Java Programming: From Problem Analysis to Program Design

Chapter 3

Introduction to Objects and the String Class

Chapter Objectives

- Learn about objects and reference variables
- Explore how to use predefined methods in a program
- Become familiar with the `class String`

Input/Output

- Input Data
- Format Output
- Output Results
- Format Output
- Read From and Write to Files

Parsing Numeric Strings

- A string consisting of only integers or decimal numbers is called a numeric string
- 1. To convert a string consisting of an integer to a value of the type `int`, we use the following expression:

```
Integer.parseInt(strExpression)

Integer.parseInt("6723") = 6723
Integer.parseInt("-823") = -823
```

Parsing Numeric Strings (continued)

- 2. To convert a string consisting of a decimal number to a value of the type float, we use the following expression:

```
Float.parseFloat(strExpression)

Float.parseFloat("34.56") = 34.56
Float.parseFloat("-542.97") = -542.97
```

Parsing Numeric Strings (continued)

- 3. To convert a string consisting of a decimal number to a value of the type double, we use the following expression:
  ```
  Double.parseDouble(strExpression)

  Double.parseDouble("345.78") = 345.78
  Double.parseDouble("-782.873") = -
  782.873
  ```

Parsing Numeric Strings (continued)

- `Integer`, `Float`, and `Double` are classes designed to convert a numeric string into a number
- These classes are called **wrapper** classes
- `parseInt` is a method of the `class` `Integer`, which converts a numeric integer string into a value of the type `int`

Parsing Numeric Strings (continued)

- `parseFloat` is a method of the `class Float` and is used to convert a numeric decimal string into an equivalent value of the type `float`
- `parseDouble` is a method of the `class Double`, which is used to convert a numeric decimal string into an equivalent value of the type `double`

Using Dialog Boxes for Input/Output

- Use a graphical user interface (GUI)
- `class JOptionPane`
  - Contained in package `javax.swing`
  - Contains methods: `showInputDialog` and `showMessageDialog`
- Syntax:

```
str = JOptionPane.showInputDialog(strExpression)
```

- Program must end with `System.exit(0);`

Parameters for the Method `showMessageDialog`

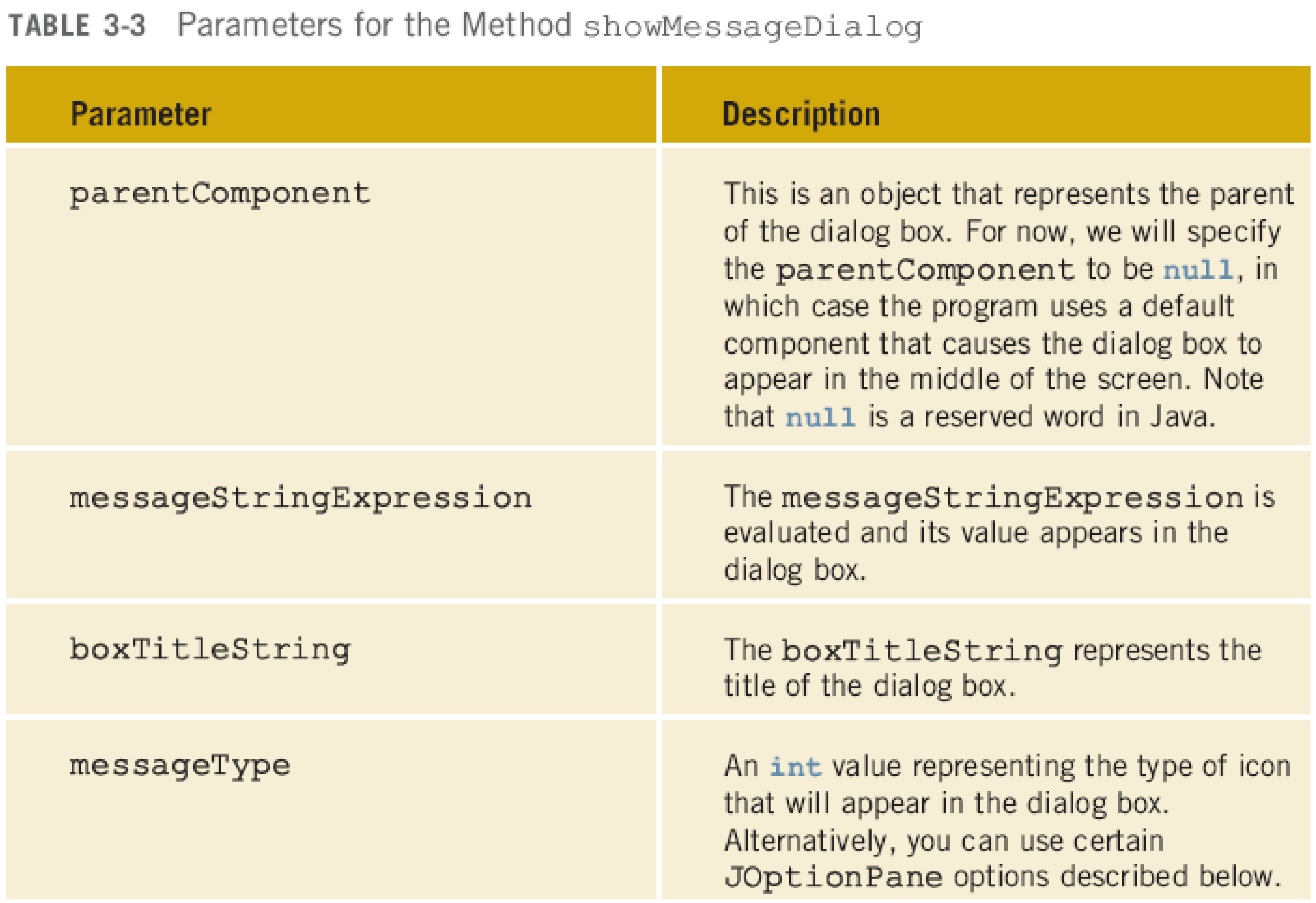


TABLE 3-3 Parameters for the Method `showMessageDialog`

Parameter	Description
`parentComponent`	This is an object that represents the parent of the dialog box. For now, we will specify the `parentComponent` to be `null`, in which case the program uses a default component that causes the dialog box to appear in the middle of the screen. Note that `null` is a reserved word in Java.
`messageStringExpression`	The `messageStringExpression` is evaluated and its value appears in the dialog box.
`boxTitleString`	The `boxTitleString` represents the title of the dialog box.
`messageType`	An `int` value representing the type of icon that will appear in the dialog box. Alternatively, you can use certain `JOptionPane` options described below.

`JOptionPane` Options for the Parameter `messageType`

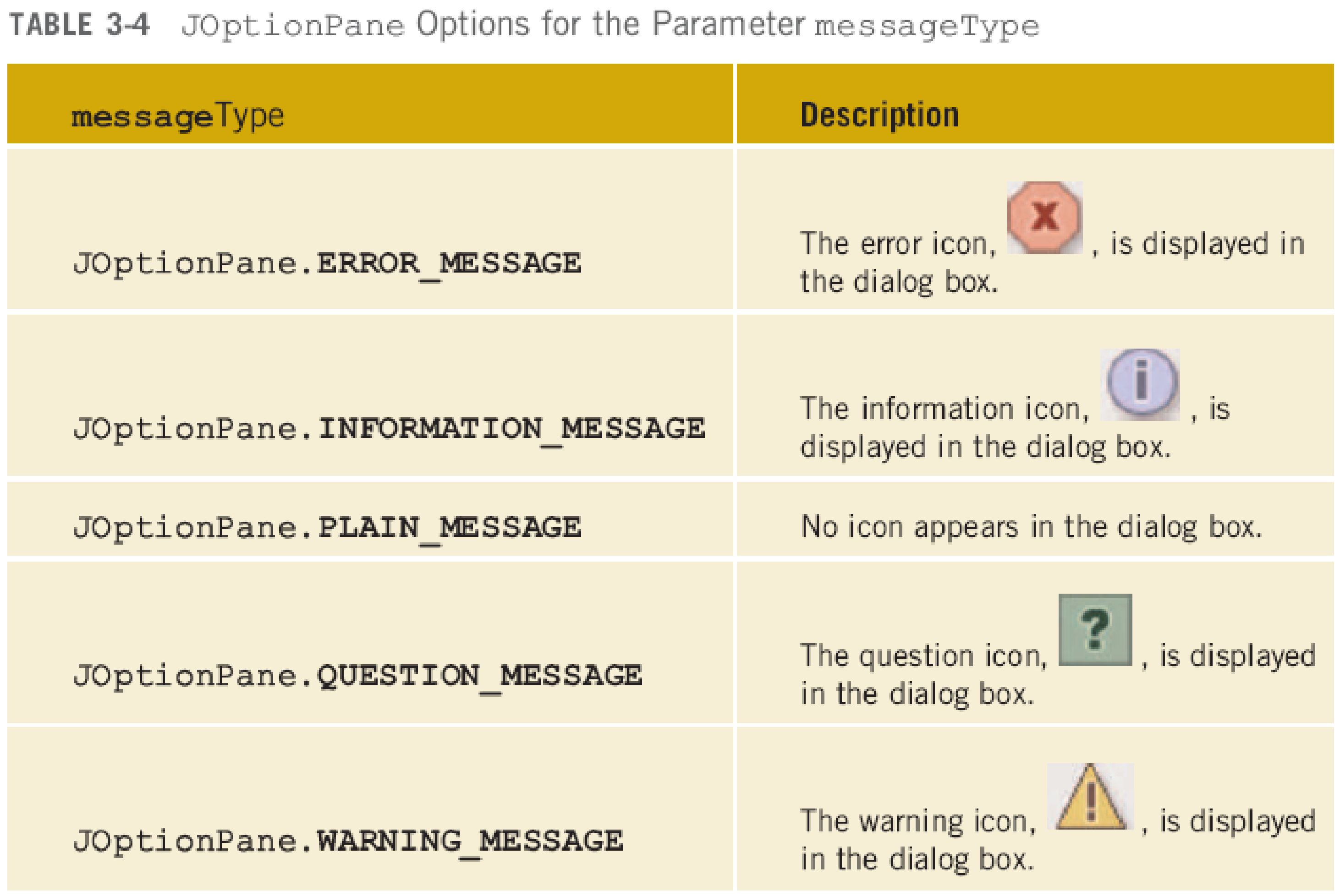


TABLE 3-4 `JOptionPane` Options for the Parameter `messageType`

`messageType`	Description
`JOptionPane.ERROR_MESSAGE`	The error icon, ✖, is displayed in the dialog box.
`JOptionPane.INFORMATION_MESSAGE`	The information icon, ℹ, is displayed in the dialog box.
`JOptionPane.PLAIN_MESSAGE`	No icon appears in the dialog box.
`JOptionPane.QUESTION_MESSAGE`	The question icon, ?, is displayed in the dialog box.
`JOptionPane.WARNING_MESSAGE`	The warning icon, ⚠, is displayed in the dialog box.

`JOptionPane` Example

The output of the statement

```
JOptionPane.showMessageDialog(null, "Hello World!", "Greetings",
                              JOptionPane.INFORMATION_MESSAGE);
```

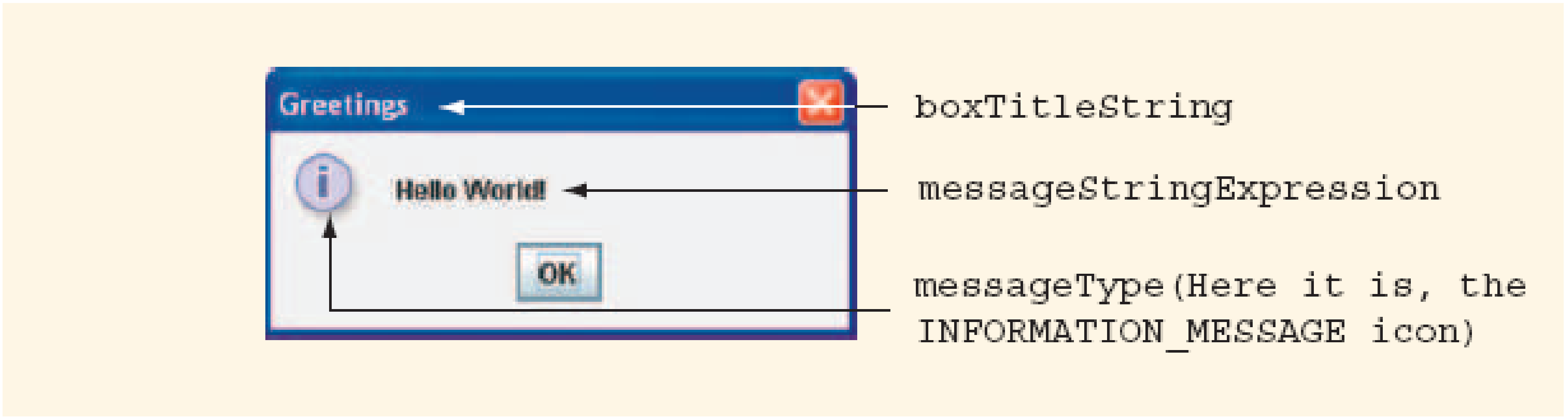


FIGURE 3-9 Message dialog box showing its various components

Formatting the Output Using the `String` Method `format`

Example 3-14

```
double x = 15.674;
double y = 235.73;
double z = 9525.9864;
int num = 83;
String str;
```

Expression	Value
`String.format("%.2f", x)`	`"15.67"`
`String.format("%.3f", y)`	`"235.730"`
`String.format("%.2f", z)`	`"9525.99"`
`String.format("%7s", "Hello")`	`"  Hello"`
`String.format("%5d%7.2f", num, x)`	`"   83  15.67"`
`String.format("The value of num = %5d", num)`	`"The value of num = 83"`
`str = String.format("%.2f", z)`	`str = "9525.99"`

File Input/Output

- File: area in secondary storage used to hold information
- You can also initialize a `Scanner` object to input sources other than the standard input device by passing an appropriate argument in place of the object `System.in`.
- We make use of the `class FileReader`.

File Input/Output (continued)

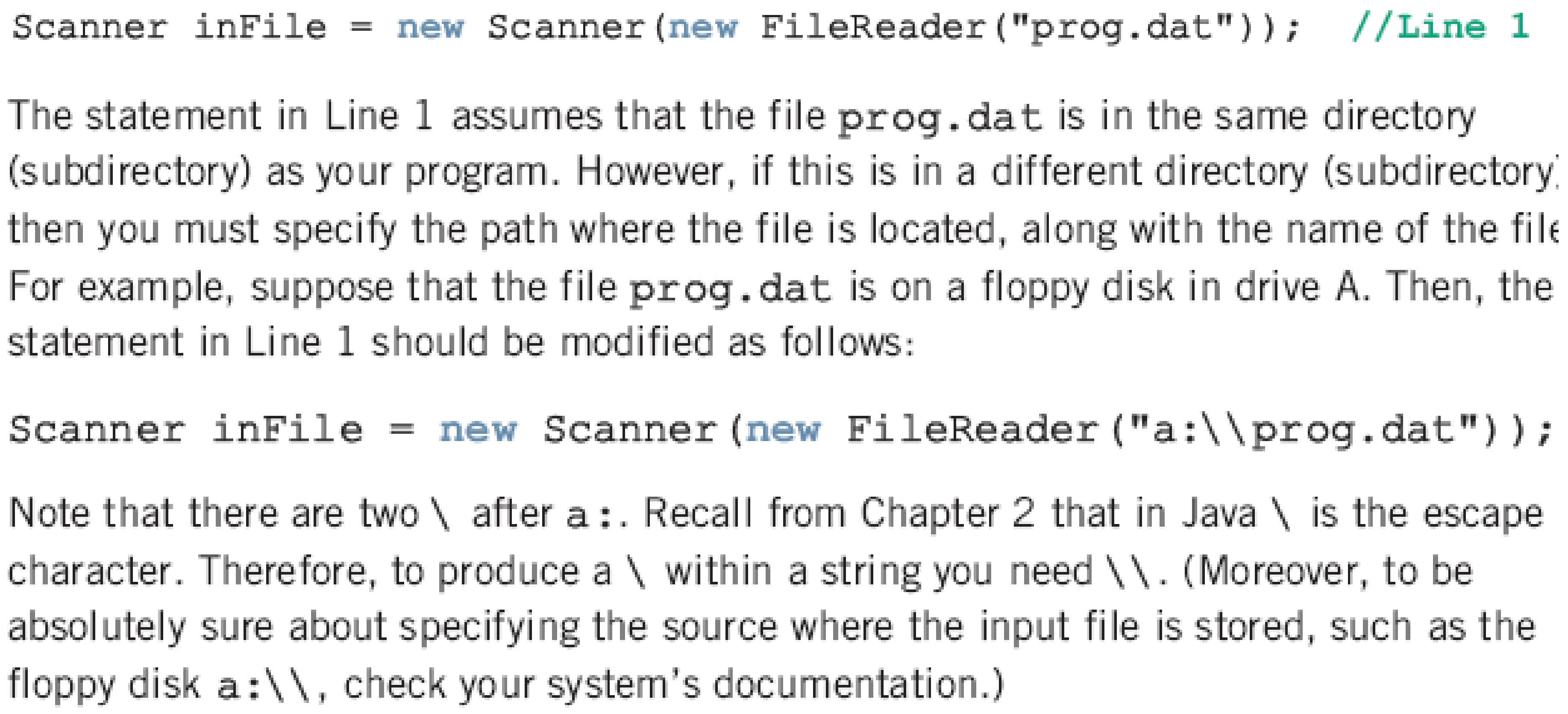
- Suppose that the input data is stored in a file, say `prog.dat`, and this file is on the floppy disk `A`
- The following statement creates the `Scanner` object `inFile` and initializes it to the file `prog.dat`

-
  ```
  Scanner inFile = new Scanner
               (new FileReader("prog.dat"));
  ```

File Input/Output (continued)

- Next, you use the object `inFile` to input data from the file `prog.dat` just the way you used the object `console` to input data from the standard input device using the methods `next`, `nextInt`, `nextDouble`, and so on

File Input/Output (continued)



```
Scanner inFile = new Scanner(new FileReader("prog.dat"));  //Line 1
```

The statement in Line 1 assumes that the file `prog.dat` is in the same directory (subdirectory) as your program. However, if this is in a different directory (subdirectory then you must specify the path where the file is located, along with the name of the fil For example, suppose that the file `prog.dat` is on a floppy disk in drive A. Then, the statement in Line 1 should be modified as follows:

```
Scanner inFile = new Scanner(new FileReader("a:\\prog.dat"));
```

Note that there are two `\` after `a:`. Recall from Chapter 2 that in Java `\` is the escape character. Therefore, to produce a `\` within a string you need `\\`. (Moreover, to be absolutely sure about specifying the source where the input file is stored, such as the floppy disk `a:\\`, check your system's documentation.)

File Input/Output (continued)

- Java file I/O process
  1. Import necessary classes from the packages `java.util` and `java.io` into the program
  2. Create and associate appropriate objects with the input/output sources
  3. Use the appropriate methods associated with the variables created in Step 2 to input/output data
  4. Close the files

File Input/Output (continued)

Example 3-17

Suppose an input file, say `employeeData.txt`, consists of the following data:

```
Emily Johnson 45 13.50

Scanner inFile = new Scanner
      (new FileReader("employeeData.txt"));
String firstName;
String lastName;
double hoursWorked;
double payRate;
double wages;
firstName = inFile.next();
lastName = inFile.next();
hoursWorked = inFile.nextDouble();
payRate = inFile.nextDouble();
wages = hoursWorked * payRate;

inFile.close();  //close the input file
```

Storing (Writing) Output to a File

- To store the output of a program in a file, you use the `class PrintWriter`
- Declare a `PrintWriter` variable and associate this variable with the destination
- Suppose the output is to be stored in the file `prog.out` on floppy disk A

Storing (Writing) Output to a File (continued)

- Consider the following statement:

```
PrintWriter outFile = new
PrintWriter("prog.out");
```

- This statement creates the `PrintWriter` object `outFile` and associates it with the file `prog.out`
- You can now use the methods `print`, `println`, `printf`, and `flush` with `outFile` just the same way they have been used with the object `System.out`

Storing (Writing) Output to a File (continued)

- The statement:

```
outFile.println("The paycheck is: $" + pay);
```

stores the output—`The paycheck is: $565.78`—in the file `prog.out`

-This statement assumes that the value of the variable `pay` is `565.78`

Storing (Writing) Output to a File (continued)

- Step 4 requires closing the file; you close the input and output files by using the method close

```
inFile.close();
outFile.close();
```

- Closing the output file ensures that the buffer holding the output will be emptied, that is, the entire output generated by the program will be sent to the output file

`throws` clause

- During program execution, various things can happen; for example, division by zero or inputting a letter for a number
- In such cases, we say that an exception has occurred.
- If an exception occurs in a method, the method should either handle the exception or *throw* it for the calling environment to handle
- If an input file does not exist, the program throws a **FileNotFoundException**

`throws` clause (continued)

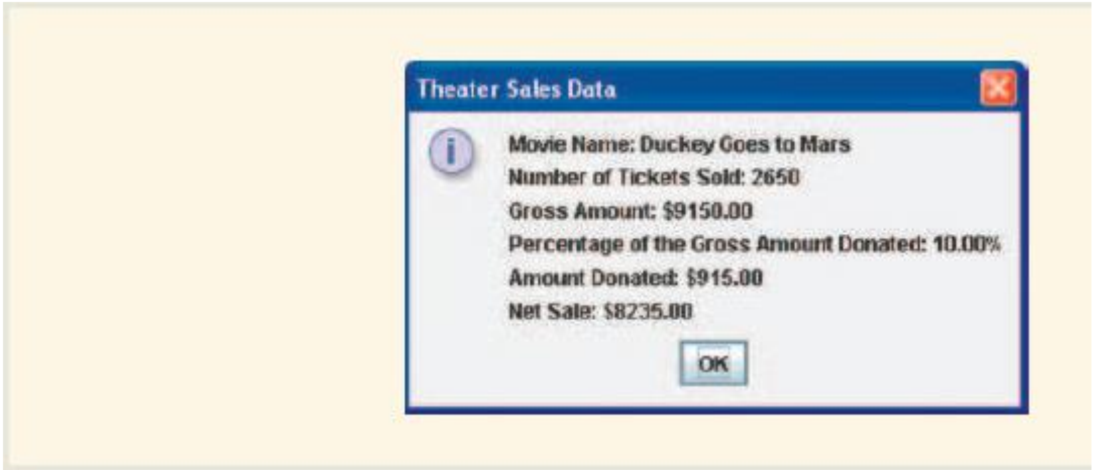
- If an output file cannot be created or accessed, the program throws a **FileNotFoundException**
- For the next few chapters, we will simply throw the exceptions
- Because we do not need the method `main` to handle the **FileNotFoundException** exception, we will include a command in the heading of the method `main` to throw the **FileNotFoundException** exception

Skeleton of I/O Program

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

//Add additional import statements as needed

public class ClassName
{
      //Declare appropriate variables
    public static void main(String[] args)
                                 throws FileNotFoundException
    {
             //Create and associate the stream objects
        Scanner inFile =
              new Scanner(new FileReader("prog.dat"));

        PrintWriter outFile = new PrintWriter("prog.out");

         //Code for data manipulation

            //Close file
        inFile.close();
        outFile.close();
    }
}
```

Programming Example: Movie Ticket Sale and Donation to Charity

- Input: movie name, adult ticket price, child ticket price, number of adult tickets sold, number of child tickets sold, percentage of gross amount to be donated to charity
- Output:



```
Movie Name: Duckey Goes to Mars
Number of Tickets Sold: 2650
Gross Amount: $9150.00
Percentage of the Gross Amount Donated: 10.00%
Amount Donated: $915.00
Net Sale: $8235.00
```

FIGURE 3-16 Output of theater sales program

Programming Example: Movie Ticket Sale and Donation to Charity (continued)

- Import appropriate packages
- Get inputs from user using `JOptionPane.showInputDialog`
- Perform appropriate calculations
- Display output using `JOptionPane.showMessageDialog`

Programming Example: Student Grade

- Input: file containing student’s first name, last name, five test scores
- Output: file containing student’s first name, last name, five test scores, average of five test scores

Programming Example: Student Grade (continued)

- Import appropriate packages
- Get input from file using the `class`es `Scanner` and `FileReader`
- Read and calculate the average of test scores
- Write to output file using the `class` `PrintWriter`
- Close files

Chapter Summary

- Primitive type variables store data into their memory space
- Reference variables store the address of the object containing the data
- An object is an instance of a class

Chapter Summary (continued)

- Operator `new` is used to instantiate an object
- Garbage collection is reclaiming memory not being used
- To use a predefined method you must know its name and the class and package it belongs to
- The dot (.) operator is used to access a certain method in a class

Chapter Summary (continued)

- Methods of the `class String` are used to manipulate input and output data
- Dialog boxes can be used to input data and output results
- Data can be read from and written to files
- Data can be formatted using the `String` method `format`