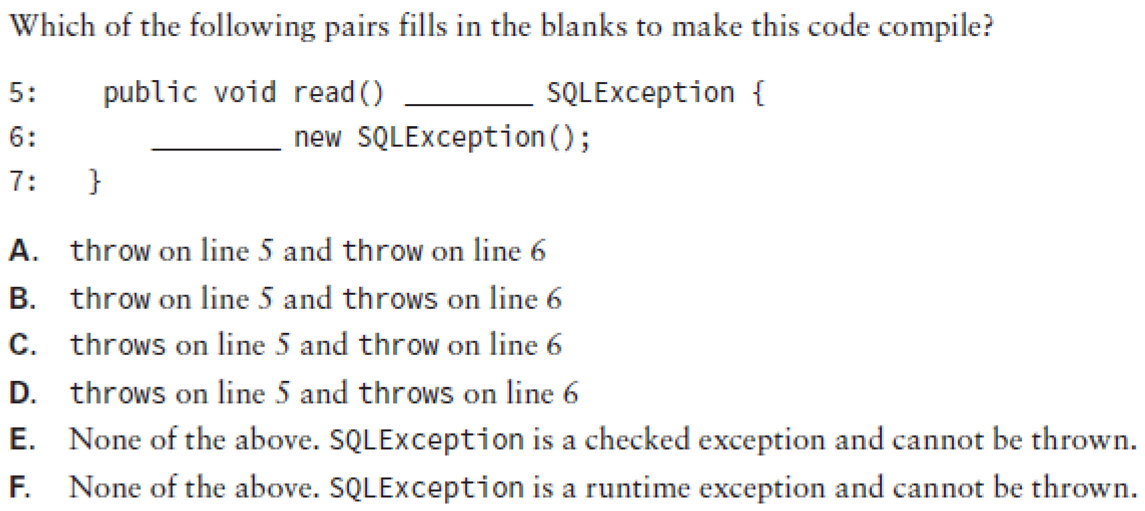
1



Which of the following pairs fills in the blanks to make this code compile?

```
5:    public void read() ________ SQLException {
6:          ________ new SQLException();
7:    }
```

A. `throw` on line 5 and `throw` on line 6

B. `throw` on line 5 and `throws` on line 6

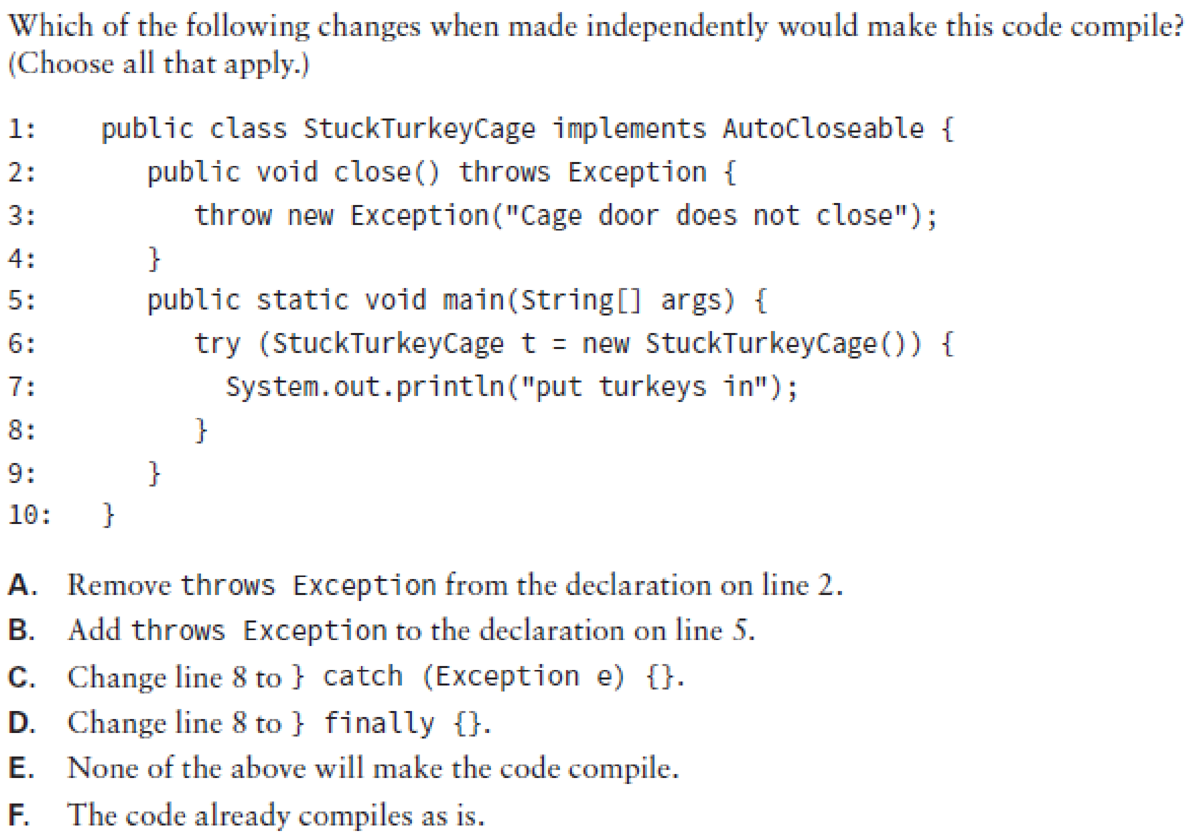
C. `throws` on line 5 and `throw` on line 6

D. `throws` on line 5 and `throws` on line 6

E. None of the above. `SQLException` is a checked exception and cannot be thrown.

F. None of the above. `SQLException` is a runtime exception and cannot be thrown.

2



Which of the following changes when made independently would make this code compile? (Choose all that apply.)

```
1:    public class StuckTurkeyCage implements AutoCloseable {
2:       public void close() throws Exception {
3:          throw new Exception("Cage door does not close");
4:       }
5:       public static void main(String[] args) {
6:          try (StuckTurkeyCage t = new StuckTurkeyCage()) {
7:            System.out.println("put turkeys in");
8:          }
9:       }
10:   }
```

A. Remove `throws Exception` from the declaration on line 2.

B. Add `throws Exception` to the declaration on line 5.

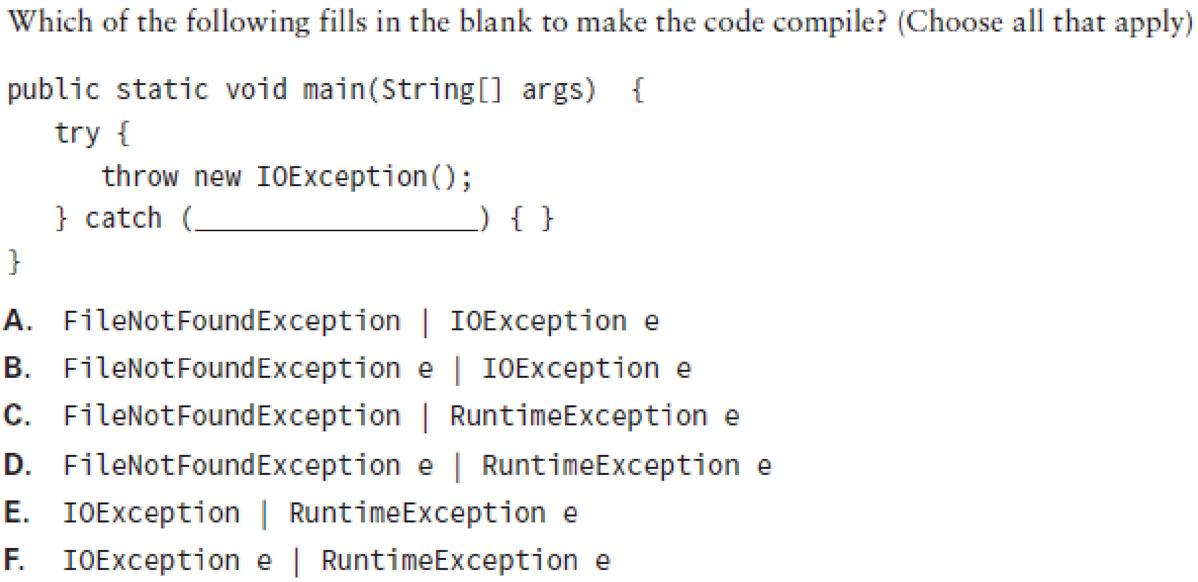
C. Change line 8 to `} catch (Exception e) {}`.

D. Change line 8 to `} finally {}`.

E. None of the above will make the code compile.

F. The code already compiles as is.

3



Which of the following fills in the blank to make the code compile? (Choose all that apply)

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

A. `FileNotFoundException | IOException e`

B. `FileNotFoundException e | IOException e`

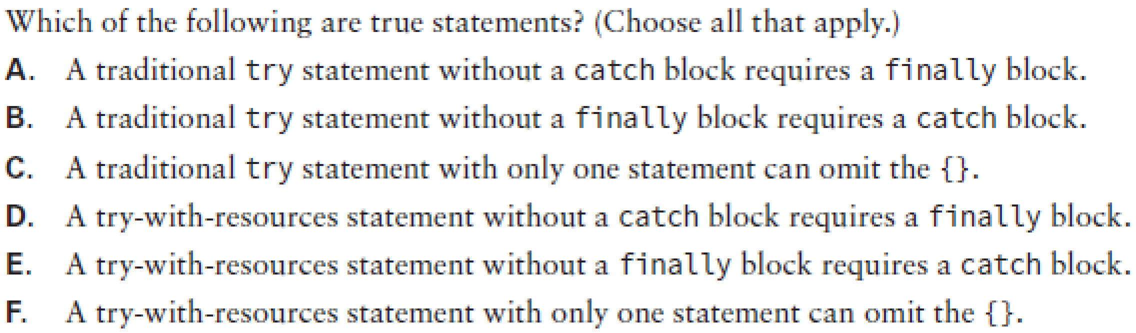
C. `FileNotFoundException | RuntimeException e`

D. `FileNotFoundException e | RuntimeException e`

E. `IOException | RuntimeException e`

F. `IOException e | RuntimeException e`

4



Which of the following are true statements? (Choose all that apply.)

A. A traditional `try` statement without a `catch` block requires a `finally` block.

B. A traditional `try` statement without a `finally` block requires a `catch` block.

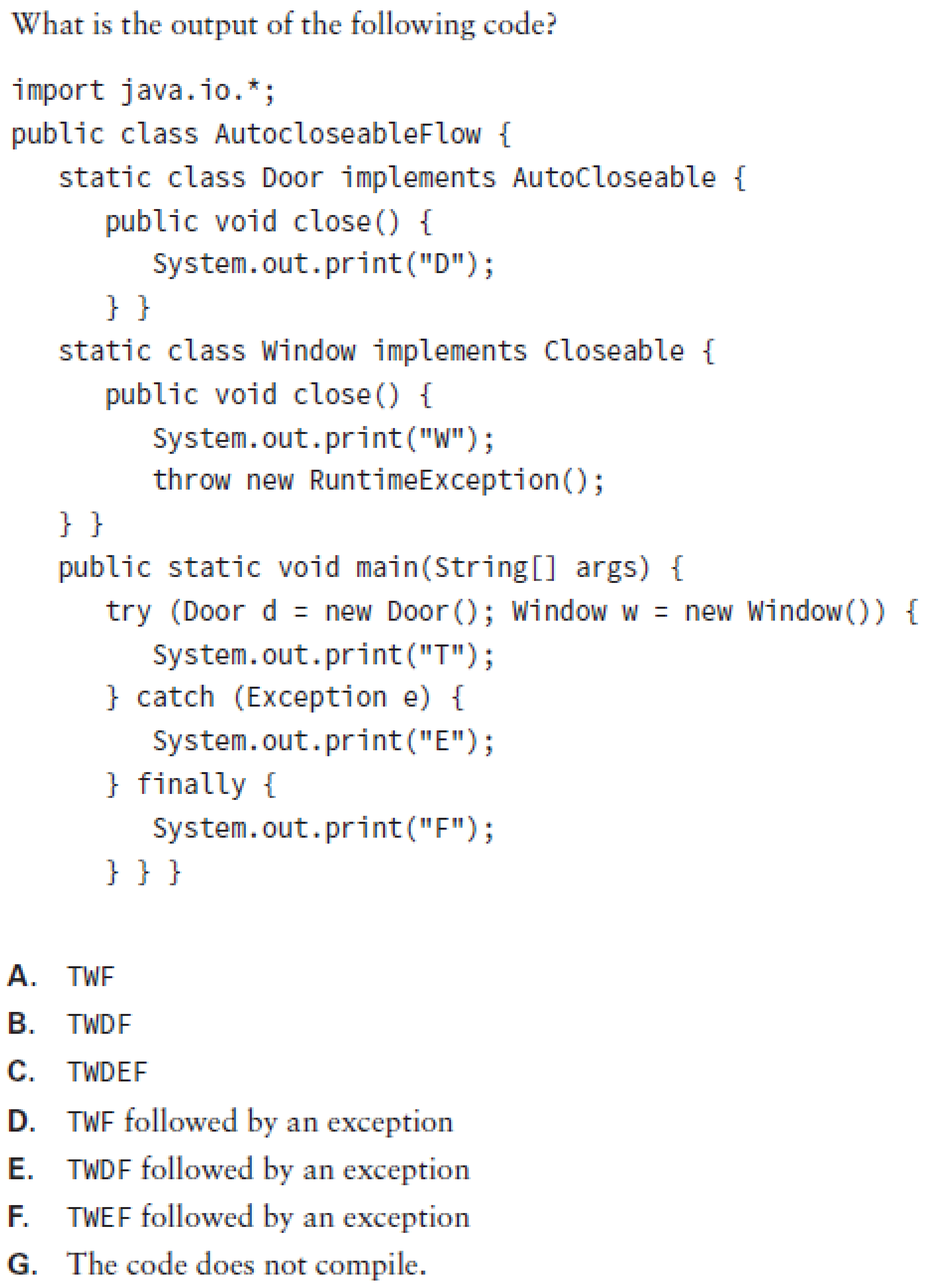
C. A traditional `try` statement with only one statement can omit the {}.

D. A try-with-resources statement without a `catch` block requires a `finally` block.

E. A try-with-resources statement without a `finally` block requires a `catch` block.

F. A try-with-resources statement with only one statement can omit the {}.

5



What is the output of the following code?

```
import java.io.*;
public class AutocloseableFlow {
   static class Door implements AutoCloseable {
      public void close() {
         System.out.print("D");
      } }
   static class Window implements Closeable {
      public void close() {
         System.out.print("W");
         throw new RuntimeException();
   } }
   public static void main(String[] args) {
      try (Door d = new Door(); Window w = new Window()) {
         System.out.print("T");
      } catch (Exception e) {
         System.out.print("E");
      } finally {
         System.out.print("F");
      } } }
```

A. TWF

B. TWDF

C. TWDEF

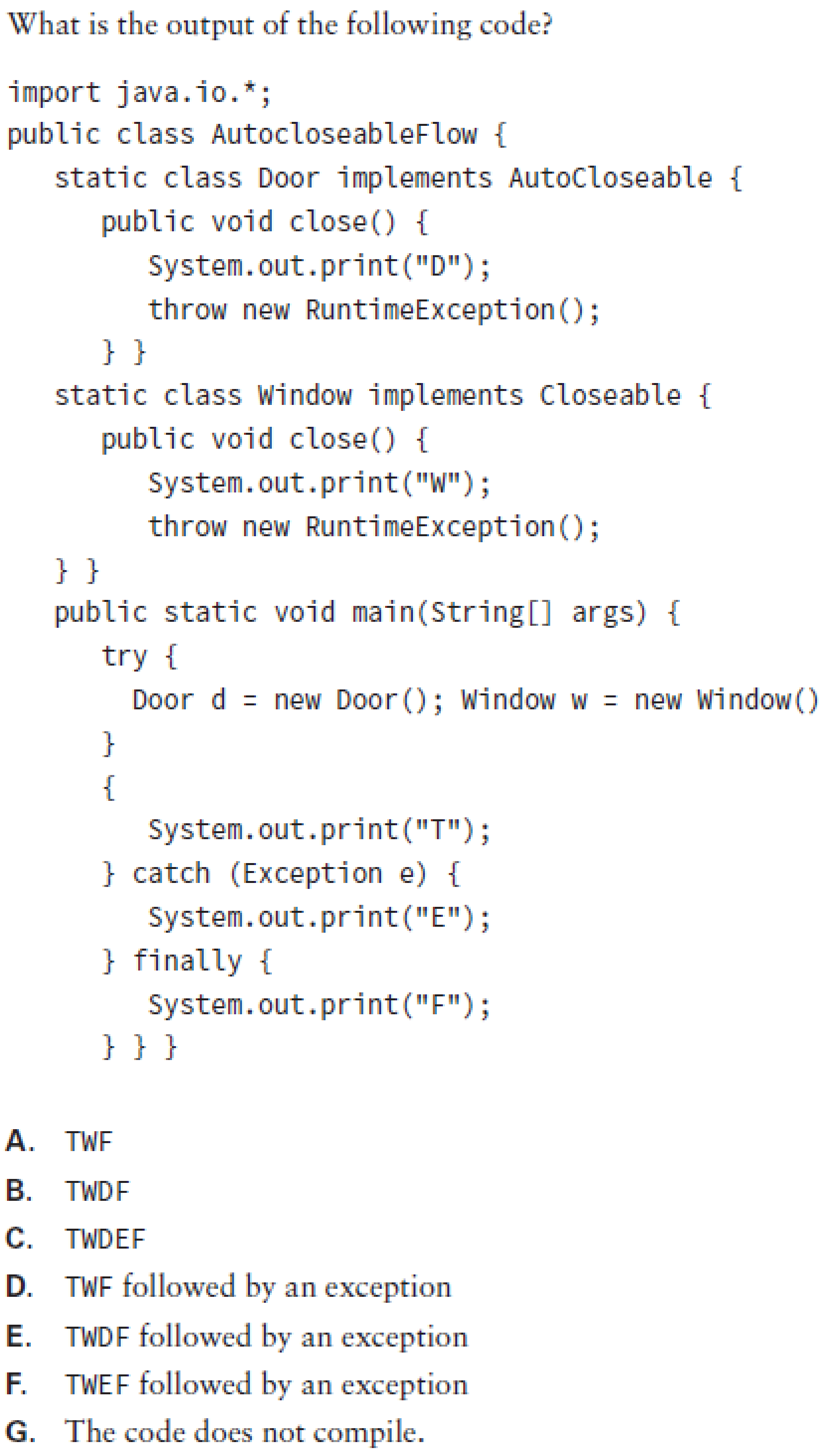
D. TWF followed by an exception

E. TWDF followed by an exception

F. TWEF followed by an exception

G. The code does not compile.

6



What is the output of the following code?

```
import java.io.*;
public class AutocloseableFlow {
   static class Door implements AutoCloseable {
      public void close() {
         System.out.print("D");
         throw new RuntimeException();
      } }
   static class Window implements Closeable {
      public void close() {
         System.out.print("W");
         throw new RuntimeException();
   } }
   public static void main(String[] args) {
      try {
        Door d = new Door(); Window w = new Window()
      }
      {
         System.out.print("T");
      } catch (Exception e) {
         System.out.print("E");
      } finally {
         System.out.print("F");
      } } }
```

A. TWF

B. TWDF

C. TWDEF

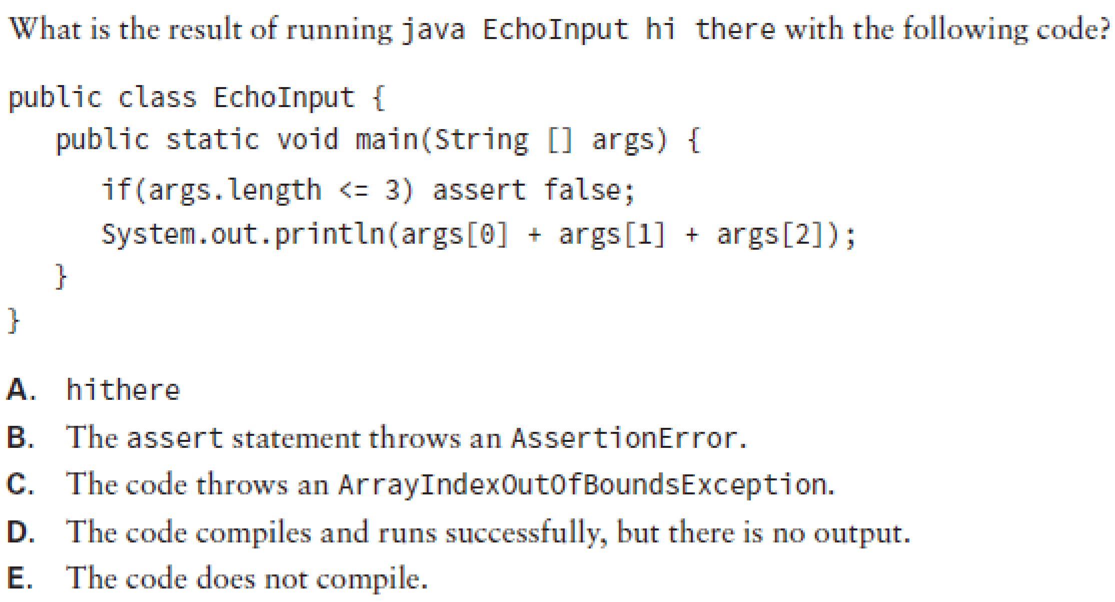
D. TWF followed by an exception

E. TWDF followed by an exception

F. TWEF followed by an exception

G. The code does not compile.

7



What is the result of running `java EchoInput hi there` with the following code?

```
public class EchoInput {
   public static void main(String [] args) {
      if(args.length <= 3) assert false;
      System.out.println(args[0] + args[1] + args[2]);
   }
}
```

A. `hithere`

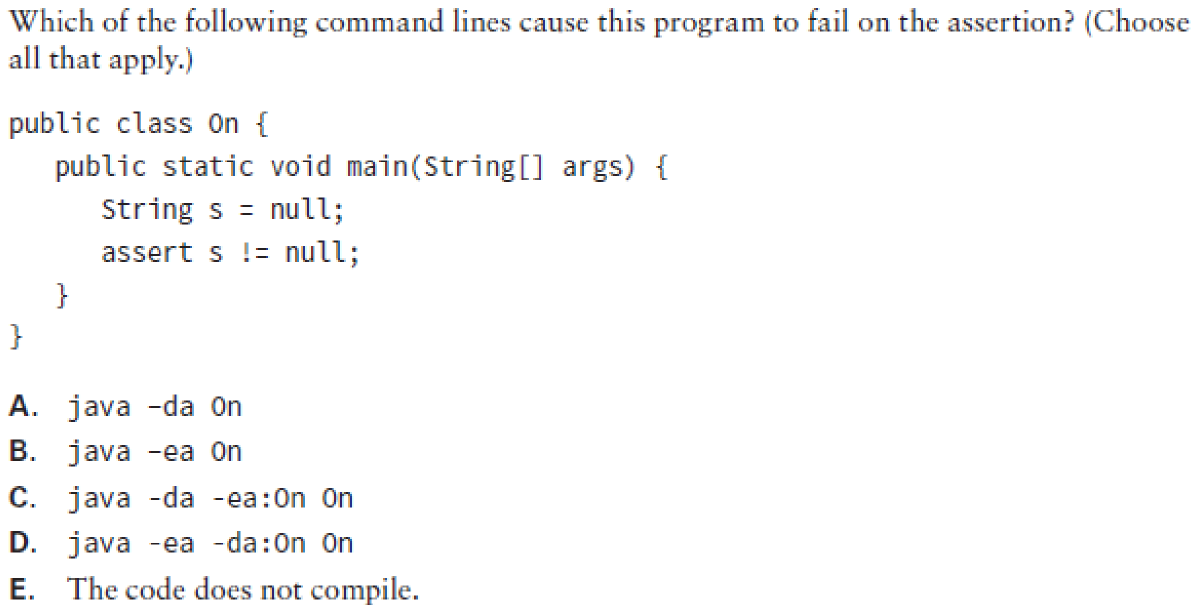
B. The `assert` statement throws an `AssertionError`.

C. The code throws an `ArrayIndexOutOfBoundsException`.

D. The code compiles and runs successfully, but there is no output.

E. The code does not compile.

8



Which of the following command lines cause this program to fail on the assertion? (Choose all that apply.)

```
public class On {
   public static void main(String[] args) {
      String s = null;
      assert s != null;
   }
}
```

A. `java -da On`

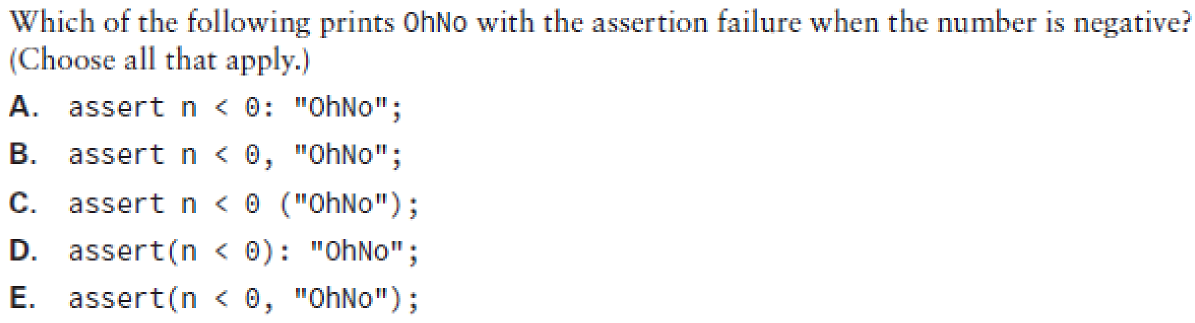
B. `java -ea On`

C. `java -da -ea:On On`

D. `java -ea -da:On On`

E. The code does not compile.

9



Which of the following prints `OhNo` with the assertion failure when the number is negative? (Choose all that apply.)

A. `assert n < 0: "OhNo";`

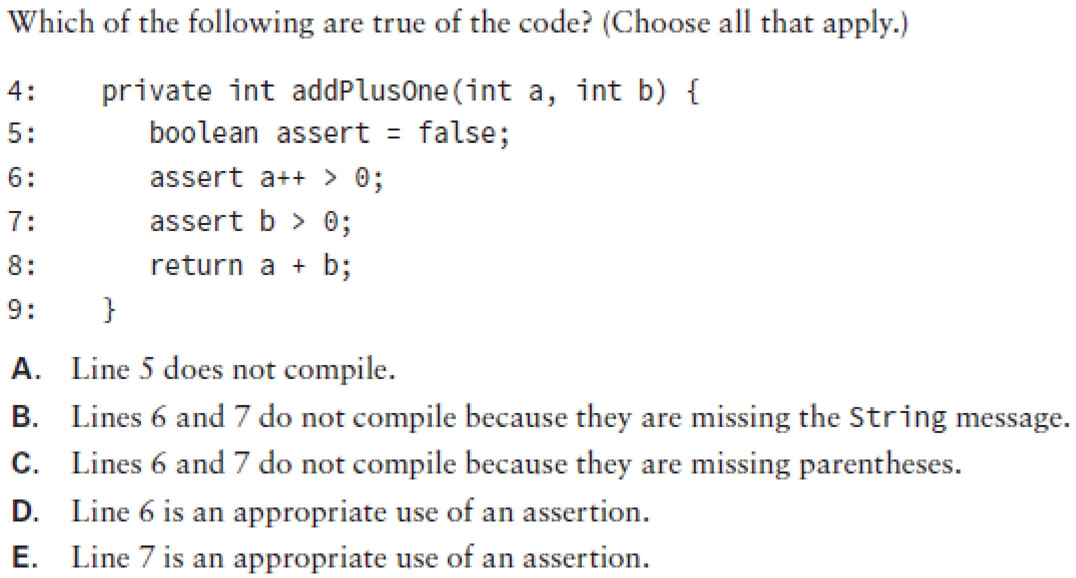
B. `assert n < 0, "OhNo";`

C. `assert n < 0 ("OhNo");`

D. `assert(n < 0): "OhNo";`

E. `assert(n < 0, "OhNo");`

10



Which of the following are true of the code? (Choose all that apply.)

```
4:    private int addPlusOne(int a, int b) {
5:       boolean assert = false;
6:       assert a++ > 0;
7:       assert b > 0;
8:       return a + b;
9:    }
```

A. Line 5 does not compile.

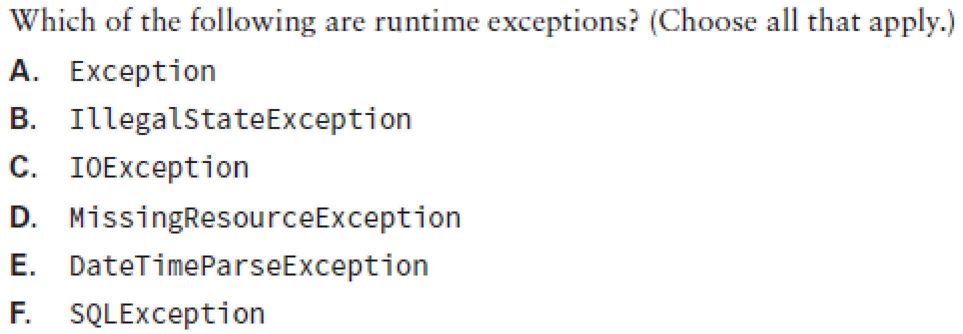
B. Lines 6 and 7 do not compile because they are missing the `String` message.

C. Lines 6 and 7 do not compile because they are missing parentheses.

D. Line 6 is an appropriate use of an assertion.

E. Line 7 is an appropriate use of an assertion.

11



Which of the following are runtime exceptions? (Choose all that apply.)

A. `Exception`

B. `IllegalStateException`

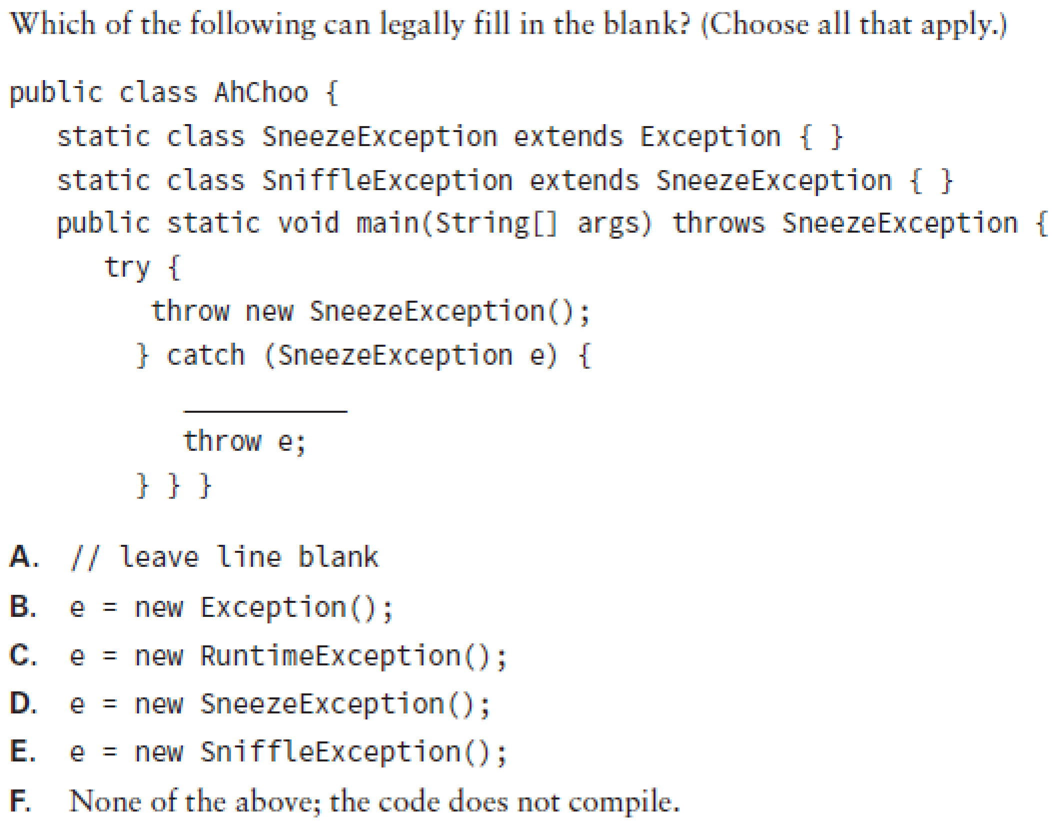
C. `IOException`

D. `MissingResourceException`

E. `DateTimeParseException`

F. `SQLException`

12



Which of the following can legally fill in the blank? (Choose all that apply.)

```
public class AhChoo {
   static class SneezeException extends Exception { }
   static class SniffleException extends SneezeException { }
   public static void main(String[] args) throws SneezeException {
      try {
         throw new SneezeException();
        } catch (SneezeException e) {
           __________
           throw e;
        } } }
```

A. `// leave line blank`

B. `e = new Exception();`

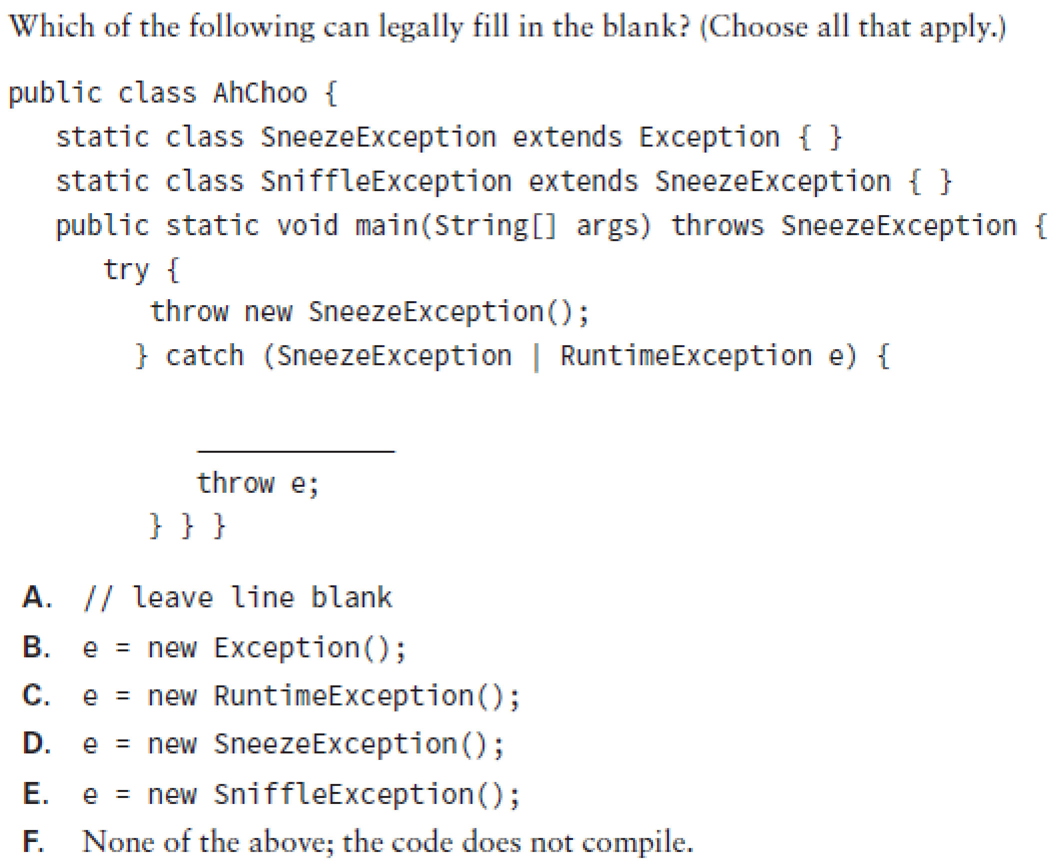
C. `e = new RuntimeException();`

D. `e = new SneezeException();`

E. `e = new SniffleException();`

F. None of the above; the code does not compile.

13



Which of the following can legally fill in the blank? (Choose all that apply.)

```
public class AhChoo {
   static class SneezeException extends Exception { }
   static class SniffleException extends SneezeException { }
   public static void main(String[] args) throws SneezeException {
      try {
         throw new SneezeException();
        } catch (SneezeException | RuntimeException e) {

           ____________
           throw e;
        } } }
```

A. `// leave line blank`

B. `e = new Exception();`

C. `e = new RuntimeException();`

D. `e = new SneezeException();`

E. `e = new SniffleException();`

F. None of the above; the code does not compile.

14

Which of the following can legally fill in the blank? (Choose all that apply.)

```
public class AhChoo {
   static class SneezeException extends Exception { }
   static class SniffleException extends SneezeException { }
   public static void main(String[] args) throws SneezeException {
      try {
         throw new SneezeException();
        } catch (SneezeException | SniffleException e) {
           _____________
           throw e;
        } } }
```

A. `// leave line blank`

B. `e = new Exception();`

C. `e = new RuntimeException();`

D. `e = new SneezeException();`

E. `e = new SniffleException();`

F. None of the above; the code does not compile.

15

Which of the following are checked exceptions? (Choose all that apply.)

```
class One extends RuntimeException{}
class Two extends Exception{}
class Three extends Error{}
class Four extends One{}
class Five extends Two{}
class Six extends Three{}
```

A. `One`

B. `Two`

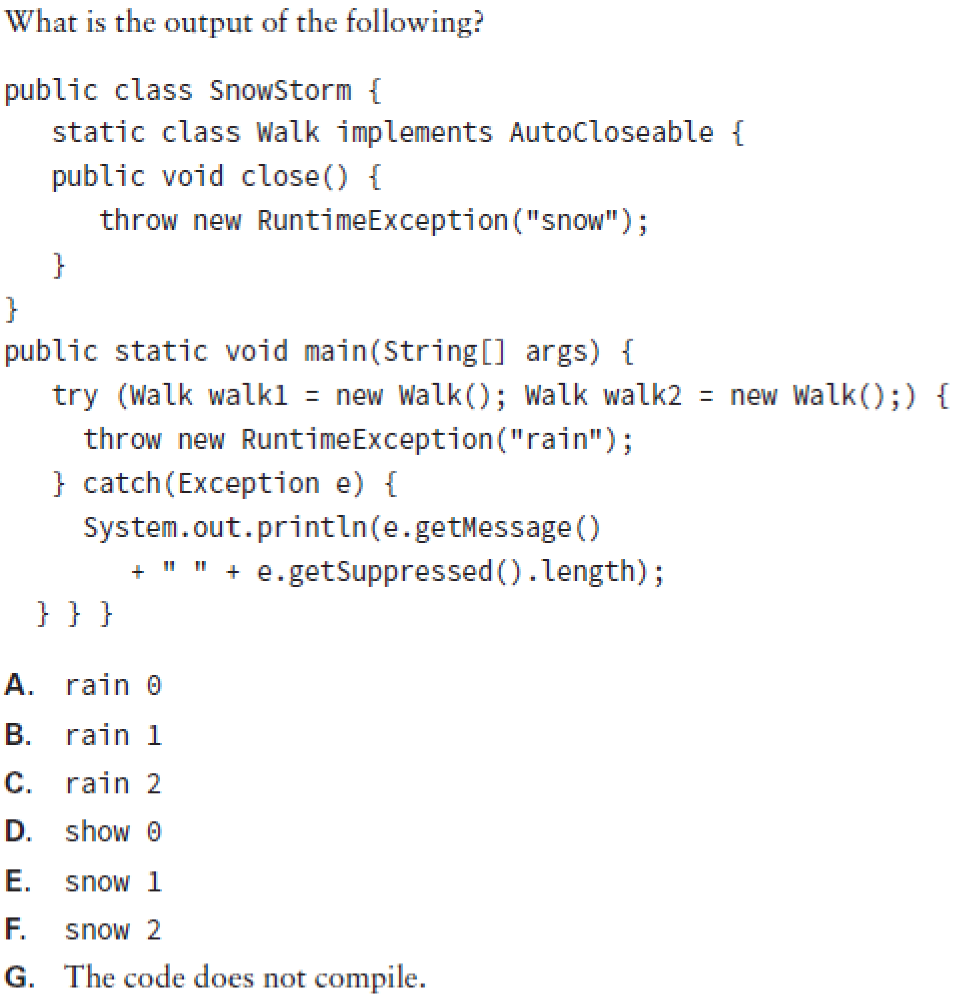
C. `Three`

D. `Four`

E. `Five`

F. `Six`

16



What is the output of the following?

```
public class SnowStorm {
   static class Walk implements AutoCloseable {
   public void close() {
      throw new RuntimeException("snow");
   }
}
public static void main(String[] args) {
   try (Walk walk1 = new Walk(); Walk walk2 = new Walk();) {
     throw new RuntimeException("rain");
   } catch(Exception e) {
     System.out.println(e.getMessage()
        + " " + e.getSuppressed().length);
  } } }
```

A. `rain 0`

B. `rain 1`

C. `rain 2`

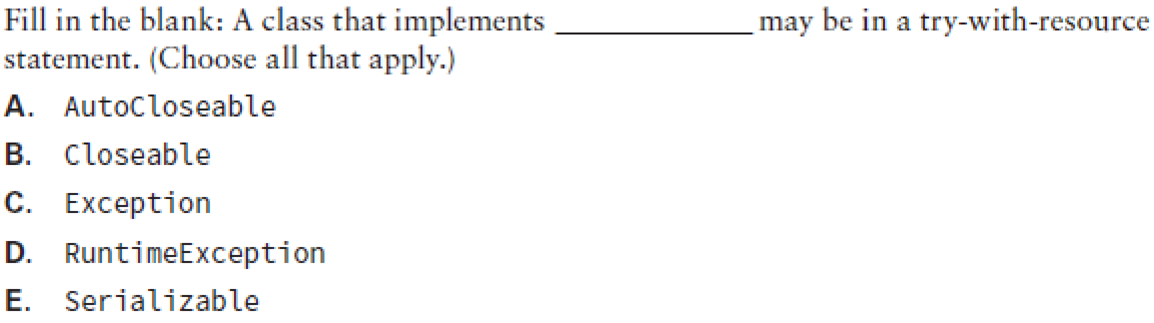
D. `show 0`

E. `snow 1`

F. `snow 2`

G. The code does not compile.

17



Fill in the blank: A class that implements _____________ may be in a try-with-resource statement. (Choose all that apply.)

A. `AutoCloseable`

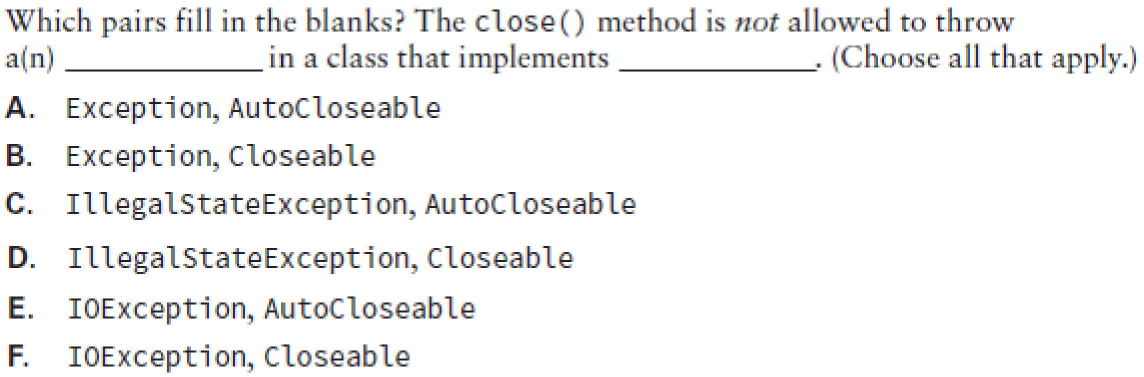
B. `Closeable`

C. `Exception`

D. `RuntimeException`

E. `Serializable`

18



Which pairs fill in the blanks? The `close()` method is *not* allowed to throw a(n) ____________ in a class that implements ____________. (Choose all that apply.)

A. `Exception`, `AutoCloseable`

B. `Exception`, `Closeable`

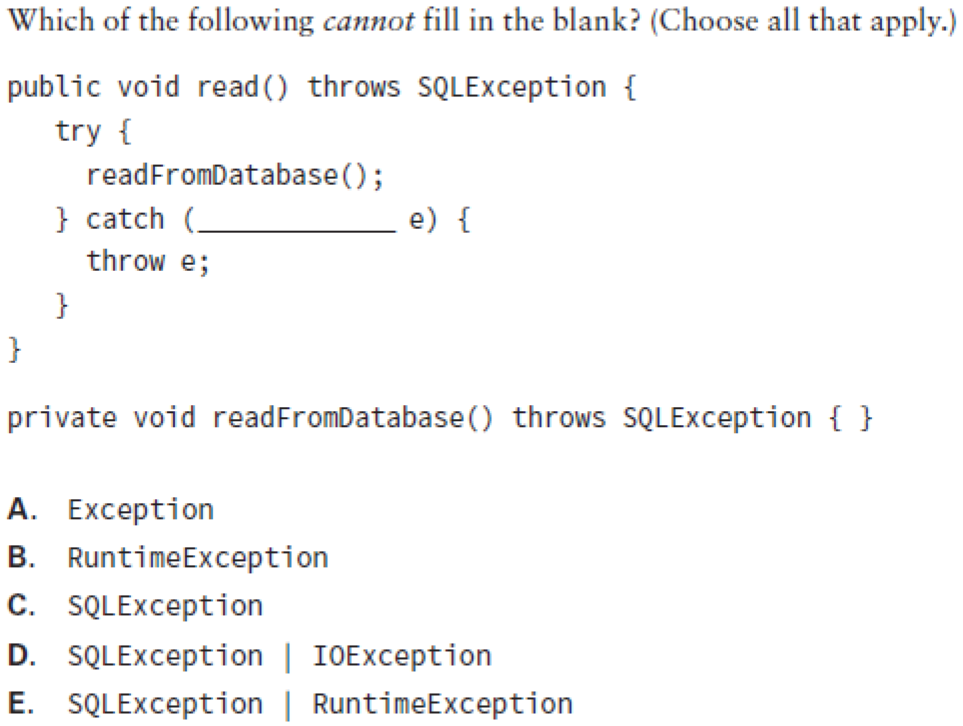
C. `IllegalStateException`, `AutoCloseable`

D. `IllegalStateException`, `Closeable`

E. `IOException`, `AutoCloseable`

F. `IOException`, `Closeable`

19



Which of the following *cannot* fill in the blank? (Choose all that apply.)

```
public void read() throws SQLException {
   try {
     readFromDatabase();
   } catch (____________ e) {
     throw e;
   }
}

private void readFromDatabase() throws SQLException { }
```

A. `Exception`

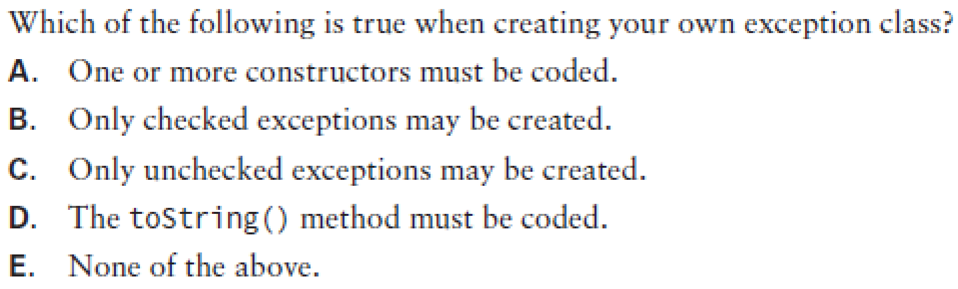
B. `RuntimeException`

C. `SQLException`

D. `SQLException | IOException`

E. `SQLException | RuntimeException`

20



Which of the following is true when creating your own exception class?

A. One or more constructors must be coded.

B. Only checked exceptions may be created.

C. Only unchecked exceptions may be created.

D. The `toString()` method must be coded.

E. None of the above.