CHAPTER 7

■ ■ ■

Input/Output

QUESTIONS AND EXERCISES

1. What does an instance of the `File` class represent?

   Answer:

   An instance of the `File` class is an abstract representation of a pathname of a file or directory in a platform-independent manner.

2. Explain the effect of the following statement:

   ```
   File file = new File("test.txt");
   ```

   Will this statement create a file named `test.txt` in the current directory if the file does not already exist?

   Answer:

   A `File` object represents an abstract pathname, which may or may not point to a real file in the file-system. The statement creates a File object for the `test.txt` abstract pathname.

   No. This statement will not create a `test.txt` file if the file does not exist.

3. What is the difference in using the `delete()` and `deleteOnExit()` methods of the `File` class?

   Answer:

   The `delete()` method deletes the file immediately while the `deleteOnExit()` method deletes file when the JVM terminates.

4. What is the difference in using the `mkdir()` and `mkdirs()` methods of the `File` class?

   Answer:

   The `mkdir()` method creates a directory only if the parent directories specified in the pathname already exists. The `mkdirs()` method creates all missing directories specified in the pathname.

5. Complete the code for the following method named `isExistentDirectory()`. It accepts a pathname as a parameter. It returns `true` if the specified pathname represents an existing directory and returns `false` otherwise.

```
public static boolean isExistentDirectory(String pathname) {
    /* your code goes here */
}
```

   Solution:

```
public static boolean isExistentDirectory(String pathname) {
    File dir = new File(pathname);
    return dir.isDirectory();
}
```

6. What classes in the `java.io` package would you use if you need to read and write values of primitive data types such as `int` and `float`?

   Answer:

   `DataInputStream` and `DataOutputStream`

7. What classes in the `java.io` package would you use if you need to serialize and deserialize objects to a file?

   Answer:

   `ObjectInputStream` and `ObjectOutputStream`.

8. In the context of object serialization, what is the difference between implementing the `Serializable` and `Externalizable` interfaces in a class?

Answer:

The `Serializable` interface is a marker interface. Java takes care of the details of reading/writing a `Serializable` object from/to a stream. The `Externalizable` interface declares two methods: `readExternal()` and `writeExternal()`. The logic to read and write object's fields needs to be implemented in these two methods.

9. In the context of object serialization, what is the significance of declaring instance variables in a class as `transient`?

Answer:

A `transient` field of a `Serializable` object is not serialized.

10. Is there a way to serialize `transient` instance variables while serializing objects?

Answer:

Yes. There are two options:

- If the object is `Externalizable`, even `transient` instance variables can be serialized in the `writeExternal()` method.
- If the `transient` field is included in the `serialPersistentFields` field, it will be serialized.

11. How will you stop the objects of your class from being serialized?

Answer:

To stop objects of a class from being serialized, add `writeObject()` and `readObject()` methods in the class and throw an exception from both methods.

12. What is `serialVersionUID` in the context of object serialization?

Answer:

Declaring a `private`, `static`, `final` instance variable of `long` type, which is named `serialVersionUID` in a class, ensures that object of this class is deserialized properly, even if class definition is changed after serializing objects.

13. Suppose you have an existing text file named `test.txt` in your current directory? Write the code to append `"Hello"` to this file.

Answer:

```
try (FileWriter fw = new FileWriter("test.txt", true))  {
    fw.append("Hello");
} catch (FileNotFoundException e) {
    System.out.println("File not found: " + e.getMessage());
} catch (IOException e) {
    e.printStackTrace();
}
```

14. Which class in the `java.io` package would you if you need to read from and write to a file at the same time?

Answer:

`RandomAccessFile`

15. What is the difference between the `InputStream` and `Reader` classes while performing I/O?

Answer:

`InputStream` is byte-based whereas `Reader` is character-based.

16. Write a program using the `Console` and `Scanner` classes. The program prompts the user for an integer. When the user enters an integer, the program prints

whether the integer is odd or even and the program prompts the user for another integer. The user can enter `Q` or `q` to exit the program any time.

Solution:

```
// OddEvenChecker.java
package com.jdojo.io.exercises;

import java.util.Scanner;

public class OddEvenChecker {
    public static void main(String[] args) {
        String promptMsg = "Input an integer to test. Enter Q or q to quit: ";
        System.out.print(promptMsg);

        Scanner in = new Scanner(System.in);
        while (in.hasNext()) {
            String s = in.next();
            if (s.trim().equalsIgnoreCase("q")) {
                break;
            }

            int n = Integer.parseInt(s);
            String numType = n % 2 == 0 ? "even" : "odd";
            System.out.println(n + " is " + numType);

            System.out.print(promptMsg);
        }
    }
}
```

17. Write a program that will read the contents of a file and it will print the number of times all vowels (a, e, i, o and u) occurs in the file. Count should be case-insensitive. That is, both 'A' and 'a' are counted as 'a'. You need to prompt the user to specify the pathname to the file. Include error handling in your code such as for the case when the specified file does not exist.

Solution:

```
// VowelCounter.java
package com.jdojo.io.exercises;

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.Scanner;

public class VowelCounter {
    public static void main(String[] args) {
```

```
        System.out.print("Enter the file path: ");
        Scanner in = new Scanner(System.in);
        try (BufferedReader br = new BufferedReader(new FileReader(in.next()))) {
            int aCount = 0;
            int eCount = 0;
            int iCount = 0;
            int oCount = 0;
            int uCount = 0;

            String text;
            while ((text = br.readLine()) != null) {
                int len = text.length();
                for (int i = 0; i < len; i++) {
                    char c = text.charAt(i);
                    switch (c) {
                        case 'A':
                        case 'a':
                            aCount++;
                            break;
                        case 'E':
                        case 'e':
                            eCount++;
                            break;
                        case 'I':
                        case 'i':
                            iCount++;
                            break;
                        case 'O':
                        case 'o':
                            oCount++;
                            break;
                        case 'U':
                        case 'u':
                            uCount++;
                            break;
                        default:
                            break;
                    }
                }
            }

            // Print the result
            System.out.printf("Counts: a=%d, e=%d, i=%d, o=%d, u=%d%n",
                    aCount, eCount, iCount, oCount, uCount);

        } catch (FileNotFoundException e) {
            System.out.println("File not found: " + e.getMessage());
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```