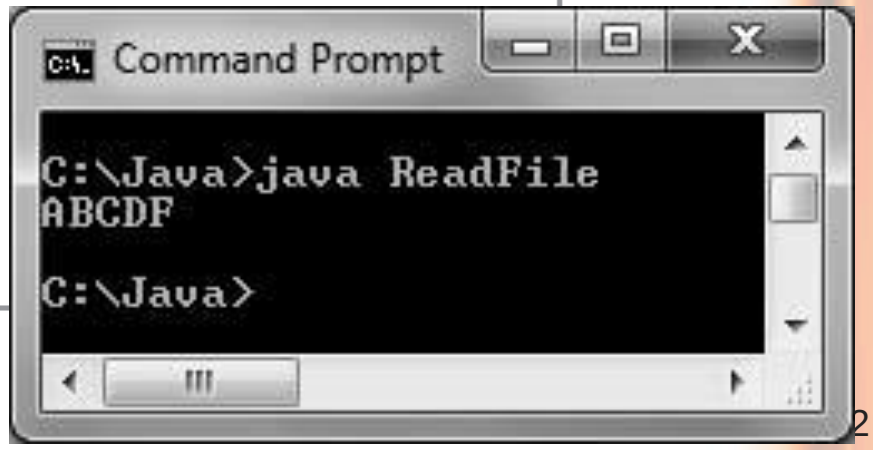


Figure 13-19 The ReadFile class

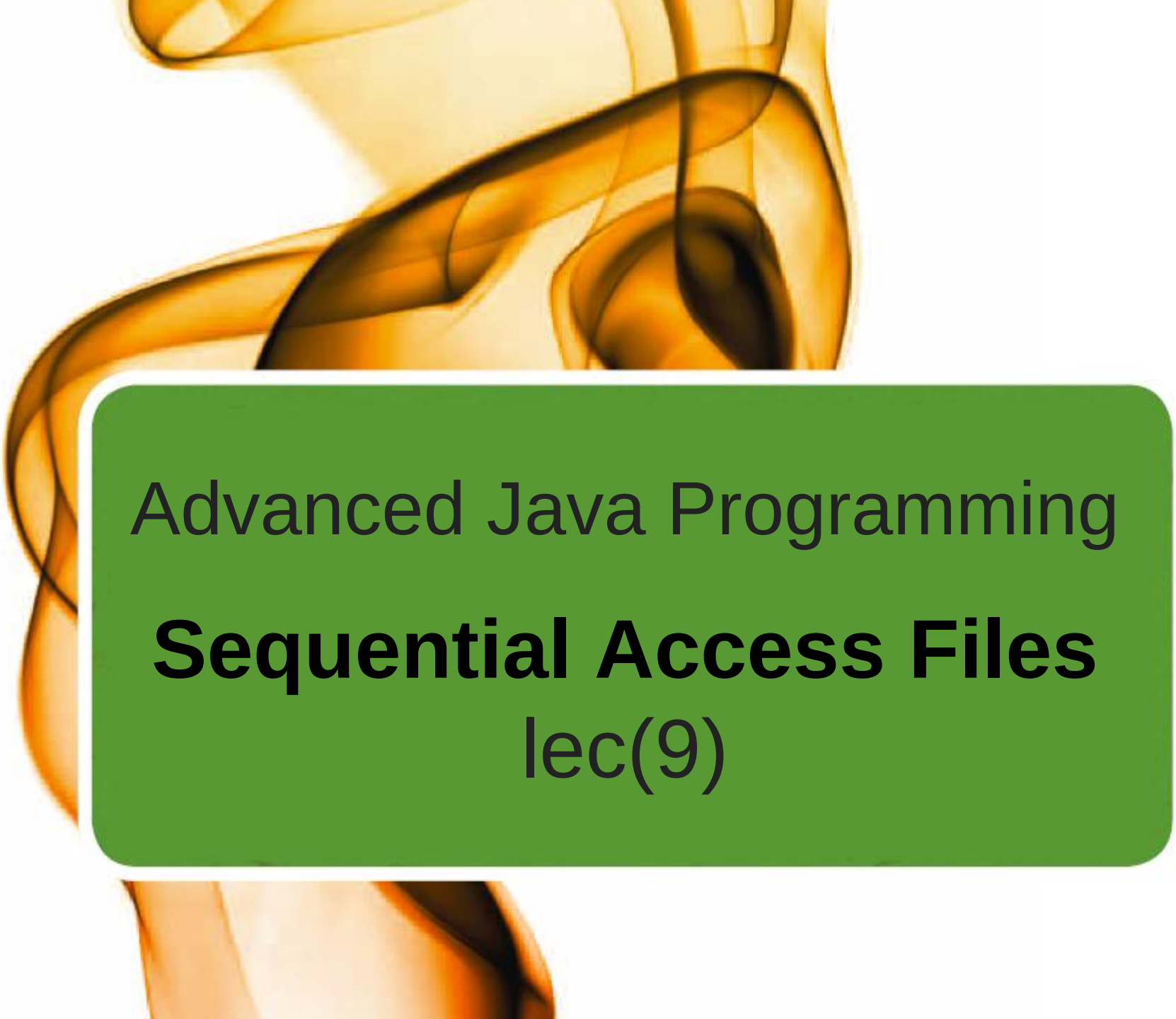
`BufferedWriter` class

- Counterpart to `BufferedReader`
- Writes text to an output stream
- Buffering the characters
- Has three overloaded `write()` methods that provide for efficient writing of characters, arrays, and strings

BufferedWriter Methods

BufferedWriter Method	Description
`close()`	Closes the stream, flushing it first
`flush()`	Flushes the stream
`newline()`	Writes a line separator
`write(String s, int off, int len)`	Writes a `String` from position `off` for length `len`
`write(char[] array, int off, int len)`	Writes a character array from position `off` for length `len`
`write(int c)`	Writes a single character

Table 13-6 `BufferedWriter` methods



Advanced Java Programming

Sequential Access Files

lec(9)

File Organization

- Businesses organize data in a hierarchy
  - **Character**: The smallest useful piece of data
  - Field: is a group of characters that has some meaning.
  - **Record**: is a collection of fields that contain data about an entity
  - Records are grouped to create **files**. Data files consist of related records, such as a company's personnel file that contains one record for each company employee

File Organization (cont'd.)

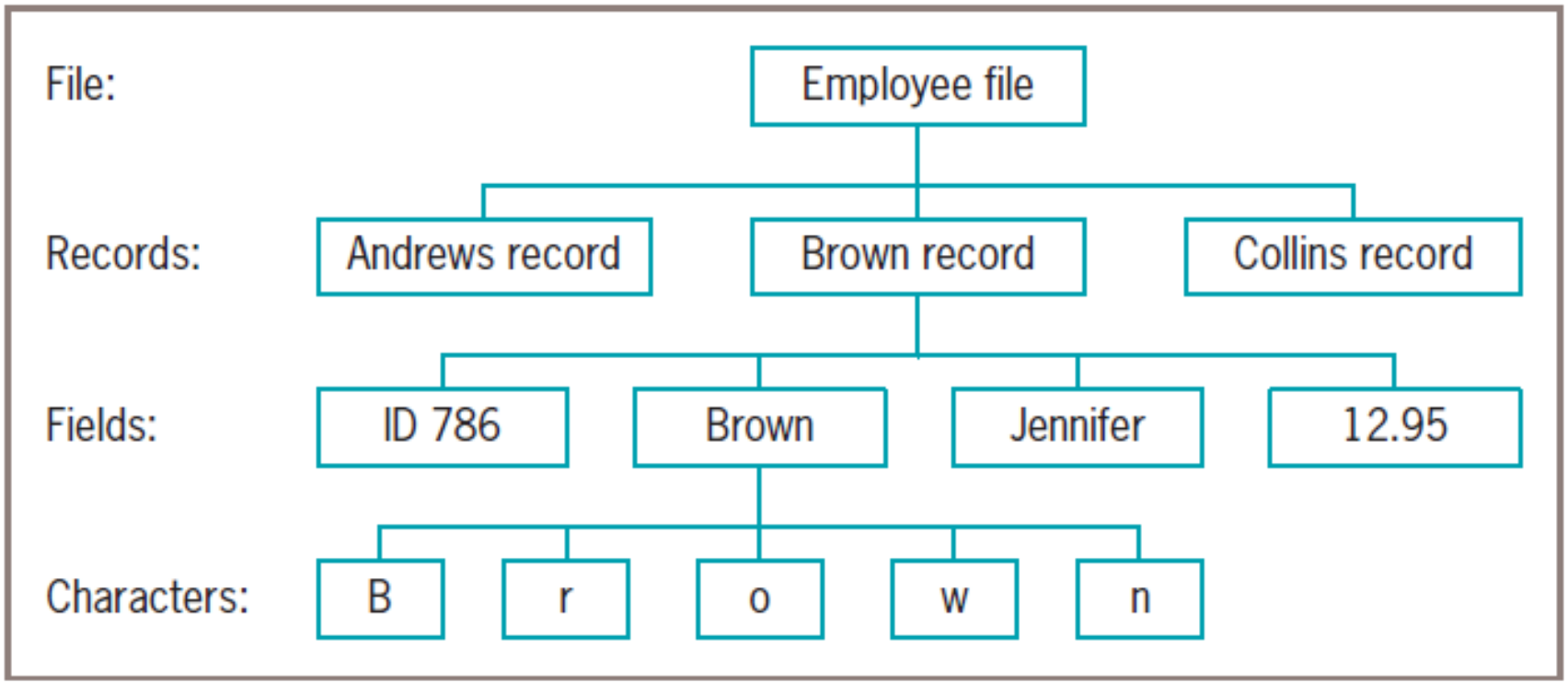


Figure 13-12 Data hierarchy

File Organization (cont'd.)

- A data file can be used as:
  - **a sequential access file** when each record is accessed one after another in the order in which it was stored. Most frequently, each record is stored in order based on the value in some field.
  - When records are not used in sequence, the file is used as **a random access file**

File Organization (cont'd.)

- **File structure**: the way in which records & fields are stored in a data file.
  - e.g: Records can be organized one to a line, and a character can be used to separate their fields.
- **comma-separated values (CSV)**
  - is a widely used format for files
  - used in all sorts of applications, including databases and spreadsheets.

Sequential Access File

- A data file / file of records
  - each record is accessed one after another in the order in which it was stored.
  - Most frequently, each record is stored in order based on the value in some field.
- **Figure 13-21** shows a program that reads employee ID numbers, names, and pay rates from the keyboard and sends them to a data file of personnel records using comma-separated format.

Sequential access files

```
import java.nio.file.*;
import java.io.*;
import static java.nio.file.StandardOpenOption.*;
import java.util.Scanner;
public class WriteEmployeeFile
{
   public static void main(String[] args)
   {
      Scanner input = new Scanner(System.in);
      Path file =
         Paths.get("C:\\Java\\Chapter.13\\Employees.txt");
      String s = "";
      String delimiter = ",";
      int id;
      String name;
      double payRate;
      final int QUIT = 999;
      try
      {
         OutputStream output = new
            BufferedOutputStream(file.newOutputStream(CREATE));
         BufferedWriter writer = new
            BufferedWriter(new OutputStreamWriter(output));
         System.out.print("Enter employee ID number >> ");
         id = input.nextInt();
         while(id != QUIT)
         {
            System.out.print("Enter name for employee #" +
               id + " >> ");
            input.nextLine();
            name = input.nextLine();
            System.out.print("Enter pay rate >> ");
            payRate = input.nextDouble();
            s = id + delimiter + name + delimiter + payRate;
            writer.write(s, 0, s.length());
            writer.newLine();
            System.out.print("Enter next ID number or " +
               QUIT + " to quit >> ");
            id = input.nextInt();
         }
         writer.close();
      }
      catch(Exception e)
      {
         System.out.println("Message: " + e);
      }
   }
}
```

Figure 13-21 The WriteEmployeeFile class

Sequential access files

Sequential access files

Figure 13-24 shows a program that reads the Employees.txt file created by the WriteEmployeeFile program.

```
import java.nio.file.*;
import java.io.*;
public class ReadEmployeeFile
{
   public static void main(String[] args)
   {
      Path file =
         Paths.get("C:\\Java\\Chapter.13\\Employees.txt");
      String s = "";
      try
      {
         InputStream input = new
            BufferedInputStream(file.newInputStream());
         BufferedReader reader = new
            BufferedReader(new InputStreamReader(input));
         s = reader.readLine();
         while(s != null)
         {
            System.out.println(s);
            s = reader.readLine();
         }
         reader.close();
      }
      catch(Exception e)
      {
        System.out.println("Message: " + e);
      }
   }
}
```

Figure 13-24 The ReadEmployeeFile class

Sequential access files