Chapter 17 Input and Output

1. `FileInputStream` for input and `FileOutputStream` for output must be used.

2. Data in binary format is read or written by `InputStream` and `OutputStream`. You cannot use `read(byte b)` or `write(byte b)` for `InputStream` and `Outstream`, because they are abstract methods.

3. The value of a byte is returned as an `int` in the range 0 to 255. If no byte is available because the end of the stream has been reached, the value -1 is returned.

4. Characters are read or written by the `Reader` and `Writer` classes. `read()` or `write(char c)` can be used in those streams.

5. The byte stream is a stream of bytes, and the character stream is a stream of characters. Different classes are used to handle these streams.

6. Bytes are read or written using file streams. You can use `read()` or `write(byte b)` in file streams.

7. The data streams (`DataInputStream` and `DataOutputStream`) read and write Java primitive types in a machine-independent fashion, which enables you to write a data file on one machine and read it on another machine with a different operating system or file structure. They are often used for importing and exporting data.

8. They are very similar except that `PrintStream` is used to print things on the console and `DataOutputStream` is used with `FileWriter` to write things into a file.

9. `JFileChooser is modal. getSelectedFile() returns an instance of File. getSelectedDirectories() returns an instance of File[]. Use setCurrentDirectory(new File(".")) to set current directory as default directory for JFileChooser.`

10.
```
InputStream System.in
PrintStream System.out
PrintStream System.err
```

11. Any objects that are instance of Serializable can be stored using the object stream. You use the writeObject method to write an object to the object output stream and use readObject to read an object from the object input stream. The readObject method returns a value of the Object type. A static variable is not serialized. If you don't want a variable to be serialized, mark it transient.

12. Yes, because they share the same interface for reading and writing data in the same format.

13.
```
RandomAccessFile raf = new
RandomAccessFile("student.dat", "rw");

DataOutputStream outfile = new DataOutputStream(new
FileWriter("student.dat"));
```

To create a `RandomAccessFile` stream, you simply use the `RandomAccessFile` constructor. To create a `DataOutputStream`, you use `DataOutputStream` wrapped on `FileWriter`.

14. `It will compile fine, but raises a run time exception.`

15. When do you use `StreamTokenizer`?
Answer: You use `StreamTokenizer` when processing text files.

How do you read data using `StreamTokenizer`?
Answer: To create a `StreamTokenizer` object, you need to create it wrapped on `FileWriter`.

Where is the token stored when you are using the `nextToken()` method?
Answer: The `nextToken` is stored in the `in_sval` variable.

How do you find the data type of the token?
Answer: To find the data type of the token, use the `in_type` variable.

16. No.