JDBC API

1. **Which of the following JDBC code snippets will correctly handle SQLExceptions in a try-with-resources statement?**

```
try (Connection conn = DriverManager.getConnection(url, user,
password);
     Statement stmt = conn.createStatement()) {
    // some SQL operations
} catch (SQLException e) {
    // handle exception
}
```

   A) Only the Connection is auto-closed.

   B) Both Connection and Statement are auto-closed.

   C) Only Statement is auto-closed.

   D) Neither Connection nor Statement is auto-closed.

2. **Which of the following code snippets correctly sets a parameter for a `PreparedStatement`?**

```
PreparedStatement pstmt = conn.prepareStatement("SELECT * FROM users
WHERE id = ?");
pstmt.setString(1, "123");
ResultSet rs = pstmt.executeQuery();
```

   A) `pstmt.setInt(1, 123);`

   B) `pstmt.setString(1, 123);`

   C) `pstmt.setInt(1, "123");`

   D) `pstmt.setObject(1, "123");`

3. **Which SQL statement is executed by the following JDBC code snippet?**

```
String sql = "INSERT INTO students (name, age) VALUES (?, ?)";
try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
    pstmt.setString(1, "John");
    pstmt.setInt(2, 20);
    pstmt.executeUpdate();
}
```

   A) A new student record is updated.

B) A new student record is inserted.

C) A student record is deleted.

D) A student record is selected.

4. **In which scenario would you use `ResultSet.TYPE_SCROLL_INSENSITIVE`?**

```
Statement stmt =
conn.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
ResultSet.CONCUR_READ_ONLY);
```

A) When you want a `ResultSet` that does not reflect changes made to the database.

B) When you want to update the `ResultSet` while scrolling.

C) When you need a `ResultSet` that reflects changes made to the database.

D) When you want to create a `ResultSet` that can only scroll backward.

.

5. **What will be the output of the following code snippet if the `employees` table contains no rows?**

```
String sql = "SELECT COUNT(*) FROM employees";
try (Statement stmt = conn.createStatement();
     ResultSet rs = stmt.executeQuery(sql)) {
    if (rs.next()) {
        System.out.println(rs.getInt(1));
    }
}
```

A) `0`

B) `null`

C) `1`

D) No output

JUnit Testing

6. **What will be the output of the following JUnit test code?**

```
@Test
public void testMethod() {
    int result = 5 / 0;
    assertEquals(5, result);
}
```

A) Test will pass successfully.

B) Test will fail due to an `ArithmeticException`.

C) Test will not compile.

D) Test will pass with a warning.

7. **Which annotation is used to define a test case that depends on other test cases?**

```
@Test
@DependsOnMethods("otherTestMethod")
public void testMethod() {
    // test code
}
```

A) `@DependsOnMethods`

B) `@Test`

C) `@Before`

D) `@After`

8. **What will the following code snippet do in JUnit 4?**

```
@Test(timeout = 1000)
public void testMethod() throws InterruptedException {
    Thread.sleep(2000);
}
```

A) The test will pass successfully.

B) The test will fail due to a timeout.

C) The test will be skipped.

D) The test will throw a compilation error.

9. **What is the purpose of `@BeforeClass` annotation in JUnit 4?**

```
@BeforeClass
public static void setup() {
    // setup code
}
```

A) To initialize resources before each test method.

B) To run setup code once before any of the test methods in the class are run.

C) To clean up resources after all tests have been run.

D) To run code after each test method.

10. **What will be the result of the following parameterized test?**

```
@RunWith(Parameterized.class)
public class MyTest {
    @Parameter
    public int input;

    @Test
    public void testMethod() {
        assertTrue(input > 0);
    }
}
```

A) The test will pass if all inputs are greater than 0.

B) The test will fail if any input is not greater than 0.

C) The test will compile but will not run.

D) The test will always fail.

Deployment and Application Enhancement

11. **Which Maven plugin is used to generate a JAR file with dependencies included?**

```
xml

<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-shade-plugin</artifactId>
</plugin>
```

A) `maven-jar-plugin`

B) `maven-assembly-plugin`

C) `maven-shade-plugin`

D) `maven-war-plugin`

12. **What does the `mvn clean install` command do?**

A) Compiles the code and installs the JAR in the local repository.

B) Deletes all previously compiled files and installs the JAR.

C) Cleans the target directory and compiles the code.

D) Runs tests and installs the JAR.

13. **Which Maven phase is responsible for creating the project's JAR file?**

A) `compile`

B) `package`

C) `verify`

D) `install`

14. **What is the effect of the following Maven build profile configuration?**

```
xml

<profiles>
    <profile>
        <id>dev</id>
        <properties>
            <env>development</env>
        </properties>
    </profile>
</profiles>
```

A) Sets a property `env` to `development` when the `dev` profile is activated.

B) Activates the `dev` profile during the build.

C) Disables the `dev` profile.

D) Creates a new Maven project named `dev`.

15. **What Maven command generates a project’s site documentation?**

```
mvn site
```

A) Generates site documentation

B) Compiles the source code

C) Creates a JAR file

D) Runs unit tests

Collections Overview

16. **What will be the output of the following code snippet using `TreeSet`?**

```
Set<Integer> set = new TreeSet<>(Arrays.asList(5, 2, 8, 1));
System.out.println(set);
```

A) `[1, 2, 5, 8]`

B) `[5, 2, 8, 1]`

C) `[8, 5, 2, 1]`

D) `null`

17. **What is the result of calling `Collections.reverse()` on a `List`?**

```
List<String> list = new ArrayList<>(Arrays.asList("A", "B", "C"));
Collections.reverse(list);
System.out.println(list);
```

A) `[C, B, A]`

B) `[A, B, C]`

C) `[C, A, B]`

D) `[B, C, A]`

18. **Which code snippet will correctly sort a `List` of custom objects by a specified property?**

```
List<MyObject> list = ...;
list.sort(Comparator.comparing(MyObject::getProperty));
```

A) `list.sort(Comparator.comparing(MyObject::getProperty));`

B) `list.sort(MyObject::getProperty);`

C) `Collections.sort(list, MyObject::getProperty);`

D) `list.sort(new MyObjectComparator());`

19. **What is the purpose of the `Stream.flatMap()` method?**

```
List<List<String>> lists = Arrays.asList(
    Arrays.asList("A", "B"),
    Arrays.asList("C", "D")
);
lists.stream().flatMap(Collection::stream).forEach(System.out::printl
n);
```

A) To flatten a stream of streams into a single stream

B) To sort the stream

C) To filter the stream

D) To map each element to a different type

20. **Which method is used to sort a `Stream` in ascending order?**

```
stream.sorted()
```

A) `sorted()`

B) `sort()`

C) `order()`

D) `arrange()`

Exceptions

21. **What is the output of the following code snippet?**

```
try {
    throw new ArithmeticException();
} catch (Exception e) {
    System.out.println("Caught Exception");
} finally {
    System.out.println("In finally block");
}
```

A) `Caught Exception`

B) `In finally block`

C) `Caught Exception` and `In finally block`

D) Only `In finally block`

22. **What happens when a `RuntimeException` is thrown and not caught?**

A) The program terminates.

B) The program continues execution.

C) The program catches it automatically.

D) The exception is logged.

23. **Which keyword is used to create a custom exception class in ?**

```
public class MyException extends Exception {
    public MyException(String message) {
        super(message);
    }
}
```

A) `throws`

B) `throw`

C) `throws`

D) `extends`

24. **What will be the output of the following code snippet if the exception is not caught?**

```
public static void main(String[] args) {
    try {
        throw new IOException();
    } catch (IOException e) {
        throw new RuntimeException(e);
    }
}
```

A) The program will compile and terminate with a `RuntimeException`.

B) The program will compile and terminate with an `IOException`.

C) The program will compile and continue execution.

D) The program will throw a compilation error.

25. **What does the `try-with-resources` statement do with the resources it manages?**

```
try (FileReader fr = new FileReader("file.txt")) {
    // use FileReader
} catch (IOException e) {
    // handle exception
}
```

A) It automatically closes resources at the end of the block.

B) It manually closes resources at the end of the block.

C) It only handles exceptions.

D) It does nothing with resources.

Miscellaneous

26. **Which code snippet will throw an `ArrayIndexOutOfBoundsException`?**

```
int[] arr = new int[5];
arr[5] = 10;
```

A) This code will not compile.

B) This code will compile and run without exceptions.

C) This code will throw an `ArrayIndexOutOfBoundsException`.

D) This code will throw a `NullPointerException`.

27. **What will be the output of the following code snippet?**

```
List<String> list = Arrays.asList("A", "B", "C");
list.add("D");
System.out.println(list);
```

A) `[A, B, C, D]`

B) `[A, B, C]`

C) `UnsupportedOperationException`

D) `null`

28. **What will be the output of the following code snippet if the `finally` block does not have a return statement?**

```
public static int test() {
    try {
        return 1;
    } finally {
        // no return statement
    }
}
```

A) `1`

B) `0`

C) `Compilation error`

D) `RuntimeException`

29. **Which of the following code snippets will not compile?**

```
try {
    int[] numbers = new int[5];
    numbers[10] = 7;
} catch (ArrayIndexOutOfBoundsException e) {
    System.out.println("Caught Exception");
} finally {
    System.out.println("In finally block");
}
```

A) This code will compile and run successfully.

B) This code will compile but will throw an exception at runtime.

C) This code will not compile due to the array index issue.

D) This code will compile but will not execute the `finally` block.

30. **Which code snippet will correctly create a new `HashMap` and put some values into it?**

```
Map<String, Integer> map = new HashMap<>();
map.put("One", 1);
map.put("Two", 2);
```

A) `Map<String, Integer> map = new HashMap<>();`

B) `Map<String, Integer> map = new HashMap<String, Integer>();`

C) `Map map = new HashMap();`

D) `HashMap<String, Integer> map = new Map<>();`

31. **What is the result of executing the following code snippet?**

```
List<String> list = Arrays.asList("A", "B", "C");
list.set(1, "X");
System.out.println(list);
```

A) `[A, X, C]`

B) `[A, B, X]`

C) `[X, B, C]`

D) `[A, B, C]`

32. **What will be the result of the following code snippet?**

```
String[] array = {"", "Python", "C++"};
List<String> list = Arrays.asList(array);
list.set(0, "Script");
System.out.println(Arrays.toString(array));
```

A) `[Script, Python, C++]`

B) `[, Python, C++]`

C) `[, Script, C++]`

D) `UnsupportedOperationException`

33. **Which code snippet demonstrates how to throw a custom exception?**

```
public class MyException extends Exception {
    public MyException(String message) {
        super(message);
    }
}

public void someMethod() throws MyException {
    throw new MyException("Custom Exception");
}
```

A) `throw new MyException("Custom Exception");`

B) `throws new MyException("Custom Exception");`

C) `throw MyException("Custom Exception");`

D) `throw new Exception("Custom Exception");`

34. **What is the outcome of the following code snippet?**

```
try {
    throw new Exception("Base Exception");
} catch (Exception e) {
    throw new RuntimeException("Wrapper Exception", e);
}
```

A) `Wrapper Exception` with the cause `Base Exception`.

B) `Base Exception` with no cause.

C) `Wrapper Exception` with no cause.

D) `Base Exception` with `Wrapper Exception` as its cause.

.

35. **Which of the following will cause a `NoSuchElementException`?**

```
List<String> list = new ArrayList<>();
list.iterator().next();
```

A) This code will compile and run successfully.

B) This code will throw a `NoSuchElementException`.

C) This code will throw a `NullPointerException`.

D) This code will throw an `IndexOutOfBoundsException`.

36. **What is the result of the following code snippet?**

```
try {
    int[] arr = new int[1];
    arr[1] = 10;
} catch (ArrayIndexOutOfBoundsException e) {
    System.out.println("Exception caught");
} finally {
    System.out.println("Finally block");
}
```

A) `Exception caught` and `Finally block`

B) `Finally block` only

C) `Exception caught` only

D) No output

37. **Which of the following will compile without errors?**

```
public class Example {
    public static void main(String[] args) {
        try {
            throw new IOException();
        } catch (IOException e) {
            throw e;
        }
    }
}
```

A) The code will compile successfully.

B) The code will not compile due to an uncaught checked exception.

C) The code will not compile due to a missing catch block.

D) The code will compile but throw an exception at runtime.

38. **What does the following code snippet demonstrate?**

```
public void method() {
    try {
        throw new IOException();
    } catch (IOException e) {
        System.out.println("IOException caught");
    } finally {
        System.out.println("Finally block");
    }
}
```

A) Demonstrates handling of `IOException`.

B) Demonstrates that `finally` block will not execute.

C) Demonstrates that the `catch` block will not execute.

D) Demonstrates that the `finally` block will execute only if there is no exception.

39. **What is the behavior of the following code snippet?**

```
public static void main(String[] args) {
    try {
        throw new Exception("Test Exception");
    } catch (Exception e) {
        System.out.println(e.getMessage());
    }
}
```

A) Prints `Test Exception`.

B) Prints `null`.

C) Prints the class name of the exception.

D) No output.

40. **What will be the output of the following code snippet?**

```
try {
    throw new RuntimeException();
} finally {
    throw new IllegalStateException();
}
```

A) `RuntimeException`

B) `IllegalStateException`

C) `RuntimeException` followed by `IllegalStateException`

D) Compilation error