7. Java Files and Streams

Files

- Long term storage or secondary storage
- Persistent data
- Open file, close file, append, etc
- Database files:
    - Tables - rows and columns
    - Columns (fields)
    - Rows (records)
    - Keys: primary, secondary, foreign

Java File Processing

- import the package: java.io which contains definitions for the stream classes
- Sequential access
- Random access

Package java.io

Files and Streams

- In Java, open a file is associated with a new file object and a stream associated with the object.

I/O classes:

- InputStream: an abstract class which is a subclass of the class Object
    - ByteArrayInputStream
    - FileInputStream: inherited from the calss InputStream for input from a file
        - BufferedInputStream
        - DataInputStream

        - PuchbackInputStream
    - FilterInputStream - for working with disk files
    - PipeInputStream
    - ObjectInputStream
    - SequenceInputStream
    - StringBufferInputStream
- OutputStream: an abstract class which is a subclass of the class Object
    - ByteArrayOutputStream
    - FileOutputStream: inherited from the calss InputStream for input from a file
        - BufferedOutputStream
        - DataOutputStream
        - PuchbackOutputStream
    - FilterOutputStream
    - ObjectOutputStream
    - PipeOutputStream
    - SequenceOutputStream
- RadomAccessFile
- Reader class
    - BufferedReader
        - LineNumberReader
    - CharArrayReader
    - FilterReader
        - PuchBackReader
    - InputStreamReader
        - FileReader
    - PipedReader
    - StringReader

- Writer
    - BufferedWriter
        - LineNumberWriter
    - CharArrayWriter
    - FilterWriter
        - PuchBackWriterr
    - InputStreamWriter
        - FileWriter
    - PipedWriter
    - StringWriter

DataStreams

The DataOutput interface supports methods for converting data from any of the Java primitive types (character, int, long, float, double, or boolean) to a series of bytes and writing these bytes to a binary stream. There is also a facility for converting a String into Java modified UTF-8 format and writing the resulting as a series of bytes.

The DataInput interface provides for reading bytes from a binary stream and reconstructing data in any of the Java primitive types. There is also a facility for reconstructing a String from data in Java modified UTF-8 format.

File reading routines using this interface will throw an EOFException if an end-of-file is reached before the desired number of bytes has been read. If any data byte cannot be read for any reason other than end of file, an IOException is thrown.

public interface DataOutput

Write various types of data to output stream.

- public void write (byte[] b) throws IOException
- public void write (byte[] b, int off, int len) throws IOException
- public void write (int b) throws IOException
- public void writeChar(int v) throws IOException
- public void writeChars(String str) throws IOException
- public void writeByte(int v) throws IOException
- public void writeBytes(String str) throws IOException
- public void writeInt(int v) throws IOException
  // Write an integer as a 4 bytes binary quantity
- public void writeShort(int v) throws IOException
- public void writeLong(float v) throws IOException
- public void writeDouble(double v) throws IOException
  // Write a double number as a 8-byte binary quantity
- public void writeBoolean(boolean v) throws IOException
- public void writeUTF(String str) throws IOException
  // Writes string data using Unicode Text Fortmat (UTF)

public interface DataInput

Read data from an input stream and store them in.

- public void readFully (byte[] b) throws IOException
- public void readFully (byte[] b, int off, int len) throws IOException
- public int skipBytes(int) throws IOException
- public boolean readBoolean() throws IOException
- public byte readByte() throws IOException
  // Read 8-bit value
- public int readUnsignedByte() throws IOException
- public short readShort() throws IOException
  // Read 16-bit value
- public short readUnsignedShort() throws IOException
- public int readInt() throws IOException
  // Read an integer as a 4 bytes binary quantity
- public long readLong() throws IOException
  // Read an long as a 8 bytes binary quantity
- public float readFloat() throws IOException
  // Read an float as a 4 bytes binary quantity
- public double readDouble(double v) throws IOException
  // Read a double as a 8 bytes binary quantity
- public char readChar() throws IOException
  // Read Unicode char
- public String readLine() throws IOException
- public String readUTF() throws IOException
  // Read string data using Unicode Text Fortmat (UTF)

Console I/O: Simple Stream Input and Output

The three stream objects (console I/Os) that are created automatically for all users include:

System.out - standard stream output object (monitor screen)

System.err - standard stream error object (monitor screen)

System.in - standard stream input objects (keyboard)

```
// SysWrite.java
//
import java.io.*;

public class SysWrite
{
  public static void main(String[] args) throws IOException
   {
    byte[] msg = {'H', 'e', 'l', 'l', 'o', '\n',
     'W', 'o', 'r', 'l', 'd'};
    System.out.write(msg);
  }
}
```

Output:

```
E:\LinJava\LinJavaExs\JavaIO>javac syswrite.java
E:\LinJava\LinJavaExs\JavaIO>java SysWrite
Hello
World
```

Redirect data to a file:

```
E:\LinJava\LinJavaExs\JavaIO>java SysWrite > syswrite.dat
```

InputStream - a class for reading a sequence of bytes

```
public abstract int read() throws IOException
int read(byte b[])
int read(byte b[], int offset, int length)
long skip(long n)      // skip n bytes in the input stream
int avilable()
void close()
void mark(int readlimit)
void reset()
boolean marksupported()   // check if the stream supports marking
```

OutputStream - a class for writing a sequence of bytes

```
public abstract void write(int b) throws IOException
int write(byte b[])
int write(byte b[], int offset, int length)
void close()
void flush()
```

Create File Object:

```
FileInputStream emp_file = new FileInputStream("employees.dat");
or
File f_name = new File("employees.dat");
FileInputStream emp_file = new FileInputStream(f_name);
```

Read Byte:

```
byte b = emp_file();
```

java.io.FileInputStream

FileInputStream(String name)

FileInputStream(File fileName)

java.io.BufferedInputStream

BufferedInputStream(InputStream in)

BufferedInputStream(InputStream in, int n)

java.io.BufferedOutputStream

BufferedOutputStream(OutputStream out)

BufferedOutputStream(OutputStream out, int n)

java.io.PushbackInputStream

PushbackInputStream(InputStream in)

PushbackInputStream(InputStream in, int n)

void unread(int ch)

Random Access File Class

This class implements DataInput and DataOutput. It lets you find and write data anywhere in a file. For example, you can create random file objects:

```
RandomAccessFile InFile = new RandomAccessFile("customer.dat", "r");
RandomAccessFile InOutFile = new RandomAccessFile("customer.dat", "rw");
```

java.io.RandomAccessFile

public class RandomAccessFile extends Object implement Dataoutput, DataInput

- RandomAccessFile(String fileName, String mode)
- RandomAccessFile(File fileName, String mode)
- void close()
  // Close and release file stream resources
- FileDescriptor getFD()
- long getFilePointer()
  // Return current offset in this file
- long length()
  // Return the length of this file
- void seek(long pos)
  // Sets the file-pointer offset
- void setLength(long newLength)
  // Sets the length of this file
- Plus all methods in DataInput and DataOutput Interfaces

FileDialog Class

```
FileDialog(Frame parent, String title, int type)
String getFile()
String getDirectory()
void setSize(int width, int height)
setVisible(boolean flag)
new File(<directory>, <fileName>)
```

javax.swing.JFileChooser

JFileChooser

//Instance Fields

protected AccessibleContext accessibleContext

static String ACCESSORY_CHANGED_PROPERTY

**static String APPROVED_BUTTON_TEXT_CHANGED_ROEPORT

static String APPROVED_BUTTON_TOOL_TIP_TEXT_CHANGED_PROPERTY

static int APPROVE_OPTION

static String APPROVE_SELECTION

static int CANCEL_OPTION

static String CANCEL_SELECTION

static String CHOOSABLE_FILE_FILTER_CHANGED_PROPERTY

static String CONTROL_BUTTONS_ARE_SHOWN_CHANGED_PROPERTY

static int CUSTOM_DIALOG

static String DIALOG_TITLE_CHANGED_PROPERTY

static String DIALOG_TYPE_CHANGED_PROPERTY

static int DIRECTORIES_ONLY

static String DIRECTORY_CHANGED_PROPERTY

static int ERROR_OPTION

static String FILE_FILTER_CHANGED_PROPERTY

static String FILE_HIDING_CHANGED_PROPERTY

static String FILE_SELECTION_MODE_CHANGED_PROPERTY

static String FILE_SYSTEM_VIEW_CHANGED_PROPERTY

static String FILE_VIEW_CHANGED_PROPERTY
static String FILES_AND_DIRECTORIES
static String FILES_ONLY
static String MULTI_SELECTION_ENABLED_CHANGED_PROPERTY
static int OPEN_DIALOG
static int SAVE_DIALOG
static String SELECTED_FILE_CHANGED_PROPERTY
static String SELECTED_FILES_CHANGED_PROPERTY

//Methods
boolean accept(File f)
void addActionListener(ActionListener l)
void addChoosableFileFilter(FileFilter filter)
void approveSelection()
void cancelSelection()
void changeToParentDirectory()
void ensureFileVisible(File f)
protected void fireActionPerformed(String command)
AccessibleContext getAccessibleContext()

JComponent getAccessory()
int getApproveButtonMneumonic()
String getApprovedButtonText()
String getApprovedButtonToolTipText()
FileFilter[ ] getChoosableFileFilters()
boolean getControlButtonsAreShown()
File getCurrentDirectory()
String getDescription(File f)
String getDialogTitle()
int getDialogType()
FileFilter getFileFilter()
int getFileSelectionMode()

FileSystemView getFileSystemView()

FileView getFileView()

Icon getIcon (File f)

String getName(File f)

File getSelectedFile()

File[ ] getSelectedFiles()

String getTypeDescription (File f)

FileChooseUI getUI()

String getUIClassID()

boolean isAcceptFileFilterUsed()

boolean isDirectorySelectionEnabled()

boolean isFileHidingEnable()

boolean isFileSelectionEnable()

boolean isMultiSelectionEnable()

boolean isTraversable(File f)

protected String paramString()

void removedActionListener(ActionListener l)

boolean removeChoosableFileFilter(FileFilter f)

void rescanCurrentDirectory()

void resetChoosableFileFilters()

void setAcceptAllFileFilterUsed(boolean b)

void setAccessory(Jcomponent newAccessory)

void setapproveButtonMnemonic(char mnemonic)

void setApproveButtonMnemonic(int mnemonic)

void setApproveButtonText(String approveButtonText)

void setApproveButtonToolTipText(String toolTipText)

void setControlButtonAreShown(boolean b)

void setCurrentDirectory(File dir)

void setDialogTitle(String dialogTitle)

void setDialogType(int dialogType)

void setFileFilter(FileFilter filter)

void setFileHidingEnabled(boolean b)
void setFileSelectionMode(int mode)
void setFileSystemView(FileSystemView fsv)
void setFileView(FileView fv)
void setFMultiSelectionEnabled(boolean b)
void setSelectedFile(File file)
void setSelectedFiles(File [ ] selectedFiles)
protected void setup(FileSystemView view)
int showdialog(Component parent, String approveButtonText)
int showOpenDialog(Component parent)
int showSaveDialog(Component parent)
void updateUI()

File Management

Java provides File class for working with the file system of a computer.
Information concerning the storage of the file on a disk such as

- Last modified date
- File size
- Directory path information

java.io.File

public class File extends Object implements Serializable, Comparable

//Instance Fields

- public static String pathSeparator
- public static final char pathSeparatorChar
- public static final String separator
- public static final char separatorChar
  // on the Win32 system '\', on UNIX systems '/'
- //Constructors

- public File(File parentPathName, String childPathName)
- public File(String pathname)
- File(String parentPathName, String childPathName)
- //Method
- boolean canRead()
  // if a file readable
- boolean canWrite()
  // if a file writeable
- int compareTo(Filepathname)
- int compareTo(Object o)
- boolean createNewFile()
- static File createTempFile(String prefix, String suffix)
- boolean delete()
- void deleteOnExit()
- boolean equal(Object obj)
- boolean exists()
  // if a file exists
- File getAbsoluteFile()
- String getAbsolutePath()
  // Return the absolute pathname string
- File getCannonicalFile()
- String getCannonicalPath()
- String getName()
  // Returns the name of the file or directory
- String getParent()
  // gets the folder's name
- File getParentFile()
- String getPath()
- int hashCode()
- boolean isAbsolute()

// On UNIX with prefix "/"

// On Win32 systems prefix with "\\"

- boolean isFile()
- boolean isHidden()
- long lastModified()

  // gets the last update time
- long length()
- String [ ] list()
- String [ ] list(FilenameFilter filter)
- File[ ] listFiles()
- File[ ] listFiles(FileFilter filter)
- File[ ] listFiles(FilenameFilter filter)
- static FilelistRoots()
- boolean mkdir(0
- boolean mkdirs()
- boolean renameTo(File dest)
- boolean setLastModified(long time)
- boolean setReadOnly()
- String toString()
- URL toRUL()

Example:

```
File tempfile = new File("tempdata.dat");
```