1) What will be the output of the program?

public class Foo

{

public static void main(String[] args)

{

try

{

return;

}

finally

{

System.out.println( "Finally" );

}

}

}

**A. Finally** B. Compilation fails. C. The code runs with no output. D. An exception is thrown at runtime.

2) What will be the output of the program?

try

{

int x = 0;

int y = 5 / x;

}

catch (Exception e)

{

System.out.println("Exception");

}

catch (ArithmeticException ae)

{

System.out.println(" Arithmetic Exception");

}

System.out.println("finished");

A. finished B. Exception **C. Compilation fails**. D. Arithmetic Exception

Answer: Option C

3)What will be the output of the program?

public class X

{

public static void main(String [] args)

{

try

{

badMethod();

System.out.print("A");

}

catch (Exception ex)

{

System.out.print("B");

}

finally

{

System.out.print("C");

}

System.out.print("D");

}

public static void badMethod()

{

throw new Error(); /\* Line 22 \*/

}

}

A. ABCD

B. Compilation fails.

**C. C is printed before exiting with an error message.**

D. BC is printed before exiting with an error message.

Answer & Explanation

Answer: Option C

4)What will be the output of the program?

public class X

{

public static void main(String [] args)

{

try

{

badMethod();

System.out.print("A");

}

catch (RuntimeException ex) /\* Line 10 \*/

{

System.out.print("B");

}

catch (Exception ex1)

{

System.out.print("C");

}

finally

{

System.out.print("D");

}

System.out.print("E");

}

public static void badMethod()

{

throw new RuntimeException();

}

}

A. BD B. BCD **C. BDE** D. BCDE

Answer: Option C

5) What will be the output of the program?

public class RTExcept

{

public static void throwit ()

{

System.out.print("throwit ");

throw new RuntimeException();

}

public static void main(String [] args)

{

try

{

System.out.print("hello ");

throwit();

}

catch (Exception re )

{

System.out.print("caught ");

}

finally

{

System.out.print("finally ");

}

System.out.println("after ");

}

}

A. hello throwit caught

B. Compilation fails

C. hello throwit RuntimeException caught after

**D. hello throwit caught finally after**

Answer & Explanation

Answer: Option D

6) What will be the output of the program?

public class Test

{

public static void aMethod() throws Exception

{

try /\* Line 5 \*/

{

throw new Exception(); /\* Line 7 \*/

}

finally /\* Line 9 \*/

{

System.out.print("finally "); /\* Line 11 \*/

}

}

public static void main(String args[])

{

try

{

aMethod();

}

catch (Exception e) /\* Line 20 \*/

{

System.out.print("exception ");

}

System.out.print("finished"); /\* Line 24 \*/

}

}

A. finally B. exception finished **C. finally exception finished** D. Compilation fails

Answer: Option C

7) What will be the output of the program?

public class X

{

public static void main(String [] args)

{

try

{

badMethod();

System.out.print("A");

}

catch (Exception ex)

{

System.out.print("B");

}

finally

{

System.out.print("C");

}

System.out.print("D");

}

public static void badMethod() {}

}

A. AC B. BC **C. ACD** D. ABCD

Answer: Option C

8) What will be the output of the program?

public class X

{

public static void main(String [] args)

{

try

{

badMethod(); /\* Line 7 \*/

System.out.print("A");

}

catch (Exception ex) /\* Line 10 \*/

{

System.out.print("B"); /\* Line 12 \*/

}

finally /\* Line 14 \*/

{

System.out.print("C"); /\* Line 16 \*/

}

System.out.print("D"); /\* Line 18 \*/

}

public static void badMethod()

{

throw new RuntimeException();

}

}

A. AB B. BC C. ABC **D. BCD**

Answer: Option D

9) What will be the output of the program?

public class MyProgram

{

public static void main(String args[])

{

try

{

System.out.print("Hello world ");

}

finally

{

System.out.println("Finally executing ");

}

}

}

A. Nothing. The program will not compile because no exceptions are specified.

B. Nothing. The program will not compile because no catch clauses are specified.

C. Hello world.

**D. Hello world Finally executing**

Answer: Option D

10) What will be the output of the program?

class Exc0 extends Exception { }

class Exc1 extends Exc0 { } /\* Line 2 \*/

public class Test

{

public static void main(String args[])

{

try

{

throw new Exc1(); /\* Line 9 \*/

}

catch (Exc0 e0) /\* Line 11 \*/

{

System.out.println("Ex0 caught");

}

catch (Exception e)

{

System.out.println("exception caught");

}

}

}

**A. Ex0 caught** B. exception caught

C. Compilation fails because of an error at line 2.

D. Compilation fails because of an error at line 9.

Answer: Option A

11. public class Main {

public static void main(String[] args) {

int a[]=null;

badMethod(a);

}

public static void badMethod(int a[])

{

System.out.println(a[2]);

}

}

What will be the output of this code?

a) throws ArrayIndexOutOfBoundsException **b) throws NullPointerException** c) No exception is thrown d) Compilation error

12. Which of the following Exception class is not a subclass of Runtime Exception?

a)ArithmeticException b)NumberFormatException c) NullPointerException d) **FileNotFoundException.**

13. What is the output of the following code?

import java.io.IOException;

class Base

{

void show() throws IOException

{

}

}

class Derived extends Base

{

void show() throws Exception

{

}

}

a) code compiles **b) compilation error** c) IOException is thrown d) none of these.

14. What is the output of the following code?

class Base

{

void show() throws Exception

{

}

}

class Derived extends Base

{

void show() throws RuntimeException

{

}

}

**a) code compiles** b) compilation error c) Exception is thrown d) none of these.

15. What is the output of the code?

public class Main

{

public static void main(String args[])

{

try

{

double x = 0;

double y = 5 / x;

System.out.println(y);

}

catch (ArithmeticException ae)

{

System.out.println(" Arithmetic Exception");

}

}

}

a) Arithmetic Exception b) No Exception **c) Infinity** d) Compilation error

16) Throwable is the Super class of Error and Exception.

**a) True** b) False.

17) public class Main

{

static void main(String args[])

{

}

}

What is the output of the code?

a) Runtime-Exception is thrown **b) Runtime-Error is thrown** c) compilation error d) None of these.

18. FileNotFoundException class is defined in which package

a) java.sql b) java.lang **c)java.io** d)java.exceptions

19.

import java.io.\*;

public class CrypticCatch

{ public static void main(String[] args) throws Exception

{

try

{

try

{

try

{ throw new FileNotFoundException();

}

catch(Exception e3)

{ throw e3; }

}

catch(IOException e2)

{ throw e2; }

}

catch(FileNotFoundException e1)

{ System.out.println("File not found exception caught");

}

System.out.println("Exception handled successfully");

}

}

What will be the output of the above program?

a. Exception handled successfully

b. Compile time error. Since exceptions should be caught in reversed hierarchy order

**c. File not found exception caught Exception handled successfully**

d. Runtime error

e. File not found exception caught

ans:c

20. Which of the following options are true? (Choose 2)

a. In a try-catch-finally structure, finally block and catch block can be placed in any order

b. On using nested try-catch blocks, only the outer most try catch block can have the finally block

**c. The finally block can have another try-catch-finally block nested inside**

**d. The catch block can have another try-catch-finally block**

21.

import java.io.\*;

public class Main

{

public static void main(String[] args) throws Exception

{

try

{

try

{ throw new FileNotFoundException();

}

catch(Exception e3)

{ throw e3;

}

finally

{

System.out.println("Inner-try Finally");

}

}

catch(FileNotFoundException e1)

{ System.out.println("File not found exception caught");

}

finally

{

System.out.println("Outer-try Finally");

}

}

}

a) Compilation error

**b) Inner-try Finally File not found exception caught Outer-try Finally**

c) Outer-try Finally File not found exception caught Inner-try Finally

d) None of these.

22.

Consider the following program:

1.

class CheckedException extends RuntimeException { }

2. class UncheckedException extends Exception { }

3. public class Check

{ 4. public static void main(String args[])

{ 5. generateException1();

6. generateException2();

7. }

8.

9. private static void generateException1()

{ 10. throw new CheckedException();

11.

}

12.

13. private static void generateException2()

{ 14. throw new UncheckedException();

15.

}

16.

} Which of the following is true regarding the above given program?

a. No compilation error but throws RuntimeException on running the code

b. Compilation error at line 5

c. Compilation error at line 10

**d. Compilation error at line 14**

e. Compilation error at line 6

23.

Consider the following code:

class A { }

class B extends A { }

public class Code2

{

public void method(A a)

{ System.out.println("A");

}

public void method(B b)

{ System.out.println("B"); }

public static void main(String args[])

{

new Code2().method(new Object());

}

} Which of the following will be the output for the above code?

a. Prints: B

b. Throws ClassCastException at runtime

**c. Compilation Error -ans**

d. Prints: A

24. class UserDefinedException extends Error { }

public class TasteIt {

public static void main(String args[]) {

try {

try {

throw new Error();

}

catch(UserDefinedException u1) {

throw u1;

}

catch(Exception e1) {

System.out.println("This is the required output");

}

finally {

throw new UserDefinedException();

}

}

catch(UserDefinedException u2) {

System.out.println("This is not the output");

}

catch(Error e2) {

System.out.println("This is the output");

}

}

}

What will be the output for the above program?

Answer:

a. Runtime Error

b. This is the required output

c. Compile-time error

**d. This is not the output -ans**

e. This is the output

25.

Consider the following program:

class RE {

public static void main(String args[]) {

try {

String s = null;

System.out.println(s.length());

}

catch(NullPointerException npe) {

System.out.println("NullPointerException handled");

throw new Exception(npe.getMessage());

}

}

}

What will be the output of the above program?

Answer:

a. Code compiles and on running creates and throws Exception type object

b. Code compiles but run without any output

**c. Code does not complile -ans**

d. Code compiles and on running it prints "NullPointerException

26.

Consider the following program:

public class TryIt {

public static void main(String args[]) {

try {

int i = 0;

try {

i = 100 / i;

}

catch(Error e) {

System.out.println("Divide by Zero error");

}

System.out.println("Error Handled");

}

catch(Exception e) {

System.out.println("Unexpected exception caught");

}

}

}

What will be the output of the above program?

Answer:

a. Divide by Zero error

Error Handled

Unexpected exception caught

b. Program compiles and runs without any output

c. Divide by Zero error

Error Handled

d. Unexpected exception caught -ans

e. Error Handled

Unexpected exception caught

27. Consider the following program:

public class Exp4

{ static String s = "smile!..";

public static void main(String[] args)

{ new Exp4().s1();

System.out.println(s);

}

void s1()

{

try

{ s2();

}

catch (Exception e)

{ s += "morning"; }

}

void s2() throws Exception

{

s3();

s += "evening";

s3();

s += "good";

}

void s3() throws Exception

{

throw new Exception();

}

} What will be the output of the above program?

Answer:

a. smile!..eveningmorning

b. smile!..morningevening

c. smile!..morning -ans

|  |
| --- |
|  |
| d. smile!..morningeveninggood |
| e. smile!.. |

28. Consider the following scenario: Here is part of the hierarchy of exceptions that may be thrown during file IO operations: Exception

+-IOException

+-File Not Found Exception

You have a method X that is supposed to open a file by name and read data from it. Given that X does not have any try-catch statements, which of the following option is true?

Answer:

a. The method X must be declared as throwing FileNotFoundException

b. Any method calling X must use try-catch, specifically catching FileNotFoundException

c. The method X must be declared as throwing IOException or Exception -ans

d. No special precautions need be taken

29. Consider the following scenario:

You are writing a set of classes related to cooking and have created your own

exception hierarchy derived from java.lang.Exception as follows:

Exception

+-Bad TasteException

+-Bitter Exception

+-Sour Excpetion

BadTasteException is defined as an abstract class. You have a method eatMe

that may throw a BitterException or a SourException.

Which of the following method declarations are acceptable to the compiler?

(Choose 3)

Answer:

a. public void eatMe() throws BitterException, SourException -ans

b. public void eatMe() throws Throwable

c. public void eatMe() throw BadTasteException

d. public void eatMe() throws BadTasteException -ans

e. public void eatMe() throws RuntimeException – ans

30. Consider the following program:

class CatchableException extends Throwable { }

class ThrowableException extends CatchableException { }

public class ThrowCatchable

{ public static void main(String args[])

{ try { tryThrowing();

}

catch(CatchableException c)

{ System.out.println("Catchable caught"); }

finally

{ tryCatching(); }

}

static void tryThrowing() throws CatchableException

{ try { tryCatching();

throw new ThrowableException();

}

catch(NullPointerException re)

{ throw re; }

}

static void tryCatching()

{ System.out.println(null + " pointer exception");

}

} What will be the output of the above program?

Answer: a. null pointer exception Catchable caught null pointer exception- ans

b. runtime error

c. Catchable caught null pointer exception null pointer exception

d. compile-time error

e. null pointer exception null pointer exception Catchable caught