Chapter 10 File I/O

1. Stream: an object that enables the flow of data between a program and some I/O device or file
   1. input stream: data flows into a program
      1. System.in: connects to keyboard

Scanner keyboard = new Scanner(System.in);

* 1. output stream: data flows out of a program
     1. System.out: connects to screen

System.out.println(Output stream“);

1. Text Files(“ASCII files”): readable by human
2. Binary Files: sequence of binary digits
   1. more efficient than text files
3. Writing to a Text File: class PrintWriter
   1. stream class that can write to a text file
   2. has print and println
   3. need to import the followings:

import java.io.PrintWriter;

import java.io.FileOutputStream;

import java.io.FileNotFoundException;

* 1. PrintWrite has no constructor to take file name, so uses FileOutputStream to convert a file name to an object that can be used as the argument to PrintWriter constructor
  2. for example

/\* 1. Open file.

2. If file already exists, replace it.

3. If file doesn’t exist, create it.

4. nested constructor invocations on right hand side\*/

PrintWriter outputStreamName = new PrintWriter(new

FileOutputStream(FileName));

* 1. close the stream when finish
     1. outputStreamName.close();
     2. release any resources used to connect the stream to the file
     3. Java automatically closes it when program ends.
     4. Programmer should explicitly close it.
  2. streams are buffered
     1. data is saved in a temporary location (buffer) instead of writing to the file ASAP
        1. Because I/O devices are slow. 累積一定程度再一次輸出比較有效率
     2. buffered data is written to file all at once, when
        1. enough data accumulates
        2. method flush is invoked: insuring all data is written to the file

1. file name
   1. suffix(.txt, .exe, ...) no meaning to Java program
   2. every input file and output file used by program has two names
      1. real file name (used by the operating system)
      2. stream name (connected to the file)
2. IOException
   1. root class for input/output exceptions, e.g. FileNotFoundException
   2. all are checked exceptions (must be caught)
3. Unchecked Exception
   1. not required to be caught
   2. for example, NoSuchElementException, InputMismatchException, and IllegalStateException
4. Appending to a Text File
   1. syntax:

PrintWriter outputStreamName = new PrintWriter(new

FileOutputStream(FileName,true));

1. toString helps with Text File Output
   1. if a class has toString() method, it can be used as an argument to System.out.println directly

//no anObject.toString() required

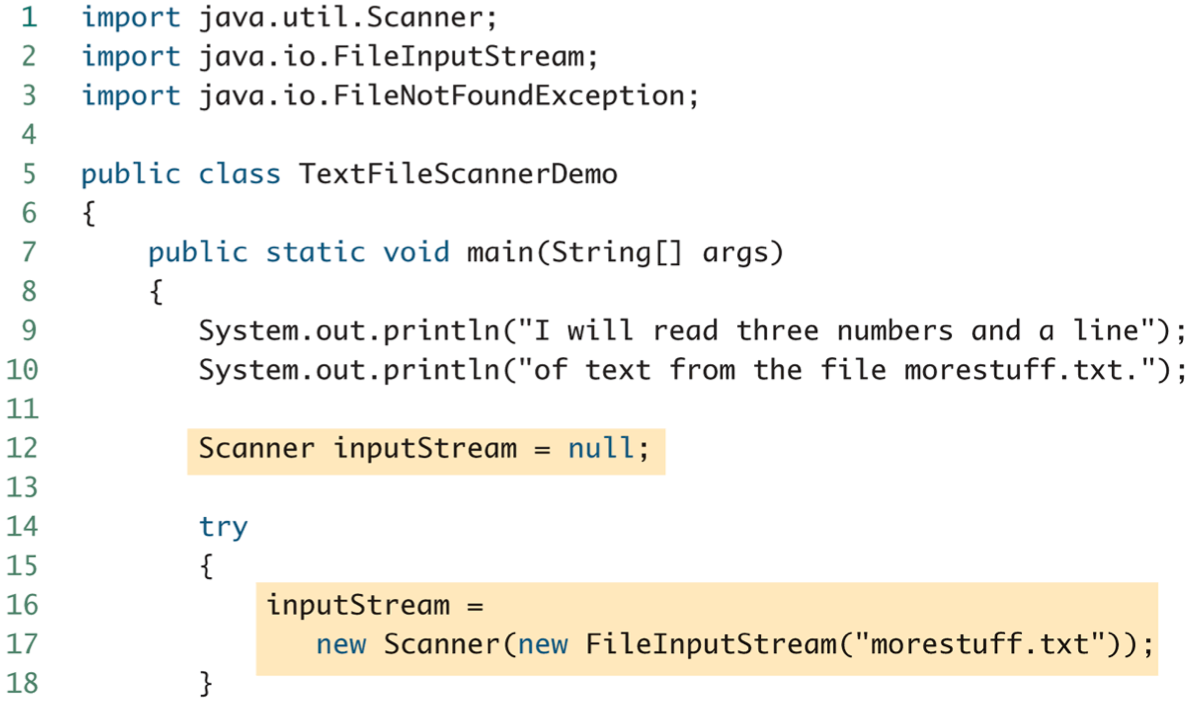
outputStreamName.println(anObject)

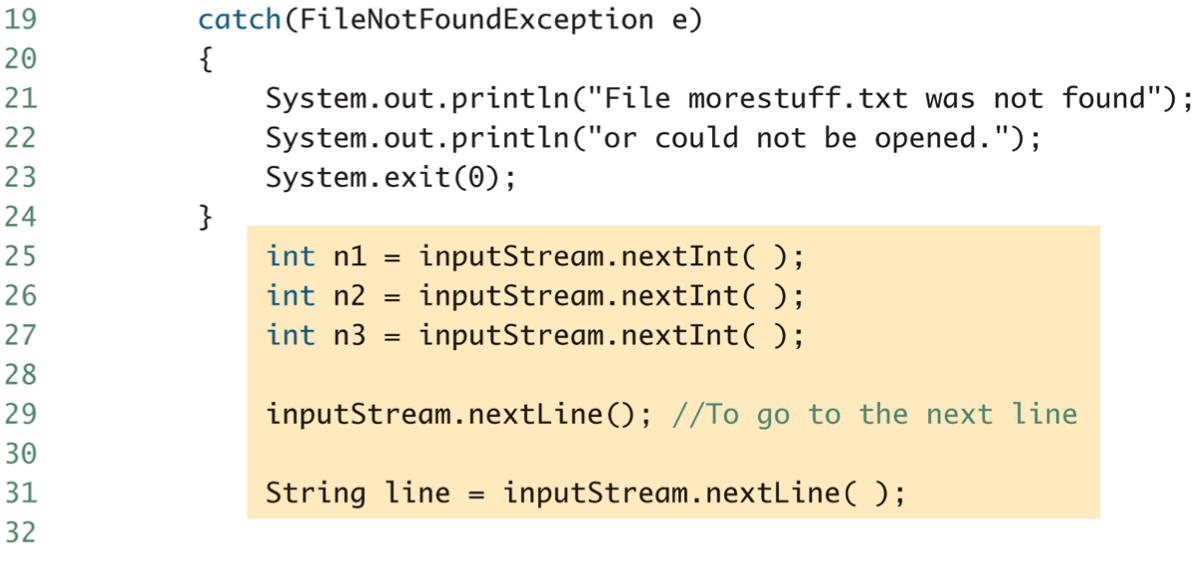
1. Reading from a text file: Scanner
   1. syntax:

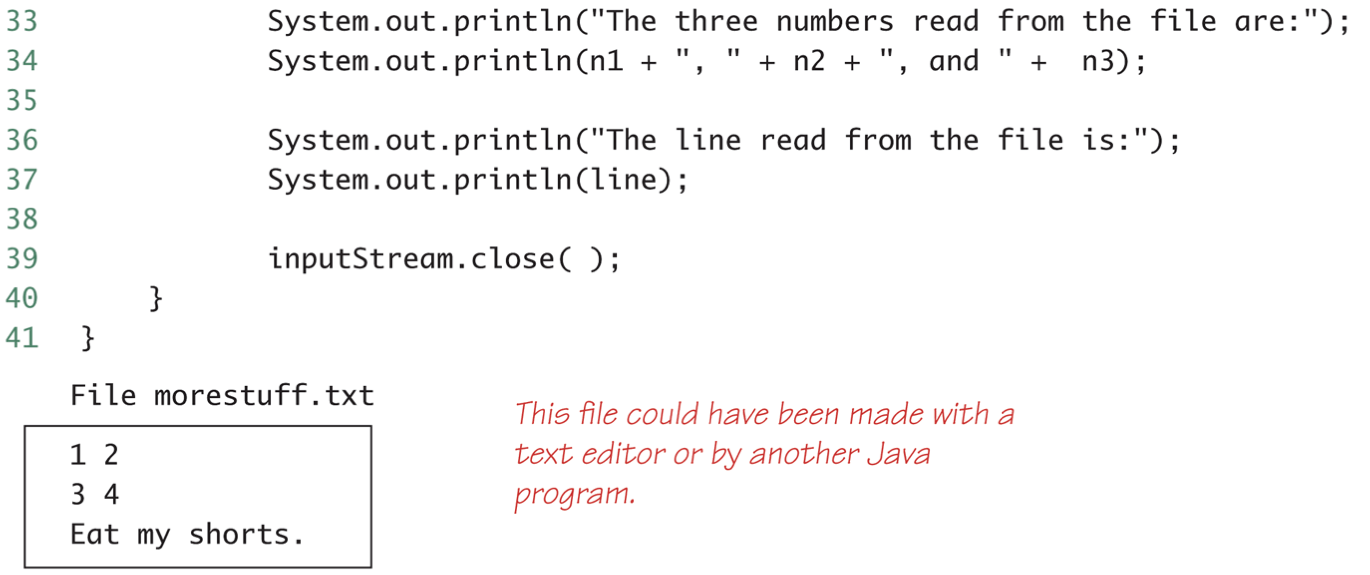
Scanner StreamObject = new Scanner(new

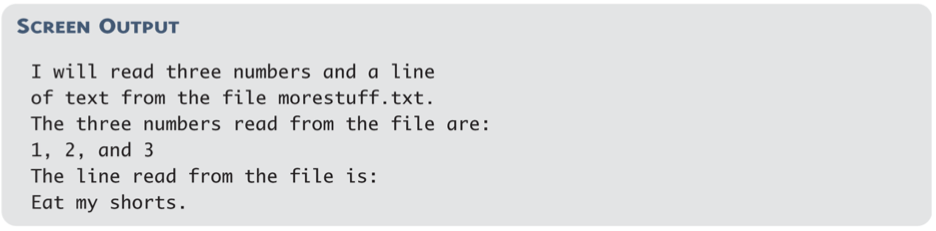
FileInputStream(FileName));

* 1. has nextInt and nextLine methods

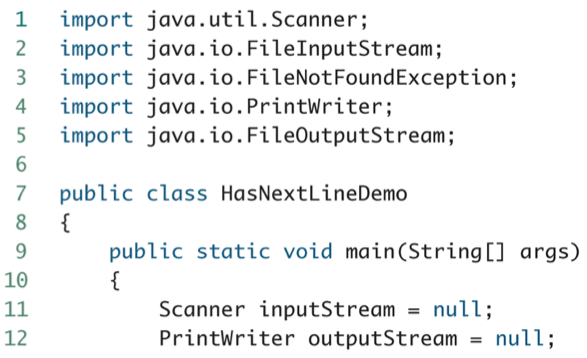


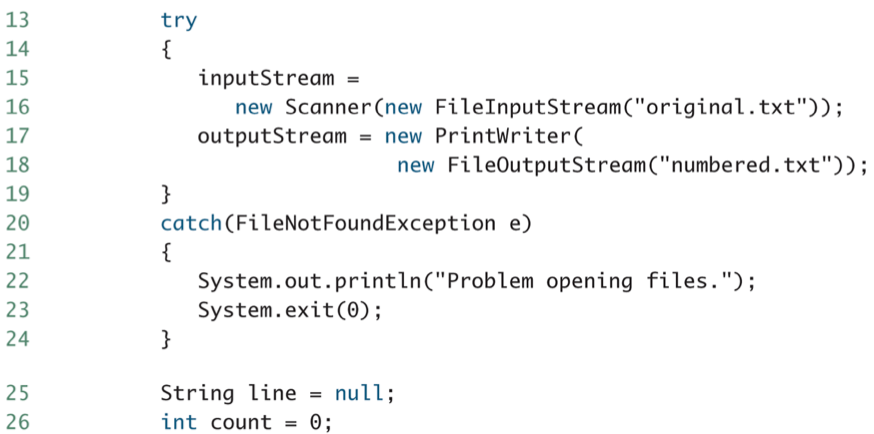


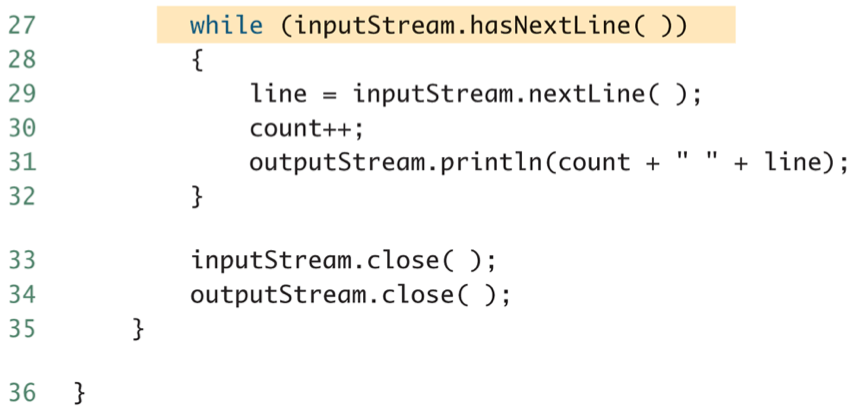




* 1. check end of text







1. Reading from a text file: BufferedReader
   1. import java.io.BufferedReader;

import java.io.FileReader;

import java.io.FileNotFoundException;

import java.io.IOException;

* 1. has read and readLine method
     1. read reads a single character, and returns type int.
        1. Can use type cast:

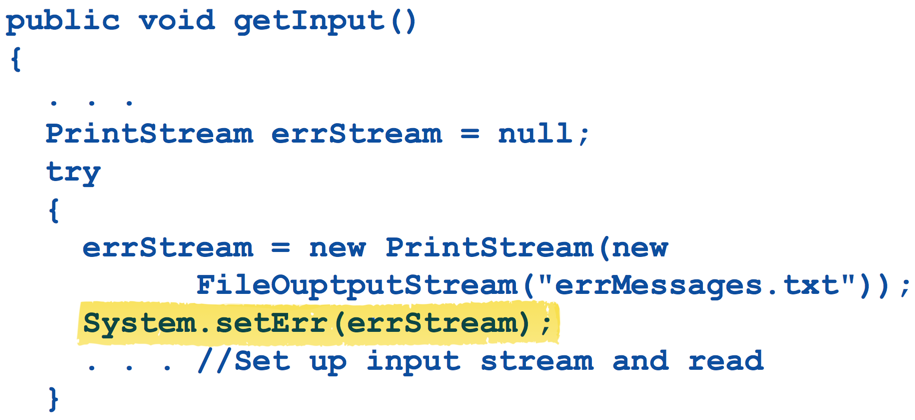
char next = (char) (readerObject.read());

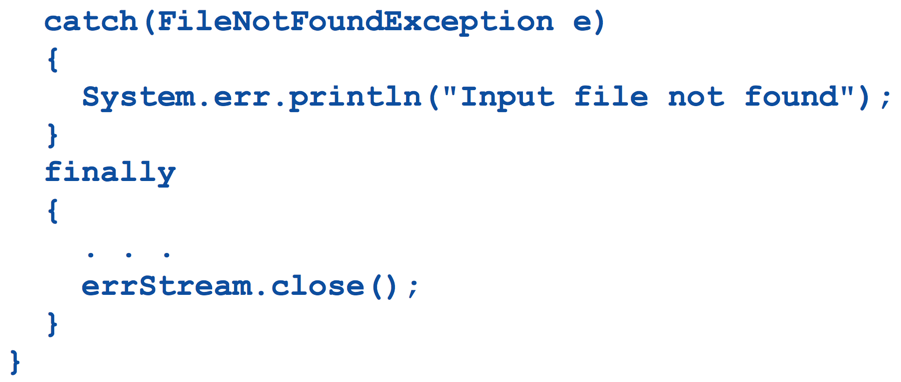
* + - 1. returns -1 when end of file
    1. readLine returns null when end of file
  1. BufferedReader readerObject = new BufferedReader(new

FileReader(FileName));

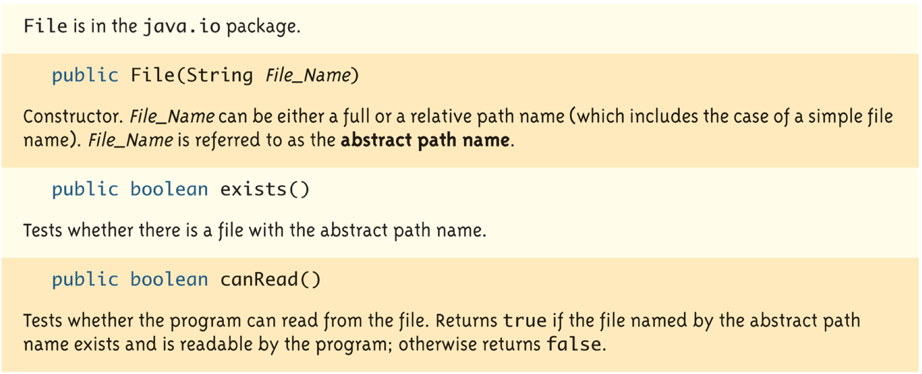
* 1. very similar to Scanner
  2. can’t read a number from text

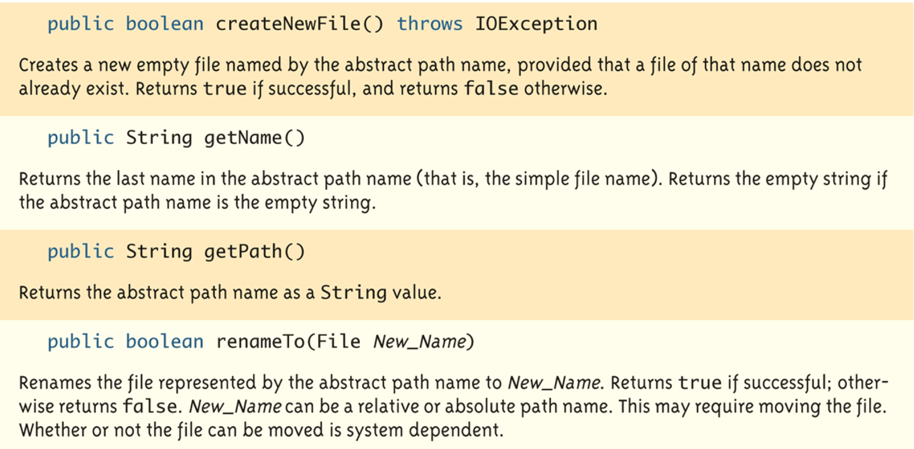
1. Path Names: must be used when file not in the same directory
   1. full path name
   2. relative path name
2. Class System
   1. System.in
   2. System.out: normal screen output
   3. System.err: error messages to the screen
   4. redirecting standard streams:
      1. public static void setIn(InputStream inStream)
      2. public static void setOut(PrintStream outStream)
      3. public static void setErr(PrintStream outStream)
      4. For example, instead of appearing on the screen, error messages could be redirected to a file. A new stream object should be created
      5. Standard streams no need to be closed





1. File class
   1. a wrapper class for file names
   2. can be used to determined information information about the file
   3. constructor and method examples





1. Writing Simple Data to a Binary File: ObjectOutputStream
   1. similar to PrintWriter class
2. Random Access to Binary File: RandomAccessFile
   1. for fast access in very large databases
   2. read and write to the same file
   3. has file pointer