Java IO

1. Fill in the blanks: Writer is ____that related stream classes_____.
A. a concrete class, extend
B. an abstract class, extend
C. an interface, extend
D. an interface, implement

2. Which of the following methods is defined in java.io.File?
A. createDirectory()
B. getLength()
C. listFile()
D. renameTo()

3. Which method in InputStream can be used in place of calling skip(1)?
A. jump()
B. mark()
C. read()
D. reset()

4. Which methods are classes that implement java.io.Serializable required to implement?
A. deserialize()
B. serial()
C. serialize()
D. None of the above

5. Fill in the blanks: Given a valid Console instance, reader () returns a __,
while writer () returns a_____ .
A. PrintReader, PrintWriter
B. PrintReader, Writer
C. Reader, Writer
D. StringReader, Writer

6. Assuming the file path referenced in the following class is accessible and able to be written, what is the output of the following program?

```
package alarm;
import java.io.*;
public class Smoke {
public void sendAlert(File fn) {
try(BufferedWriter w = new BufferedWriter(new FileOutputStream(fn))) {
w.write("ALERT!");
w.flush();
w.write('!');
System.out.print("1");
} catch (IOException e) {
System.out.print("2");
} finally {
System.out.print("3");
}
}
```

```
public static void main(String[] testSignal) {
new Smoke().sendAlert(new File("alarm.txt"));
}
}
```
A. 3
B. 13
C. 23
D. The code does not compile.

7. Which class is used to read information about a directory within the file system?
A. java.io.File
B. java.io.Directories
C. java.io.Directory
D. java.io.Path

8. Which of the following is a high-level stream class that can only be used to wrap a low-level stream?
A. FileOutputStream
B. FileReader
C. ObjectInputStream
D. PrintWriter

9. Assume the file prime6.txt exists and contains the first six prime numbers as bytes: 2, 3,5, 7, 11, 13.
What is the output of the following application?
```
package numbers;
import java.io.*;
public class PrimeReader {
public static void main(String[] real) throws Exception {
try (InputStream is = new FileInputStream("prime6.txt")) {
is.skip(1);
is.read();
is.skip(1);
is.read();
is.mark(4);
is.skip(1);
is.reset();
System.out.print(is.read());
}
}
}
```
A. 11
B. 13
C. The code does not compile.
D. The code compiles but throws an exception at runtime.

10. Fill in the blanks: For a given file, the absolute is the path from the _____to the file, while the relative path is the path from the_____ to the file.
A. current directory, current working directory
B. parent directory, temporary directory
C. root directory, current working directory

D. root directory, parent directory

11. Which statement best describes the following two methods?

```
public void writeSecret1() throws IOException {
final Writer w = new BufferedWriter(
new FileWriter("dont.open"));
w.write("Secret passcode");
w.close();
}
public void writeSecret2() throws IOException {
try(final Writer w = new BufferedWriter(
new FileWriter("dont.open"))) {
w.write("Secret passcode");
}
}
```

A. Both methods compile and are equivalent to each other.
B. Neither method compiles.
C. Only one of the methods compiles.
D. The methods compile, but one method may lead to a resource leak.

14. Let's say we want to write an instance of Cereal to disk, having a name value of CornLoops. What is the value of name after this object has been read using the ObjectInputStream's readObject() method?

```
package breakfast;
public class Cereal {
private String name = "CocoaCookies";
private transient int sugar;
public Cereal() {
super();
this.name = "CaptainPebbles";
}
{
name = "SugarPops";
}
public String getName() { return name; }
public void setName(String name) {
this.name = name;
}
public int getSugar() { return sugar; }
public void setSugar(int sugar) {
this.sugar = sugar;
}
}
```

A. CaptainPebbles
B. CornLoops
C. SugarPops
D. None of the above

15. Which statement best describes the difference between a Writer and an OutputStream class?

A. Only one of them can write text or character data.
B. Only one of them has built-in methods for writing character data.
C. Only one of them has a flush() method to force the data to be written out.
D. One uses a byte array to process character data more efficiently.

16. What is the output of the following application? It is safe to assume the directories referenced in the class do not exist prior to the execution of the program and that the file system is available and able to be written.

```
package job;
import java.io.*;
public class Resume {
public void resetWorkingDirectory() throws Exception {
File f1 = new File("/templates/proofs");
f1.mkdirs();
File f2 = new File("/templates");
f2.mkdir(); // k1
new File(f2,"draft.doc").createNewFile();
f1.delete();
f2.delete(); // k2
}
public static void main(String... leads) {
try {
new Resume().resetWorkingDirectory();
} catch (Exception e) {
new RuntimeException(e);
}
}
}
```

A. Line k1 does not compile or triggers an exception at runtime.
B. Line k2 does not compile or triggers an exception at runtime.
C. The code compiles and runs without printing an exception.
D. None of the above

18. How many compilation errors does the following class contain?

```
package hero;
import java.io.*;
public class Guitar {
public void readMusic(File f) {
try (BufferedReader r = new BufferedReader(FileReader(f))) {
final String music = null;
try {
while((music = r.readLine()) != null)
System.out.println(music);
} catch (IOException e) {}
} catch (FileNotFoundException e) {
throw new RuntimeException(e);
} finally {}
}}
```

A. None
B. One
C. Two

D. Three

19. What is the difference between the two Console methods, format() and printf()?
A. One of them takes an optional list of arguments; the other does not.
B. One of them takes String as input; the other takes an Object.
C. There is no difference between the two methods.
D. Trick question! printf() is not defined in Console.

20. Let's say you want to write a lot of text data to a file in an efficient manner. Which two java.io stream classes are best to use?
A. FileOutputStream and BufferedOutputStream
B. FileOutputWriter and FileBufferedWriter
C. FileWriter and BufferedWriter
D. ObjectOutputStream and BufferedWriter

Answers

1. B. Writer is an abstract class, making Option B the correct answer. Note that InputStream, OutputStream, and Reader are also abstract classes.

2. D. File uses mkdir() and mkdirs() to create a directory, not createDirectory(), making Option A incorrect. Note there is a createDirectory() method in the NIO.2 Files class. The getLength() method also does not exist, as the correct method is called length(). Next, there is a listFiles() method used to read the contents of a directory, but there is no listFile() method. That leaves us with renameTo(), which does exist and is used to rename file system paths.

3. C. The skip() method just reads a single byte and discards the value. The read() method can be used for a similar purpose, making Option C the correct answer. Option A is incorrect because there is no jump() method defined in InputStream. Options B and D are incorrect because they cannot be used to skip data, only to mark a location and return to it later, respectively.

4. D. Serializable is a marker or tagging interface, which means it does not contain any methods and is used to provide information about an object at runtime. Therefore, Option D is the correct answer because the interface does not define any abstract methods.

5. C. Given a valid instance of Console, reader() returns an instance of Reader, while writer() returns an instance of PrintWriter. Reader and PrintWriter was not an answer choice though, making Option C the next best choice since PrintWriter inherits Writer. Options A and B are incorrect because PrintReader is not defined in the java .io library. Option D is incorrect because the type of the instance returned by reader() is Reader, which does not inherit StringReader.

6. D. BufferedWriter is a wrapper class that requires an instance of Writer to operate on. In the Smoke class, a FileOutputStream is passed, which does not inherit Writer, causing the class not to compile, and making Option D the correct answer. If FileWriter was used instead of FileOutputStream, the code would compile without issue and print 13, making Option B the correct answer.

7. A. The File class is used to read both files and directories within a file system, making Option A the correct answer. The other three classes do not exist. Note there is an NIO.2 interface, java.nio.file.Path, used to read both file and path information.

8. C. FileOutputStream and FileReader are both low-level streams that operate directly on files, making Options A and B incorrect. ObjectInputStream is a high-level stream that can only wrap an existing InputStream. For this reason, Option C is the correct answer. PrintWriter can operate on other streams, but it can also operate on files. Since the question asks which class can only wrap low-level streams, Option D is incorrect.

9. D. The code compiles, so Option C is incorrect. The FileInputStream does not support marks, though, leading to an IOException at runtime when the reset() method is called. For this reason, Option D is the correct answer. Be suspicious of any code samples that call the mark() or reset() method without first calling markSupported().

10. C. The absolute path is the full path from the root directory to the file, while the relative path is the path from the current working directory to the file. For this reason, Option C is the correct answer.

11. D. The difference between the two methods is that writeSecret1() does not take any steps to ensure the close() method is called after the resource is allocated. On the other hand, writeSecret2() uses a try-with-resources block, which automatically tries to close the resource after it is used. Without a try-with-resources statement or an equivalent finally block, any exception thrown by the write() method would cause the resource not to be closed in the writeSecret1() method, possibly leading to a resource leak. For this reason, Option D is the correct answer. Option A is incorrect since they are not equivalent to each other. Finally, Options B and C are incorrect because both compile without issue.

12. A. The constructor for Console is private. Therefore, attempting to call new Console() outside the class results in a compilation error, making Option A the correct answer. The correct way to obtain a Console instance is to call System.console(). Even if the correct way of obtaining a Console had been used, and the Console was available at runtime, stuff is null in the printItinerary() method. Referencing stuff.activities results in a NullPointerException, which would make Option B the correct answer.

13. A. While you might not be familiar with FilterOutputStream, the diagram shows that the two classes must inherit from OutputStream. Options B and C can be eliminated as choices since PrintOutputStream and Stream are not the name of any java.io classes. Option D can also be eliminated because OutputStream is already in the diagram, and you cannot have a circular class dependency. That leaves us with the correct answer, Option A, with BufferedOutputStream and PrintStream both extending FilterOutputStream. Note that ByteArrayOutputStream and FileOutputStream referenced in Options C and D, respectively, do not extend FilterOutputStream, although knowing this fact was not required to solve the problem.

14. D. The Cereal class does not implement the Serializable interface; therefore, attempting to write the instance to disk, or calling readObject() using ObjectInputStream, will result in a NotSerializableException at runtime. For this reason, Option D is the correct answer. If the class did implement Serializable, then the value of name would be CornLoops, since none of the constructor, initializers, or setters methods are used on deserialization, making Option B the correct answer.

15. B. An OutputStream is used to write bytes, while a Writer is used to write character data. Both can write character data, the OutputStream just needs the data converted to bytes first. For this reason, Option A is incorrect. Option B is the correct answer, with Writer containing numerous methods for writing character or String data. Both interfaces contain a flush() method, making Option C incorrect. Finally, because both can be used with a byte array, Option D is incorrect.

16. C. First off, the code compiles without issue. The first method call to mkdirs() creates two directories, /templates and /templates/proofs. The next mkdir() call is unnecessary, since /templates/proofs already exists. That said, calling it on an existing directory is harmless and does not cause an exception to be thrown at runtime. Next, a file draft.doc is created in the /templates directory. The final two lines attempt to remove the newly created directories. The first call to delete() is successful because /templates/proofs is an empty directory. On the other hand, the second call to delete() fails to delete the directory /templates because it is non-empty, containing the file draft.doc. Neither of these calls trigger an exception at runtime, though, with delete() just returning a boolean value indicating whether the call was successful. Therefore, our program ends without throwing any exceptions, and Option C is the correct answer.

17. D. To answer the question, you need to identify three of the four ways to call the systemindependent file name separator. For example, the file name separator is often a forwardslash (/) in Linux-based systems and a backward-slash (\) in Windows-based systems. Option A is valid because it is the fully qualified name of the property. Option B is also valid because File.separator and File.separatorChar are equivalent. While accessing a static variable using an instance is discouraged, as shown in Option B, it is allowed. Option C is valid and a common way to read the character using the System class. Finally, Option D is the correct answer and one call that cannot be used to get the system-dependent name separator character. Note that System.getProperty("path.separator") is used to separate sets of paths, not names within a single path.

18. D. The first compilation error is that the FileReader constructor call is missing the new keyword. The second compilation error is that the music variable is marked final, but then modified in the while loop. The third compilation problem is that the readMusic() method fails to declare or handle an IOException. Even though the IOException thrown by readLine() is caught, the one thrown by the implicit call to close() via the try-withresources block is not caught. Due to these three compilation errors, Option D is the correct answer.

19. C. Both of the methods do exist, making Option D incorrect. Both methods take the same arguments and do the exact same thing, making Option C the correct answer. The printf() was added as a convenience method, since many other languages use printf() to accomplish the same task as format().

20. C. FileWriter and BufferedWriter can be used in conjunction to write large amounts of text data to a file in an efficient manner, making Option C the correct answer. While you can write text data using FileOutputStream and BufferedOutputStream, they are primarily used for binary data. Since there is a better choice available, Option A is incorrect. Option B is incorrect since FileOutputWriter and FileBufferedWriter are not classes that exist within the java.io API. Option D is incorrect since ObjectOutputStream is a

high-level binary stream. Also, while it can write String data, it writes it in a binary format,not a text format.