Exceptions and Assertions - 0

Given code of Test.java file:

```
public class Test {
    public static void convert(String s)
            throws IllegalArgumentException, RuntimeException,
        Exception {
        if(s.length() == 0) {
            throw new RuntimeException("Length should be greater
        than 0.");
        }
    }
    public static void main(String [] args) {
        try {
            convert("");
        }
        catch(IllegalArgumentException | RuntimeException |
        Exception e) { //Line 14
            System.out.println(e.getMessage()); //Line 15
        } //Line 16
        catch(Exception e) {
            e.printStackTrace();
        }
    }
}
```

Line 14 is giving compilation error. Which of the following changes enables to code to print 'Length should be greater than 0.'?

A - Replace Line 14 with catch(IllegalArgumentException | RuntimeException e) {

B - Replace Line 14 with catch(IllegalArgumentException | Exception e) {

C - Comment out Line 14, Line 15 and Line 16

D - Replace Line 14 with catch(RuntimeException e) {

E - Replace Line 14 with catch(RuntimeException | Exception e) {

Exceptions and Assertions - 1

Consider the following interface declaration:

```
public interface I1 {
    void m1() throws java.io.IOException;
}
```

Which of the following incorrectly implements interface I1?

A -

```
public class C3 implements I1 {
  public void m1() throws java.io. IOException{}
}
```

B -

```
public class C1 implements I1 {
  public void m1() {}
}
```

C -

```
public class C2 implements I1 {
  public void m1() throws java.io.FileNotFoundException{}
}
```

D -

```
public class C4 implements I1 {
  public void m1() throws Exception{}
}
```

Exceptions and Assertions - 2

Given code of Test.java file:

```
public class Test {
    public static void main(String[] args) {
        try { //outer
            try { //inner
                System.out.println(1/0);
            } catch(ArithmeticException e) {
                System.out.println("Inner");
            } finally {
                System.out.println("Finally 1");
            }
        } catch(ArithmeticException e) {
            System.out.println("Outer");
        } finally {
            System.out.println("Finally 2");
        }
    }
}
```

What will be the result of compiling and executing Test class?

A -

```
Inner
Finally 1
```

B -

```
Inner
Finally 1
Finally 2
```

C -

```
Outer
Finally
```

D -

```
Inner
Finally 2
```

Exceptions and Assertions - 3

Given code of Test.java file:

```
class Resource1 {
    public void close() {
        System.out.println("Resource1");
    }
}

class Resource2 {
    public void close() {
        System.out.println("Resource2");
    }
}

public class Test {
    public static void main(String[] args) {
        try(Resource1 r1 = new Resource1(); Resource2 r2 = new
         Resource2()) {
            System.out.println("Test");
        }
    }
}
```

What will be the result of compiling and executing Test class?

A -

```
Test
Resource1
Resource2
```

B - Compilation Error

C -

```
Test
Resource2
Resource1
```

Exceptions and Assertions - 4

Given code of Test.java file:

```
import java.util.Scanner;

public class Test {
    public static void main(String[] args) {
        System.out.print("Enter some text: ");
        try(Scanner scan = new Scanner(System.in)) {
            String s = scan.nextLine();
            System.out.println(s);
            scan.close();
            scan.close();
        }
    }
}
```

What will be the result of compiling and executing Test class? User input is: HELLO

A - On execution program terminates successfully after printing ‘HELLO’ on to the console

B - Runtime Exception

C - Compilation error

Exceptions and Assertions - 5

Given code of Test.java file:

```
package com.training.ocp;

public class Test {
    private static void checkStatus() {
        assert 1 == 2 : 2 == 2;
    }

    public static void main(String[] args) {
        try {
            checkStatus();
        } catch (AssertionError ae) {
            System.out.println(ae.getCause());
        }
    }
}
```

What will be the result of executing Test class with below command? java -ea com.training.ocp.Test

A - Compilation error

B - null

C - true

D - false

Exceptions and Assertions - 6

Given code of Test.java file:

```
import java.sql.SQLException;

public class Test {
    private static void m() throws SQLException {
        try {
            throw new SQLException();
        } catch (Exception e) {
            throw e;
        }
    }

    public static void main(String[] args) {
        try {
            m();
        } catch(SQLException e) {
            System.out.println("Caught Successfully.");
        }
    }
}
```

What will be the result of compiling and executing Test class?

A - Program ends abruptly.

B - Method main(String []) causes compilation error.

C - Method m() causes compilation error.

D - Caught Successfully.

Exceptions and Assertions - 7

Given code of Test.java file:

```
class MyResource implements AutoCloseable {
    @Override
    public void close() {
        System.out.println("Closing");
    }
}

public class Test {
    public static void main(String[] args) {
        try(AutoCloseable resource = new MyResource()) {

        }
    }
}
```

What will be the result of compiling and executing Test class?

A - Closing

B - Compilation error in MyResource class

C - Compilation error in Test class

Exceptions and Assertions - 8

Given code of Test.java file:

```
class MyException extends RuntimeException {}

class YourException extends RuntimeException {}

public class Test {
    public static void main(String[] args) {
        try {
            throw new YourException();
        } catch(MyException e1 | YourException e2){
            System.out.println("Caught");
        }
    }
}
```

What will be the result of compiling and executing Test class?

A - Compilation error

B - Caught

C - Runtime Exception

Exceptions and Assertions - 9

Given code of Test.java file:

```
import java.io.IOException;
import java.sql.SQLException;

class MyResource implements AutoCloseable {
    @Override
    public void close() throws IOException{
        throw new IOException("IOException");
    }

    public void execute() throws SQLException {
        throw new SQLException("SQLException");
    }
}

public class Test {
    public static void main(String[] args) {
        try(MyResource resource = new MyResource()) {
            resource.execute();
        } catch(Exception e) {
            System.out.println(e.getMessage());
        }
    }
}
```

What will be the result of compiling and executing Test class?

A - Compilation error

B - SQLException

C - IOException

Exceptions and Assertions - 10

Given Code:

```
import java.io.*;

class ReadTheFile {
    static void print() { //Line 4
        throw new IOException(); //Line 5
    }
}

public class Test {
    public static void main(String[] args) { //Line 10
        ReadTheFile.print(); //Line 11
        //Line 12
    }
}
```

Which 2 changes are necessary so that code compiles successfully?

A - Replace Line 10 with public static void main(String[] args) throws IOException {

B - Surround Line 11 with below try-catch block:

```
try {
  ReadTheFile.print();
} catch(Exception e) { (Correc|
  e.printStackTrace();
}
```

C - Surround Line 11 with below try-catch block:

```
try {
  ReadTheFile.print();
} catch(IOException e) {
  e.printStackTrace();
}
```

D - Surround Line 11 with below try-catch block:

```
try {
  ReadTheFile.print();
} catch(IOException | Exception e) {
  e.printStackTrace();
}
```

E - Replace Line 4 with static void print() throws Throwable {

F - Replace Line 4 with static void print() throws Exception {

Exceptions and Assertions - 11

Given code of Test.java file:

```
import java.io.PrintWriter;

public class Test {
    public static void main(String[] args) {
        try(PrintWriter writer = new PrintWriter(System.out)) {
            writer.println("Hello");
        } catch(Exception ex) {
            writer.close();
        }
    }
}
```

What will be the result of compiling and executing Test class?

A - Program ends abruptly

B - Hello

C - Compilation error

Exceptions and Assertions - 12

Given code of Test.java file:

```
package com.training.ocp;

public class Test {
    enum STATUS {
        PASS, FAIL;
    }

    private static boolean checkStatus(STATUS status) {
        switch(status) {
            case PASS:
                return true;
            case FAIL:
                return false;
            default: {
                assert false : "<<<DANGER ZONE>>>";
                return false;
            }
        }
    }

    public static void main(String[] args) {
        checkStatus(null);
    }
}
```

What will be the result of executing Test class with below command? java -ea com.training.ocp.Test

A - NullPointerException is thrown and program ends abruptly

B - AssertionError is thrown and program ends abruptly

C - No output and program terminates successfully

Exceptions and Assertions - 13

Given code of Test.java file:

```
class Resource1 implements AutoCloseable {
    public void m1() throws Exception {
        System.out.print("A");
        throw new Exception("B");
    }

    public void close() {
        System.out.print("C");
    }
}

class Resource2 implements AutoCloseable {
    public void m2() {
        System.out.print("D");
    }

    public void close() throws Exception {
        System.out.print("E");
    }
}

public class Test {
    public static void main(String[] args) {
        try (Resource1 r1 = new Resource1();
             Resource2 r2 = new Resource2()) {
            r1.m1();
            r2.m2();
        } catch (Exception e) {
            System.out.print(e.getMessage());
        }
    }
}
```

What will be the result of compiling and executing Test class?

A - ACEB

B - ABEC

C - AECB

D - Compilation error

E - ABCE

Exceptions and Assertions - 14

Exceptions and Assertions - 15

Given code of Test.java file:

```
class Resource1 implements AutoCloseable {
    @Override
    public void close() {
        System.out.println("Resource1");
    }
}

class Resource2 implements AutoCloseable {
    @Override
    public void close() {
        System.out.println("Resource2");
    }
}

public class Test {
    public static void main(String[] args) {
        try(Resource1 r1 = new Resource1(); Resource2 r2 = new
        Resource2()) {
            System.out.println("Test");
        }
    }
}
```

What will be the result of compiling and executing Test class?

A -

```
Test
Resource2
Resource1
```

B -

```
Test
Resource1
Resource2
```

C - Compilation Error

Exceptions and Assertions - 16

Given code of Test.java file:

```
import java.sql.SQLException;

public class Test {
    private static void m() throws SQLException {
        try {
            throw new SQLException();
        } catch (Exception e) {
            e = new SQLException();
            throw e;
        }
    }

    public static void main(String[] args) {
        try {
            m();
        } catch(SQLException e) {
            System.out.println("Caught Successfully.");
        }
    }
}
```

What will be the result of compiling and executing Test class?

A - Caught Successfully.

B - Program ends abruptly.

C - Method m() causes compilation error.

D - Method main(String []) causes compilation error.

Exceptions and Assertions - 17

Given code of Test.java file:

```
import java.io.FileNotFoundException;
import java.io.FileReader;

public class Test {
    public static void main(String[] args) {
        try(FileReader fr = new FileReader("C:/temp.txt")) {

        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }
    }
}
```

Does above code compile successfully?

A - YES

B - NO

Exceptions and Assertions - 18

Given code of Test.java file:

```
import java.util.Scanner;

public class Test {
    public static void main(String[] args) {
        try(Scanner scanner = new Scanner(System.in)) {
            int i = scanner.nextInt();
            if(i % 2 != 0) {
                assert false;
            }
        } catch(Exception ex) {
            System.out.println("ONE");
        } catch(Error ex) {
            System.out.println("TWO");
        }
    }
}
```

What will be the result of compiling and executing Test class? User input: 1

A - Program ends abruptly

B - ONE

C - No output and program terminates successfully

D - TWO

Exceptions and Assertions - 19

Given code of Test.java file:

```
public class Test {
    private static void div(int i, int j) {
        try {
            System.out.println(i / j);
        } catch(ArithmeticException e) {
            throw (RuntimeException)e;
        }
    }
    public static void main(String[] args) {
        try {
            div(5, 0);
        } catch(ArithmeticException e) {
            System.out.println("AE");
        } catch(RuntimeException e) {
            System.out.println("RE");
        }
    }
}
```

What will be the result of compiling and executing Test class?

A - RE

B - AE

C - Compilation error

D - Program ends abruptly

Exceptions and Assertions - 20

Given code of Test.java file:

```
import java.io.PrintWriter;

public class Test {
    public static void main(String[] args) {
        try(PrintWriter writer = null) {
            System.out.println("HELLO");
        }
    }
}
```

What will be the result of compiling and executing Test class?

A - NullPointerException is thrown at runtime

B - Compilation error

C - HELLO

Exceptions and Assertions - 21

Given code of Test.java file:

```
public class Test {
    public static void main(String[] args) {
        try {
            main(args);
        } catch (Exception ex) {
            System.out.println("CATCH-");
        }
        System.out.println("OUT");
    }
}
```

What will be the result of compiling and executing Test class?

A - OUT

B - None of the System.out.println statements are executed

C - Compilation error

D - CATCH-OUT

Exceptions and Assertions - 22

Given code of Test.java file:

```
import java.sql.SQLException;

public class Test {
    private static void m() throws SQLException {
        try {
            throw new SQLException();
        } catch (Exception e) {
            throw null; //Line 10
        }
    }

    public static void main(String[] args) {
        try {
            m(); //Line 16
        } catch(SQLException e) {
            System.out.println("Caught Successfully.");
        }
    }
}
```

What will be the result of compiling and executing Test class?

A - Line 10 causes compilation failure

B - Line 16 causes compilation failure

C - Caught Successfully

D - Program ends abruptly

Exceptions and Assertions - 23

Given code of Test.java file:

```
class MyException extends RuntimeException {
    public void log() {
        System.out.println("Logging MyException");
    }
}

class YourException extends RuntimeException {
    public void log() {
        System.out.println("Logging YourException");
    }
}

public class Test {
    public static void main(String[] args) {
        try {
            throw new MyException();
        } catch(MyException | YourException ex){
            ex.log();
        }
    }
}
```

What will be the result of compiling and executing Test class?

A - Logging MyException

B - Compilation error

C - Logging YourException

D - Runtime Exception

Exceptions and Assertions - 24

Given code of Test.java file:

```
import java.io.IOException;
import java.sql.SQLException;

class MyResource implements AutoCloseable {
    @Override
    public void close() throws IOException{
        throw new IOException("IOException");
    }

    public void execute() throws SQLException {
        throw new SQLException("SQLException");
    }
}

public class Test {
    public static void main(String[] args) {
        try(MyResource resource = new MyResource()) {
            resource.execute();
        } catch(Exception e) {
            System.out.println(e.getSuppressed()[0].getMessage());
        }
    }
}
```

What will be the result of compiling and executing Test class?

A - IOException

B - Compilation error

C - SQLException

Exceptions and Assertions - 25

Given code of Test.java file:

```
package com.training.ocp;

public class Test {
    private static String msg = "Hello";

    private static String changeMsg(String m) {
        msg = m;
        return null;
    }

    public static void main(String[] args) {
        if(args.length == 0) {
            assert false : changeMsg("Bye");
        }
        System.out.println(msg);
    }
}
```

What will be the result of executing Test class with below command? java -ea com.training.ocp.Test

A - Bye

B - AssertionError is thrown at runtime and program terminates abruptly

C - Hello

Exceptions and Assertions - 26

Given code of Test.java file:

```
public class Test {
    private static void m1() throws Exception {
        throw new Exception();
    }

    public static void main(String[] args) {
        try {
            m1();
        } finally {
            System.out.println("A");
        }
    }
}
```

What will be the result of compiling and executing Test class?

A - A is printed to the console and program ends normally

B - A is printed to the console, stack trace is printed and then program ends abruptly

C - A is printed to the console, stack trace is printed and then program ends normally

D - Compilation error

Exceptions and Assertions - 27

Given code of Test.java file:

```
public class Test {
    public static void main(String[] args) {
        try {
            try {
                System.out.println(1/0);
            } catch(ArithmeticException e) {
                System.out.println("Inner");
                throw e;
            } finally {
                System.out.println("Finally 1");
            }
        } catch(ArithmeticException e) {
            System.out.println("Outer");
        }
    }
}
```

What will be the result of compiling and executing Test class?

A -

```
Inner
Outer
Finally 1
```

B - Compilation Error

C -

```
Inner
Finally 1
```

D -

```
Inner
Finally 1
Outer
```

Exceptions and Assertions - 28

Given code of Test.java file:

```
import java.io.FileNotFoundException;
import java.io.IOException;

abstract class Super {
    public abstract void m1() throws IOException;
}

class Sub extends Super {
    @Override
    public void m1() throws IOException {
        throw new FileNotFoundException();
    }
}

public class Test {
    public static void main(String[] args) {
        Super s = new Sub();
        try {
            s.m1();
        } catch (FileNotFoundException e) {
            System.out.print("M");
        } finally {
            System.out.print("N");
        }
    }
}
```

What will be the result of compiling and executing Test class?

A - Program ends abruptly

B - MN

C - Compilation error

D - N

Exceptions and Assertions - 29

Given code of Test.java file:

```
class Base {
    public void m1() throws NullPointerException {
        System.out.println("Base: m1()");
    }
}

class Derived extends Base {
    public void m1() throws RuntimeException {
        System.out.println("Derived: m1()");
    }
}

public class Test {
    public static void main(String[] args) {
        Base obj = new Derived();
        obj.m1();
    }
}
```

What will be the result of compiling and executing Test class?

A - Compilation error in Test class

B - Base: m1()

C - Compilation error in Derived class

D - Derived: m1()

Exceptions and Assertions - 30

Given code of Test.java file:

```
package com.training.ocp;

public class Test {
    private static void checkStatus(boolean flag) {
        assert flag = true : flag = false;
    }

    public static void main(String[] args) {
        checkStatus(false);
    }
}
```

What will be the result of executing Test class with below command?

java -ea:com.training... com.training.ocp.Test

A - AssertionError is thrown and program terminates abruptly

B - No output and program terminates successfully

C - Compilation error

Exceptions and Assertions - 31

Given code of Test.java file:

```
import java.util.Scanner;

public class Test {
    public static void main(String[] args) {
        try(Scanner scan = new Scanner(System.in)) {
            String s = scan.nextLine();
            System.out.println(s);
            scan = null;
        }
    }
}
```

What will be the result of compiling and executing Test class?

A - Exception is thrown at runtime

B - Compilation error

C - Normal Termination

Exceptions and Assertions - 32

Given code of Test.java file:

```
class MyResource implements AutoCloseable {
    public void execute() {
        System.out.println("Executing");
    }

    @Override
    public void close() {
        System.out.println("Closing");
    }
}

public class Test {
    public static void main(String[] args) {
        try(MyResource resource = new MyResource()) {
            resource.execute();
        }
    }
}
```

What will be the result of compiling and executing Test class?

A - Runtime Exception

B -

```
Executing
Closing
```

C - Compilation Error

D - Executing

Exceptions and Assertions - 33

Given code of Test.java file:

```
class Resource implements AutoCloseable {
    public void close() {
        System.out.println("CLOSE");
    }
}

public class Test {
    public static void main(String[] args) {
        try(Resource r = null) {
            r = new Resource();
            System.out.println("HELLO");
        }
    }
}
```

What will be the result of compiling and executing Test class?

A - Compilation error

B -

```
HELLO
CLOSE
```

C - NullPointerException is thrown at runtime

D - HELLO

Exceptions and Assertions - 34

Given code of Test.java file:

```
import java.io.FileNotFoundException;
import java.io.IOException;

abstract class Super {
    public abstract void m1() throws IOException;
}

class Sub extends Super {
    @Override
    public void m1() throws IOException {
        throw new FileNotFoundException();
    }
}

public class Test {
    public static void main(String[] args) {
        Super s = new Sub();
        try {
            s.m1();
        } catch (FileNotFoundException e) {
            System.out.print("X");
        } catch (IOException e) {
            System.out.print("Y");
        } finally {
            System.out.print("Z");
        }
    }
}
```

What will be the result of compiling and executing Test class?

A - YZ

B - Compilation Error

C - XZ

D - XYZ

Exceptions and Assertions - 35

Given code of Test.java file:

```
public class Test {
    public static void main(String[] args) {
        try {
            check();
        } catch(RuntimeException e) {
            System.out.println(e.getClass().getName()); //Line n1
        }
    }

    private static void check() {
        try {
            RuntimeException re = new RuntimeException(); //Line n2
            throw re; //Line n3
        } catch(RuntimeException e) {
            System.out.println(1);
            ArithmeticException ex = (ArithmeticException)e; //Line
         n4
            System.out.println(2);
            throw ex;
        }
    }
}
```

What will be the result of compiling and executing Test class?

A -

```
1
2
java.lang.ArithmeticException
```

B -

```
1
2
java.lang.RuntimeException
```

C -

```
1
java.lang.ClassCastException
```

D -

```
1
java.lang.ArithmeticException
```

E -

```
1
java.lang.RuntimeException
```

Exceptions and Assertions - 36

What will be the result of compiling and executing the following program?

```
public class Test {
    private static String s;
    public static void main(String[] args) {
        try {
            System.out.println(s.length());
        } catch(NullPointerException | RuntimeException ex) {
            System.out.println("DONE");
        }
    }
}
```

A - Executes successfully but no output

B - DONE

C - Compilation error

Exceptions and Assertions - 37

Given code of Test.java file:

```
class MyException extends RuntimeException {}

class YourException extends RuntimeException {}

public class Test {
    public static void main(String[] args) {
        try {
            throw new YourException();
        } catch(MyException | YourException e){
            e = null;
        }
    }
}
```

What will be the result of compiling and executing Test class?

A - Compilation error

B - Runtime Exception

C - Nothing is printed on to the console and program terminates successfully

Exceptions and Assertions - 38

Given code of Test.java file:

```
import java.sql.SQLException;

public class Test {
    private static void m() throws SQLException {
        try {
            throw new SQLException();
        } catch (Exception e) {
            e = null; //Line 10
            throw e; //Line 11
        }
    }

    public static void main(String[] args) {
        try {
            m(); //Line 17
        } catch(SQLException e) {
            System.out.println("Caught Successfully.");
        }
    }
}
```

What will be the result of compiling and executing Test class?

A - Caught Successfully.

B - Line 17 causes compilation failure

C - Program ends abruptly

D - Line 10 causes compilation failure

E - Line 11 causes compilation failure

Exceptions and Assertions - 39

Given code of Test.java file:

```
class MyException1 extends RuntimeException {}

class MyException2 extends RuntimeException {}

public class Test {
    private static void m() {
        try {
            throw new RuntimeException();
        } catch(RuntimeException ex) {
            throw new MyException1();
        } finally {
            throw new MyException2();
        }
    }

    public static void main(String[] args) {
        try {
            m();
        } catch(MyException1 e) {
            System.out.println("MyException1");
        } catch(MyException2 e) {
            System.out.println("MyException2");
        } catch (RuntimeException e) {
            System.out.println("RuntimeException");
        }
    }
}
```

What will be the result of compiling and executing Test class?

A - MyException1

B - RuntimeException

C - MyException2

Exceptions and Assertions - 40

Given code of Test.java file:

```
class MyResource implements AutoCloseable {
    public void execute() {
        System.out.println("Executing");
    }

    @Override
    public void close() throws Exception {
        System.out.println("Closing");
    }
}

public class Test {
    public static void main(String[] args) {
        try(MyResource resource = new MyResource()) {
            resource.execute();
        }
    }
}
```

What will be the result of compiling and executing Test class?

A - Runtime Exception

B - Executing

C - Compilation Error

D -

```
Executing
Closing
```

Exceptions and Assertions - 41

Given code of Test.java file:

```
public class Test {
    private static void checkStatus() {
        /*INSERT*/
    }

    private static String get() {
        return "TEST";
    }

    public static void main(String[] args) {
        try {
            checkStatus();
        } catch (AssertionError ae) {
            System.out.println(ae.getCause());
        }
    }
}
```

Which of the following options can replace /*INSERT*/ such that there are no compilation error?

A - assert 1 == 2 : Test::get;

B - assert 1 == 2 : return 1;

C - assert 1 == 2 : 1;

D - assert 1 == 2 : () -> “a”;

Exceptions and Assertions - 42

Given code of Test.java file:

```
import java.util.Scanner;

public class Test {
    public static void main(String[] args) {
        System.out.print("Enter some text: ");
        try(Scanner scan = new Scanner(System.in)) {
            String s = scan.nextLine();
            System.out.println(s);
            scan.close();
            scan.nextLine();
        }
    }
}
```

What will be the result of compiling and executing Test class?

User input is: HELLO

A - Compilation error

B - On execution program terminates successfully after printing ‘HELLO’ on to the console

C - Runtime Exception

Exceptions and Assertions - 43

Given code of Test.java file:

```
import java.io.FileNotFoundException;

public class Test {
    public static void main(String[] args) {
        try {
            System.out.println(1);
        } catch (NullPointerException ex) {
            System.out.println("ONE");
        } catch (FileNotFoundException ex) {
            System.out.println("TWO");
        }
        System.out.println("THREE");
    }
}
```

What will be the result of compiling and executing Test class?

A -

```
ONE
THREE
```

B - None of the System.out.println statement is executed

C - Compilation error

D -

```
TWO
THREE
THREE
```

Exceptions and Assertions - 44

Given code of Test.java file:

```
import java.io.PrintWriter;

public class Test {
    public static void main(String[] args) {
        try(PrintWriter writer;) {
            writer = new PrintWriter(System.out);
            writer.println("HELLO");
        }
    }
}
```

What will be the result of compiling and executing Test class?

A - Compilation error

B - Runtime exception

C - HELLO

Exceptions and Assertions - 45

Given code of Test.java file:

```
public class Test {
    private static void m1() {
        System.out.println(1/0);
    }

    public static void main(String[] args) {
        try {
            m1();
        } finally {
            System.out.println("A");
        }
    }
}
```

What will be the result of compiling and executing Test class?

A - Compilation error

B - A is printed to the console and program ends normally

C - A is printed to the console, stack trace is printed and then program ends abruptly

D - A is printed to the console, stack trace is printed and then program ends normally

Exceptions and Assertions - 46

Which of the following keywords is used to manually throw an exception?

A - thrown

B - catch

C - throw

D - throws

Exceptions and Assertions - 47

Given code of Test.java file:

```
public class Test {
    private static void div(int i, int j) {
        try {
            System.out.println(i / j);
        } catch(ArithmeticException e) {
            Exception ex = new Exception(e);
            throw ex;
        }
    }
    public static void main(String[] args) {
        try {
            div(5, 0);
        } catch(Exception e) {
            System.out.println("END");
        }
    }
}
```

What will be the result of compiling and executing Test class?

A - Compilation error

B - END is printed and program terminates successfully

C - END is printed and program terminates abruptly

D - END is not printed and program terminates abruptly

Exceptions and Assertions - 48

Given code of Test.java file:

```
class TestException extends Exception {
    public TestException() {
        super();
    }

    public TestException(String s) {
        super(s);
    }
}

public class Test {
    public void m1() throws __________ {
        throw new TestException();
    }
}
```

For the above code, fill in the blank with one option.

A - Error

B - Object

C - Exception

D - RuntimeException

Exceptions and Assertions - 49

Given code of Test.java file:

```
package com.training.ocp;

public class Test {
    private static void checkStatus() {
        assert 1 == 2 : 2 == 2;
    }

    public static void main(String[] args) {
        try {
            checkStatus();
        } catch (AssertionError ae) {
            System.out.println(ae.getMessage());
        }
    }
}
```

What will be the result of executing Test class with below command?

java -ea com.training.ocp.Test

A - true

B - Compilation error

C - false

D - null

Exceptions and Assertions - 50

Given code of Test.java file:

```
package com.training.ocp;

public class Test {
    private static void checkStatus(boolean flag) {
        assert flag : flag = true;
    }

    public static void main(String[] args) {
        checkStatus(false);
    }
}
```

What will be the result of executing Test class with below command?

java -ea:com.training... com.training.ocp.Test

A - AssertionError is thrown and program terminates abruptly

B - No output and program terminates successfully

C - Compilation error

Exceptions and Assertions - 51

Given code of Test.java file:

```
import java.io.IOException;
import java.sql.SQLException;

class MyResource implements AutoCloseable {
    @Override
    public void close() throws IOException{

    }

    public void execute() throws SQLException {
        throw new SQLException("SQLException");
    }
}

public class Test {
    public static void main(String[] args) {
        try(MyResource resource = new MyResource()) {
            resource.execute();
        } catch(Exception e) {
            System.out.println(e.getSuppressed().length);
        }
    }
}
```

What will be the result of compiling and executing Test class?

A - 1

B - 0

C - NullPointerException is thrown

Exceptions and Assertions - 52

Given code of Test.java file:

```
class Resource implements AutoCloseable {
    public void close() {
        System.out.println("CLOSE");
    }
}

public class Test {
    public static void main(String[] args) {
        try(Resource r = null) {
            System.out.println("HELLO");
        }
    }
}
```

What will be the result of compiling and executing Test class?

A - Compilation error

B - NullPointerException is thrown at runtime

C -

```
HELLO
CLOSE
```

D - HELLO

Exceptions and Assertions - 53

Given code of Test.java file:

```
import java.io.FileNotFoundException;
import java.io.IOException;

abstract class Super {
    public abstract void m1() throws IOException;
}

class Sub extends Super {
    @Override
    public void m1() throws IOException {
        throw new FileNotFoundException();
    }
}

public class Test {
    public static void main(String[] args) {
        Super s = new Sub();
        try {
            s.m1();
        } catch (IOException e) {
            System.out.print("A");
        } catch(FileNotFoundException e) {
            System.out.print("B");
        } finally {
            System.out.print("C");
        }
    }
}
```

What will be the result of compiling and executing Test class?

A - class Test causes compilation error

B - AC

C - class Sub causes compilation error

D - BC

Exceptions and Assertions - 54

Given code of Test.java file:

```
import java.sql.SQLException;

public class Test {
    private static void m() throws SQLException {
        throw null; //Line 7
    }

    public static void main(String[] args) {
        try {
            m(); //Line 12
        } catch(SQLException e) {
            System.out.println("Caught Successfully.");
        }
    }
}
```

What will be the result of compiling and executing Test class?

A - Line 12 causes compilation failure

B - Line 7 causes compilation failure

C - Program ends abruptly

D - Caught Successfully