ตัวอย่างที่ 1 ตัวอย่างการใช้คลาส File สำหรับสอบถามรายละเอียดเกี่ยวกับแฟ้มข้อมูล

```
1:   import java.io.*;
2:   import java.util.Date;
3:   class TestFile {
4:    public static void main (String args[]) {
5:      try {
6:       File f;
7:       String filename="c:\\autoexec.bat";
8:
9:       System.out.println("List of fields of an instance of class File");
10:      System.out.println("path.Separator is "+File.pathSeparator);
11:      System.out.println("separator is "+File.separator);
12:      System.out.println("pathSeparatorChar is "+File.pathSeparatorChar);
13:      System.out.println("separatorChar is "+File.separatorChar);
14:
15:      f = new File(filename);
16:      System.out.println("file path is "+f);
17:      System.out.println("\nCalling some methods of the class File");
18:      long ldate = f.lastModified();
19:      if (ldate == 0)
20:         System.out.println("File "+filename+" does not exist");
21:      else {
22:         Date fdate = new Date(ldate);
23:         System.out.println("lastModified() is "+fdate);
24:      }
25:
26:      System.out.println("canRead() is "+f.canRead());
27:      System.out.println("canWrite() is "+f.canWrite());
28:      System.out.println("exists() is "+f.exists());
29:      System.out.println("isFile() is "+f.isFile());
30:      System.out.println("isDirectory() is "+f.isDirectory());
31:      System.out.println("length() is "+f.length());
32:     }
33:     catch (NullPointerException e) {
34:      System.out.println("No File specified");
35:     }
36:   }
37:  }
```

ตัวอย่างนี้แสดงการใช้คลาส File เพื่อสอบถามข้อมูลเกี่ยวกับแฟ้มข้อมูลหรือไดเรกทอรีโดยไม่มีการอ่านหรือเขียนข้อมูลที่อยู่ในแฟ้มแต่อย่างใด บรรทัดที่ 10-13 พิมพ์ค่าของตัวแปรเกี่ยวกับตัวคั่นชื่อ path และชื่อไฟล์ ซึ่งเก็บไว้ในตัวแปร static ของคลาส File ออกมา ค่าที่พิมพ์ออกมาจะแตกต่างกันไปสำหรับแต่ละระบบปฏิบัติการที่นำโปรแ กรมนี้ไปรัน บรรทัดที่ ๑๘ เป็นการถามวันที่ของไฟล์ที่มีการปรับปรุงแก้ไขครั้งล่าสุด จะคืนค่าเป็นชนิด long ซึ่งมีการเข้ารหัสวัน เดือน ปี ชั่วโมง นาที วินาที และ มิลลิวินาทีไว้เป็นข้อมูลชนิด long เพียงตัวเดียว ถ้าไม่มีไฟล์ตามที่ระบุ ค่านี้จะเป็นศูนย์ ตรงบรรทัดที่ ๑๙ จึงมีการตรวจดูว่าค่านี้เป็น 0 หรือไม่ ถ้าเป็นแสดงว่าไม่พบไฟล์ตามที่ระบุ แต่ถ้าไม่เป็น 0 แสดงว่าได้วันที่ที่มีการปรับปรุงไฟล์ ซึ่งจะนำไปใช้เป็นพารามิเตอร์ให้กับ constructor ของ Date ในบรรทัดที่ ๒๒ เพื่อให้คลาส Date ช่วยถอดรหัสออกมาเพื่อพิมพ์เป็นวันและเวลาที่เราดูแล้วเข้าใจง่ายในบรรทัดที่ ๒๓

บรรทัดที่ ๒๖-๓๐ เป็นการสอบถามคุณสมบัติต่าง ๆ ของไฟล์หรือไดเรกทอรี ส่วนบรรทัดที่ ๓๑ เป็นการแสดงความยาวหรือขนาดของไฟล์ โดยถามจากเมท็อด length()

ตัวอย่างผลลัพธ์จากการทำงานของตัวอย่างที่ 1

```
List of fields of an instance of class File
path.Separator is ;
separator is \
pathSeparatorChar is ;
separatorChar is \
file path is c:\autoexec.bat

Calling some methods of the class File
lastModified() is Sun Aug 24 12:04:34  1997
canRead() is true
canWrite() is true
exists() is true
isFile() is true
isDirectory() is false
length() is 280
```

ตัวอย่างที่ 2 ตัวอย่างการใช้คลาส FileInputStream สำหรับอ่านข้อมูลจากแฟ้ม

```
1:   import java.io.FileInputStream;
2:   import java.io.IOException;
3:   import java.io.FileNotFoundException;
4:   class TestFileInputStream {
5:    public static void main (String args[]) {
6:       String filename = "c:\\autoexec.bat";
7:       int bytesAvailable;
8:       try {
9:          FileInputStream in = new FileInputStream(filename);
10:         bytesAvailable = in.available();
11:         System.out.println("bytes available = " + bytesAvailable);
12:         int ch;
13:         while ( (ch=in.read() ) > -1)
14:            System.out.print( (char)ch );
15:         in.close();
16:      } catch (FileNotFoundException e) {
17:         System.err.println(filename + " is not found");
18:      } catch (IOException e) {
19:         e.printStackTrace();
20:      }
21:   }
22:  }
```

ตัวอย่างนี้แสดงการใช้คลาส FileInputStream อ่านข้อมูลเข้ามาจากแฟ้มทีละไบต์ แล้วนำมาแสดงผลทางจอภาพ เนื่องจากตัวอย่างนี้อ่านข้อมูลจากแฟ้มc:\autoexec.bat ซึ่งเป็นแฟ้มข้อความ (text file) ซึ่งเก็บ 1 ไบต์ต่อ1 ตัวอักษร จึงสามารถนำมาแสดงผลได้ โปรแกรมนี้จะวนลูปอ่านข้อมูลเข้ามาทีละไบต์แล้วพิมพ์ออกมา

บรรทัดที่ ๙ เป็นการเปิดแฟ้มเพื่อการอ่านโดยใช้คลาส FileInputStream ส่วนบรรทัดที่ ๑๐ เป็นการถามว่ามีข้อมูลที่พร้อมให้อ่านจำนวนกี่ไบต์ เก็บไว้ในตัวแปร byteAvailable แล้วนำไปพิมพ์ที่บรรทัด ๑๑

บรรทัด ๑๓ วนลูปอ่านข้อมูลเข้ามาจากแฟ้มทีละไบต์ใส่ไว้ในตัวแปร ch เมท็อด read() ถึงอ่านข้อมูลจาก stream เข้ามาครั้งละ 1 ไบต์และคืนค่าเป็นชนิด int ถ้าอ่านข้อมูลเข้ามาไม่ได้จะคืนค่าเป็น -1 ดังนั้นจึงตรวจสอบว่าค่าที่อ่านมาได้

มากกว่า -1 ใช่หรือไม่ ถ้าใช่แสดงว่าอ่านข้อมูลเข้ามาได้ จะนำข้อมูลไป cast ให้เป็นชนิด character แล้วจึงพิมพ์ออกไป แต่ถ้าไม่ใช่ แสดงว่าอ่านข้อมูลเข้ามาไม่ได้ จะหลุดออกจากลูปไปแล้วปิด stream ตรงบรรทัดที่ ๑๕

โปรแกรมนี้มีการดักจับสิ่งผิดปรกติชนิด FileNotFoundException ด้วยเนื่องจากมีโอกาสที่จะเกิดสิ่งผิดปรกติชนิดนี้ขึ้นได้ถ้าหาแฟ้มที่ต้องการไม่พบ

ตัวอย่างที่ 3 การใช้คลาส FileOutputStream สำหรับเขียนข้อมูลทีละไบต์ลงแฟ้ม

```
1:   import java.io.FileOutputStream;
2:   import java.io.IOException;
3:
4:   class TestFileOutputStream {
5:    public static void main (String args[]) {
6:       String filename = "TestOutputFile.txt";
7:       String s = "Hello Thailand";
8:
9:       try {
10:         FileOutputStream out = new FileOutputStream(filename);
11:         byte buf[] = s.getBytes();  // get array of bytes from string
12:         out.write(buf);
13:         out.close();
14:      } catch (IOException e) {
15:         e.printStackTrace();
16:      }
17:   }
18:  }
```

ตัวอย่างนี้เขียนข้อมูลประเภทข้อความลงในแฟ้ม ข้อความนำมาจากสายอักขระ s ซึ่งเก็บแบบ Unicode 2 ไบต์ต่ออักขระ เรานำสายอักขระนี้มาแปลงให้เป็นแถวลำดับของไบต์เสียก่อนที่บรรทัดที่ 11 แล้วจึงเขียนทุกไบต์ของแถวลำดับลงแฟ้มด้วยเมท็อด write ในบรรทัดที่ 12 แต่ก่อนที่จะเขียนข้อมูลลงแฟ้มได้เราต้องสร้างอ็อบเจ็กต์สำหรับทำหน้าที่ส่งข้อมูลออกขึ้นมาก่อน ในที่นี้คือสร้างอ็อบเจ็กต์จากคลาส FileOutputStream ตรงบรรทัดที่ 10 โดยใส่พารามิเตอร์เป็นชื่อแฟ้มข้อมูลที่ต้องการสร้าง

ตัวอย่างที่ 4 การใช้คลาส DataOutputStream สำหรับเขียนข้อมูลชนิดพื้นฐานต่าง ๆ

```
1:   import java.io.*;
2:
3:   class TestDataOutputStream {
4:       public static void main(String[] args) {
5:           if (args.length != 1) {
6:               System.err.println(
                            "Usage: java TestDataOutputStream <output file>");
7:               System.exit(1);//exit program with return code 1
8:           }
9:           FileOutputStream file_out;
10:          DataOutputStream data_out;
11:
12:          try {
13:              file_out = new FileOutputStream(args[0]);
14:              data_out = new DataOutputStream(file_out);
15:
16:              char a = 'a';
17:              byte b = 2;
```

```
18:                 String c = "abc";
19:                 short d = 4;
20:                 byte[] b2 = {(byte)'a',(byte) 'b',(byte) 'c'};
21:
22:                 data_out.write(b);
23:                 data_out.write(b2, 0, b2.length);
24:                 data_out.writeBoolean(true);
25:                 data_out.writeChar(a);
26:                 data_out.writeBytes(c);
27:                 data_out.writeChars(c);
28:                 data_out.writeDouble(123.456);
29:                 data_out.writeFloat(123.456f);
30:                 data_out.writeInt(678);
31:                 data_out.writeLong(678l);
32:                 data_out.writeShort(d);
33:                 data_out.writeUTF(c);
34:                 data_out.writeUTF("abc\n");
35:                 data_out.write(b);
36:                 data_out.writeShort(d);
37:                 data_out.flush();
38:                 System.out.println("Size of file written: " +
                                       data_out.size());
39:                 data_out.close();
40:             } catch (IOException e) {
41:                 System.out.println(e);
42:             }
43:         }
44:   }
45:
```

ตัวอย่างนี้เขียนข้อมูลประเภทต่าง ๆ ซึ่งกำหนดไว้แล้วในโปรแกรมลงในแฟ้มโดยใช้ตัวกรอง (filter) DataOutputStream ซึ่งสามารถรับข้อมูลพื้นฐานชนิดต่างๆ มาแล้วเขียนออกไปในลักษณะของ byte stream

โปรแกรมนี้ผู้ใช้โปรแกรมเป็นผู้กำหนดชื่อแฟ้มเองว่าจะส่งข้อมูลออกไปเก็บในแฟ้มชื่ออะไร โดยการส่งชื่อแฟ้มให้โปรแกรมในรูปของ command line

บรรทัดที่ ๕ ตรวจสอบว่ามีข้อมูลเข้ามาทาง command line หรือไม่ โดยตรวจจากขนาดหรือความยาวของแถวลำดับ args ถ้าเป็นศูนย์แสดงว่าผู้ใช้ไม่ได้ใส่ข้อมูลเข้ามาทาง command line โปรแกรมจะแสดงวิธีใช้ให้ผู้ใช้ทราบว่าต้องใส่ชื่อแฟ้มเข้ามาด้วยในตอนสั่งรันโปรแกรม

บรรทัด ๑๖-๒๐ เป็นการเตียมข้อมูลที่จะนำไปเขียนลงแฟ้มไว้ในตัวแปรต่าง ๆ ซึ่งจะนำไปใช้ในขั้นตอนต่อไป

บรรทัด ๒๒-๓๖ จะนำข้อมูลที่เตรียมไว้ไปเขียนลงแฟ้มด้วยเมท็อดต่าง ๆ ตามชนิดของข้อมูลที่ต้องการเขียน ตรงบรรทัด ๓๓ และ ๓๔ เมท็อด writeUTF เป็นเมท็อดสำหรับเขียนสายอักขระซึ่งเก็บโดยใช้รหัสอักขระแบบ Unicode 2 ไบต์ต่ออักขระออกไปในรูปแบบมาตรฐาน UTF ซึ่งจะพยายามเขียนข้อมูลให้กระทัดรัดโดยเขียนข้อมูล 1 ไบต์ต่ออักขระถ้าสามารถทำได้ (รายละเอียดว่า UTF มีการเข้ารหัสข้อมูลอย่างไร อยู่นอกเหนือขอบเขตวิชานี้)

บรรทัดที่ ๓๗ สั่ง flush เพื่อไล่ข้อมูลที่อาจคั่งค้างอยู่ในหน่วยความจำให้ลงแฟ้ม

บรรทัดที่ ๓๘ พิมพ์ขนาดของแฟ้มโดยใช้เมท็อด size() ซึ่งคืนค่ามาเป็นจำนวนไบต์ของข้อมูลที่เขียนลงแฟ้มไปแล้ว

บรรทัดที่ ๓๙ ทำการปิดแฟ้มด้วยเมท็อด close() หลังจากปิดแล้วจะเขียนข้อมูลใด ๆ ลงแฟ้มนี้อีกไม่ได้

ตัวอย่างที่ 5 แสดงการใช้.DataInputStream อ่านข้อมูลจากแฟ้มเข้ามาใส่ตัวแปรพื้นฐานชนิดต่างๆ

```
1:  import java.io.*;
2:  class TestDataInputStream {
3:      public static void main(String[] args) {
4:          if (args.length != 1) {
5:              System.err.println(
6:                  "Usage: java TestDataInputStream <output " +
                    "from TestDataOutput example>");
7:              System.exit(1);//exit program with return code 1
8:          }
9:          FileInputStream file_in;
10:         DataInputStream data_in;
11:
12:         try {
13:             file_in = new FileInputStream(args[0]);
14:             data_in = new DataInputStream(file_in);
15:
16:             System.out.println("Available: " + data_in.available());
17:
18:             byte b;
19:             byte[] b2 = new byte[1];
20:             b = data_in.readByte();
21:             System.out.println("Byte: " + b);
22:             data_in.read(b2);
23:             System.out.println("Byte[0]: " + (char)b2[0]);
24:             data_in.read(b2, 0, b2.length);
25:             System.out.println("Byte[0]: " + (char)b2[0]);
26:             int b = data_in.readUnsignedByte();
27:             System.out.println("Unsigned Byte: " + b);
28:             System.out.println("Boolean: " + data_in.readBoolean());
29:             char a = data_in.readChar();
30:             System.out.println("Char: " + a);
31:
32:             byte[] b3 = new byte[3];
33:             data_in.readFully(b3);
34:             System.out.println("readFully: " + (char)b3[0] + (char)b3[1]
35:                                + (char)b3[2]);
36:             data_in.skipBytes(6); // skip string 'abc'
37:             double d1 = data_in.readDouble();
38:             float f1 = data_in.readFloat();
39:             int i = data_in.readInt();
40:             long l = data_in.readLong();
41:             short s = data_in.readShort();
42:             String str = data_in.readUTF();
43:             ub = data_in.readUnsignedByte();
44:             int us = data_in.readUnsignedShort();
45:             System.out.println("UTF String:" + str);
46:         } catch (IOException e) {
47:             System.out.println(e);
48:         }
49:     }
50: }
```

ตัวอย่างนี้อ่านข้อมูลซึ่งถูกเขียนลงแฟ้มจากตัวอย่างที่ 4 เข้าสู่ตัวแปร โปรดสังเกตุว่าลำดับของชนิดของข้อมูลที่อ่านจะต้องตรงกับที่เขียน มิฉะนั้นจะอ่านข้อมูลกลับคืนมาไม่ถูกต้อง

ตัวอย่างที่ 6 การใช้ FileWriter สำหรับเขียน text file

```
1:   import java.io.FileWriter;
2:   import java.io.IOException;
3:
4:   class TestFileWriter {
5:     public static void main (String args[]) {
6:     String filename = "TestFileWriter.txt";
7:     String s = "Hello Thailand";
8:     String CrLf = "\r\n";
9:     try {
10:       FileWriter fr = new FileWriter(filename);
11:       fr.write(s);
12:       fr.write(CrLf);
13:       fr.write(s,6,8);
14:       fr.close();
15:    } catch (IOException e) {
16:       e.printStackTrace();
17:       }
18:    }
19:  }
```

ตัวอย่างนี้สาธิตการเขียนสายอักขระลงแฟ้มข้อข้อความโดยใช้คลาส FileWriter

บรรทัดที่ ๑๑ เขียนข้อมูลสายอักขระ s ลงไฟล์ ส่วนบรรทัดที่๑๒ เขียนอักขระควบคุมเพื่อบอกจุดจบบรรทัดก่อนที่จะเขียนข้อความอื่นเพิ่มเติม บรรทัดนี้เขียน '\r' ซึ่งเป็นอักขระควบคุมการพิมพ์ให้เลื่อนหัวพิมพ์กลับไปชิดซ้าย และ '\n' ซึ่งเป็นอักขระควบคุมให้เลื่อนบรรทัด ในกรณีที่ใช้แฟ้มข้อความในระบบปฏิบัติการ DOS หรือ Windows จะต้องใช้อักขระควบคุมสองตัวนี้คู่กันเพื่อบอกให้ขึ้นบรรทัดใหม่

บรรทัดที่ ๑๓ เขียนสายอักขระ s เริ่มจากตำแหน่งออฟเซ็ท ๖ (ตำแหน่งตัวที่ ๗) จำนวน ๘ คือคำ Thailand ตัวลงแฟ้ม

ตัวอย่างที่ 7 แสดงการใช้คลาส `FileReader`

```
import java.io.FileReader;
import java.io.LineNumberReader;
1:   import java.io.IOException;
2:
3:   class TestFileReader {
4:       public static void main (String args[]) {
5:    String filename = "TestFileWriter.txt";
6:    int c;
7:    try {
8:        FileReader fr = new FileReader(filename);
9:
10:       while (true) {
11:        c = fr.read();
12:        if (c == -1)  // no more data?
13:           break;
14:           System.out.print((char)c);
15:       }
```

```
16:    } catch (IOException e) {
17:        e.printStackTrace();
18:    }
19:       }
20:  }
```

ตัวอย่างนี้อ่านข้อมูลที่เป็น Text file ที่เขียนจากตัวอย่างที่ ๗ เข้ามาทีละอักขระแล้วแสดงผล ตัวอย่างนี้ใช้ลูป while โดยกำหนดเงื่อนไขของลูปให้เป็นจริงเสมอ แล้วไปตรวจเงื่อนไขที่จะออกจากลูปภายในลูป บรรทัดที่ ๑๑ อ่านข้อมูลเข้ามา ๑ ตัวอักษรเก็บไว้ในตัวแปร c

บรรทัดที่ ๑๒ ตรวจว่า ข้อมูลที่อ่านเข้ามามีค่าเป็น -1 ซึ่งหมายถึงอ่านข้อมูลไม่ได้หรือไม่มีข้อมูลให้อ่านอีกต่อไป จึงควรออกจากลูปโดยใช้คำสั่ง break ในบรรทัดที่ ๑๓

ตัวอย่างที่ 8 การใช้ FileReader สำหรับอ่าน text file และการใช้ LineNumberReader แปลงเป็น String

```
1:   import java.io.FileReader;
2:   import java.io.LineNumberReader;
3:   import java.io.IOException;
4:
5:   class TestLineNumberReader {
6:   public static void main (String args[]) {
7:      String filename = "TestFileWriter.txt";
8:      String s;
9:      try {
10:       FileReader fr = new FileReader(filename);
11:       LineNumberReader line = new LineNumberReader(fr);
12:       while ( ( s = line.readLine()) != null) {
13:           System.out.println(line.getLineNumber() + ": " + s);
14:       }
15:   } catch (IOException e) {
16:       e.printStackTrace();
17:     }
18:   }
19:  }
```

จากตัวอย่างที่ 7 ต้องอ่านข้อมูลเข้ามาทีละตัวอักษร ถ้าเราต้องการอ่านเข้ามาเป็น string อาจใช้คลาส `LineNumberReader` ช่วยได้ ตัวอย่างที่ 8 แสดงการใช้เมท็อด `readLine` ของคลาส `LineNumber Reader` ซึ่งคืนค่าเป็นสายอักขระ นอกจากนี้มันยังนับบรรทัดให้ด้วย ถ้าต้องการทราบเลขที่บรรทัดสามารถใช้เมท็อด `getLineNumber` ถามได้

ตัวอย่างที่ 9 การใช้แฟ้มแบบสุ่ม

```
1:   import java.io.*;
2:
3:   class TestRandomAccessFile {
4:       public static void main(String[] args) {
5:           if (args.length != 1) {
6:               System.err.println(
7:                   "Usage: java TestRandomAccessFile <output file>");
8:               System.exit(-1);
9:           }
10:          try {
11:              RandomAccessFile raf = new RandomAccessFile(args[0], "rw");
12:              char a = 'a';
13:              byte b = 2;
```

```
14:             String c = "abc";
15:             short d = 4;
16:             byte[] b2 = {(byte)'a', (byte)'b',(byte) 'c'};
17:
18:             // write some stuff out
19:             long file_start = raf.getFilePointer();
20:             raf.write(b);
21:             raf.write(b2, 0, b2.length);
22:             raf.writeBoolean(true);
23:             raf.writeChar(a);
24:             raf.writeBytes(c);
25:             raf.writeChars(c);
26:             raf.writeDouble(123.456);
27:             raf.writeFloat(123.456f);
28:             raf.writeInt(678);
29:             raf.writeLong(678l);
30:             raf.writeShort(d);
31:             raf.writeUTF(c);
32:             raf.writeUTF("abc\n");
33:             raf.write(b);
34:             raf.writeShort(d);
35:
36:             System.out.println("Length of file: " + raf.length());
37:
38:             // read the stuff back
39:             raf.seek(file_start);
40:
41:             b2 = new byte[1];
42:             b = raf.readByte();
43:             System.out.println("Byte: " + b);
44:             raf.read(b2);
45:             System.out.println("Byte[0]: " + (char)b2[0]);
46:             raf.read(b2, 0, b2.length);
47:             System.out.println("Byte[0]: " + (char)b2[0]);
48:             int ub = raf.readUnsignedByte();
49:             System.out.println("Unsigned Byte: " + b);
50:             System.out.println("Boolean: " + raf.readBoolean());
51:             a = raf.readChar();
52:             System.out.println("Char: " + a);
53:
54:             byte[] b3 = new byte[3];
55:             raf.readFully(b3);
56:             System.out.println("readFully: " + (char)b3[0] + (char)b3[1] +
57:                                (char)b3[2]);
58:             raf.skipBytes(6); // skip string 'abc'
59:             double d1 = raf.readDouble();
60:             float f1 = raf.readFloat();
61:             int i = raf.readInt();
62:             long l = raf.readLong();
63:             short s = raf.readShort();
64:             String str = raf.readUTF();
65:             ub = raf.readUnsignedByte();
66:             int us = raf.readUnsignedShort();
67:             System.out.println("UTF String" + str);
68:
69:             raf.close();
70:         } catch (IOException e) {
71:             System.err.println(e);
72:         }
73:     }
74: }
```

ตัวอย่างที่ 9 นี้แสดงการใช้คลาส RandomAccessFile ซึ่งสามารถอ่านและเขียนข้อมูลแบบสุ่มได้ในคลาสเดียว ตัวอย่างนี้นำโปรแกรมจากตัวอย่างที่ ๔ และ ๕ มาผนวกรวมกัน โดยช่วงต้นจะเป็นการเขียนข้อมูลลงไฟล์ แล้วอ่านข้อมูลกลับคืนมาในช่วงท้าย แต่ก่อนที่จะอ่านกลับคืนมาจะเปลี่ยนตำแหน่ง file pointer ให้กลับไปเริ่มต้นใหม่เนื่องจากเมื่อมีการอ่านหรือเขียน file pointer จะขยับออกไปตามจำนวนไบต์ที่อ่านหรือเขียนเสมอเพื่อเตรียมพร้อมที่จะอ่านหรือเขียนข้อมูลในครั้งต่อไป ในส่วนแรกของโปรแกรมนี้เป็นการเขียนข้อมูล ต่อมาต้องการอ่านข้อมูลที่เขียนไปแล้วนั้นกลับคืนมาจึงจำเป็นต้องเปลี่ยน file pointer ให้ชี้กลับไปตำแหน่งจุดเริ่มต้นอีกครั้งด้วยเมท็อด seek ตรงบรรทัด ๓๙ ซึ่งก่อนหน้านี้ตรงบรรทัด ๑๙ มีการเก็บค่าของ file pointer ไว้ ซึ่ง ณ ขณะนั้นยังไม่มีการอ่านหรือเขียนข้อมูล ตำแหน่งของ file pointer จะชี้ไปยังจุดเริ่มต้นของแฟ้ม หลังจากมีการเขียนข้อมูลไปแล้ว ตำแหน่งของ file pointer จะเปลี่ยนไป แต่เราเก็บค่าของ file pointer ตรงตำแหน่งเริ่มต้นไว้ในตัวแปร file_start แล้ว ซึ่งนำค่าของมันไปใช้ในการ seek ตรงบรรทัดที่ ๓๙ เพื่อให้ file pointer ชี้กลับไปยังจุดเริ่มต้นอีกครั้ง ความจริงถ้าต้องการให้ file pointer เปลี่ยนกลับไปยังจุดเริ่มต้นแฟ้ม สามารถสั่ง seek(0) ได้เลยไม่จำเป็นต้องเก็บค่า file pointer ไว้ในตัวแปร file_start เนื่องจากจุดเริ่มต้นแฟ้มมีความ file pointer เป็นศูนย์ แต่ตัวอย่างนี้ต้องการสาธิตการสอบถามค่าของ file pointer ด้วยเมท็อด getFilePointer() จึงทำเช่นนี้

ตัวอย่างที่ 10 การประยุกต์ใช้แฟ้มแบบสุ่มกับการเก็บข้อมูลบัญชีเงินฝาก

```
1:   import java.io.*;
2:   import java.lang.NumberFormatException;
3:   class Account {
4:    final static int ACCOUNT_NAME_SIZE = 20;
5:    int accountNumber;
6:    String accountName;
7:    double balance;
8:
9:    Account(int accountNumber, String accountName) {
10:      this.accountNumber = accountNumber;
11:      this.accountName = accountName;
12:   }
13:
14:   Account(RandomAccessFile file) {
15:      try {
16:         accountNumber = file.readInt();
17:         byte buff[] = new byte[ACCOUNT_NAME_SIZE];
18:         file.read(buff);
19:         String name = new String(buff);
20:         accountName = name;
21:         balance = file.readDouble();
22:      }
23:      catch (IOException e) {
24:         System.err.println("Error : cannot read data from file");
25:      }
26:   }
27:
28:   void deposit(double amount) {
29:      balance += amount;
30:   }
31:
```

```
32:   public String toString() {
33:      StringBuffer sb = new StringBuffer(60);
34:
35:      sb.append(accountNumber);
36:      sb.append(" ");
37:      sb.append(accountName);
38:      sb.append(" balance is ");
39:      sb.append(balance);
40:      return sb.toString();
41:   }
42:  }
43:
44:
45:  class AccountFile {
46:   /*
47:   record structure
48:   int account number    length 4
49:   byte account name    length 20
50:   double balance       length 8
51:   */
52:   final static int recordLength = 4 + Account.ACCOUNT_NAME_SIZE + 8;
53:   static int NextAccountNumber = 0;
54:   static RandomAccessFile af;
55:
56:   public static void main(String args[]) {
57:      boolean isStillWorking = true;
58:
59:      System.out.println("Welcome to First Java Bank\n");
60:      try {
61:         af = new RandomAccessFile("BankAccount.data","rw");
62:         long fileLength = af.length();
63:         if ( fileLength > 0 )
64:            NextAccountNumber = (int)(fileLength / recordLength);
65:         System.out.println("Next new account number will be " +
    NextAccountNumber);
66:
67:         while (isStillWorking) {
68:            System.out.println("\nMain menu");
69:            System.out.println("N) open a New account");
70:            System.out.println("D) Deposit");
71:            System.out.println("X) eXit program");
72:            prompt("Please select an item from menu : ");
73:
74:            switch (getMenuItem())
75:            {
76:               case 'N' : OpenNewAccount(); break;
77:               case 'D' : Deposit(); break;
78:               case 'X' : isStillWorking = false; break;
79:               default :  System.out.println("Error : Invalid choice");
80:            }
81:         }
82:         af.close();
83:      } catch (IOException e) {
84:         System.err.println("Error : cannot open account file");
85:         System.exit(-1);
```

```
86:       }
87:       System.out.println("Bye");
88:    }
89:
90:    static void OpenNewAccount() {
91:       // int ac_number=0;
92:       String name;
93:       prompt("Enter account name : ");
94:       name = readLine();
95:
96:       Account newAcc = new Account(NextAccountNumber++, name);
97:
98:       System.out.println(newAcc);
99:       try {
100:         af.seek(newAcc.accountNumber * recordLength);
101:         af.writeInt( newAcc.accountNumber);
102:         byte buff[] = new byte[Account.ACCOUNT_NAME_SIZE];
103:         buff = newAcc.accountName.getBytes();
104:         for (int i=newAcc.accountName.length(); i < buff.length; i++)
105:            buff[i] = (byte)' ';  // clear buffer to space
106:         af.write(buff);
107:         af.writeDouble(newAcc.balance);
108:      } catch (IOException e) {
109:         System.out.println("Error : cannot write account record to file");
110:      }
111:   }
112:
113:   static void Deposit() {
114:      int ac_number = getAccountNumber();
115:          try {
116:         af.seek(ac_number * recordLength);
117:      } catch (IOException e) {
118:         System.err.println("Error : cannot locate record");
119:      }
120:      Account acc = new Account(af);
121:
122:
123:      //  ... other stuff
124:
125:
126:      System.out.println(acc);
127:
128:   }
129:
130:
131:   static String readLine() {
132:      StringBuffer sb = new StringBuffer(80);
133:      char ch;
134:      try {
135:
136:         while ( ( ch=(char)System.in.read() ) != -1)    {
137:            if ( Character.isSpaceChar(ch) )
138:               continue;  // omit leading whitespace
139:            else
140:               break;
```

```
141:         }
142:
143:         do  {
144:            if (ch == '\n')
145:               break;
146:            sb.append(ch);
147:         } while ( ( ch = (char) System.in.read() ) != -1);
148:      } catch (IOException e) {
149:         System.err.println(e);
150:      }
151:      return sb.toString();
152:   }
153:
154:   static int getAccountNumber() {
155:      Integer iac= null;
156:      prompt("Please enter account number (0 - "+ (NextAccountNumber -1)+"):
     ");
157:
158:      try {
159:         iac = new Integer( readLine() );
160:      } catch (NumberFormatException e) {
161:         System.err.println("Account number must be a number");
162:      }
163:      return iac.intValue();
164:   }
165:
166:   static char getMenuItem() {
167:      char keyin='X';
168:      try {
169:         keyin = (char)System.in.read();
170:         while ( System.in.read() != '\n'); // omit the rest until newline
171:      } catch (IOException e)  {
172:         System.err.println("Error while get menu item");
173:      }
174:      return Character.toUpperCase(keyin);
175:   }
176:
177:   static void prompt(String s) {
178:      System.out.print(s);
179:      System.out.flush();
180:      return;
181:   }
182: }
```

ตัวอย่างนี้แสดงการประยุกต์ใช้การอ่านและเขียนแฟ้มข้อมูลแบบสุ่ม โดยสมมติการใช้งานกับการฝากและถอนเงินบัญชีธนาคาร ตัวอย่างนี้ค่อนข้างยาวจะขออธิบายในหลักการมากกว่ารายละเอียด

เนื่องจากต้องการอ่านข้อมูลแบบสุ่ม นั่นคือจะอ่านระเบียนใดก่อนหลังอย่างไรก็ได้ และต้องการออกแบบแฟ้มข้อมูลให้ง่ายต่อการใช้งานจึงออกแบบระเบียนให้มีขนาดตายตัวเพื่อที่จะได้คำนวณหาตำแหน่งของระเบียนได้ง่าย และใช้กำหนดเลขที่บัญชีให้สัมพันธ์กับเลขที่ระเบียนโดยตรงเพื่อง่ายในการนำเลขที่บัญชีมาคำนวณแปลงเป็นเลขที่ระเบียน

โปรแกรมนี้ค่อนข้างยาว อาจารย์จะปรับปรุงใหม่ให้ทำความเข้าใจง่ายกว่านี้และทำให้เป็นโปรแกรมแบบ GUI แล้วจะ post ไว้ในเว็บไซต์ต่อไป

ตัวอย่างที่ 11 การใช้ ObjectOutputStream ในการเขียน object ลงแฟ้ม

```
1:  import java.io.ObjectOutputStream;
2:  import java.io.FileOutputStream;
3:  import java.io.IOException;
4:
5:  class TestObjectOutputStream {
6:      public static void main (String args[]) {
7:      String filename = "TestObjectOutputStream.dat";
8:      String s = "Hello Thailand";
9:      try {
10:        FileOutputStream ostream = new FileOutputStream(filename);
11:        ObjectOutputStream out = new ObjectOutputStream(ostream);
12:        out.writeObject(s);
13:        out.close();
14:     } catch (IOException e) {
15:        e.printStackTrace();
16:     }
17:   }
18: }
```

ตัวอย่างที่ 12 การใช้ ObjectInputStream ในการอ่าน object จากแฟ้ม

```
1:  import java.io.ObjectInputStream;
2:  import java.io.FileInputStream;
3:  import java.io.IOException;
4:  class TestObjectInputStream {
5:   public static void main (String args[]) {
6:      String filename = "TestObjectOutputStream.dat";
7:      String s;
8:      try {
9:         FileInputStream istream = new FileInputStream(filename);
10:        ObjectInputStream in = new ObjectInputStream(istream);
11:        s = (String)in.readObject();
12:        System.out.println(s);
13:        in.close();
14:     } catch (ClassNotFoundException e) {
15:        e.printStackTrace();
16:     } catch (IOException e) {
17:        e.printStackTrace();
18:     }
19:   }
20: }
```

การใช้ เมท็อด **readObject** จำเป็นต้องดักจับสิ่งผิดปรกติ `ClassNotFoundException` ดังนั้นจึงต้องเขียน `catch` block สำหรับดักจับสิ่งผิดปรกติชนิดนี้ดังตัวอย่างบรรทัดที่ 15

ตัวอย่างที่ 13 การเขียน object คลาส Date ลงแฟ้ม

```
1:   import java.io.ObjectOutputStream;
2:   import java.io.FileOutputStream;
3:   import java.io.IOException;
4:   import java.util.*;
5:   class ObjectOutputDemo {
6:    public static void main(String [] args) {
7:       try {
8:          FileOutputStream fileOut = new FileOutputStream("tab");
9:          ObjectOutputStream out = new ObjectOutputStream(fileOut);
10:         Date today = new Date();
11:         out.writeObject(today);
12:         out.close();
13:         System.out.println(today);
14:      }
15:      catch (IOException e) {
16:         e.printStackTrace();
17:      }
18:   }
19:  }
```

ดูตัวอย่างและคำอธิบายเพิ่มเติมเกี่ยวกับ Serialization ได้ในชีท serialization