**Core Java, Quiz-1, Date: 06/09/2018**

**1. names that refer to values or names - letters, digits, \_, and $.**

**-rules for creating a name in a program**

a. runtime error

b. input error

c. identifier

d. expression

**2. a very large int, more precise**

1. int type

2. double type

3. constant

4. long type

**3. casting from a small type to a larger type, this is done manually.**

a. floating-point number

b. assignment statement

c. widening (of types)

d. narrowing (of types)

**4. abstract is a---**

a. keyword

b. statement

c. int type

d. final keyword

**5. escape sequence**

1. final

2. keywords

3. Bit

4. \"

**6. Numbers with a decimal point (var double)**

1. floating-point number

2. concatenate strings

3. Floating point

4. widening (of types)

**7. occurs when a program does not perform the way it was intended to**

1. preincrement

2. logic error

3. identifier

4. compiler

**8. float**

1. name of type

2. name of a type

3. variable name

4. data type

**9. same as lazy operator - && or || (and, or)**

1. dangling else ambiguity

2. selection statement

3. floating-point number

4. short circuit operator

**10. when else matches with the most recent if statement**

1. assignment statement

2. dangling else ambiguity

3. short circuit operator

4. Conditional Expression ( ? :)

**11. /\*XXXXXXXXXXXXX\*/**

1. Block Comment

2. postincrement

3. preprocessor

4. Floating point

**12. ++ placed after variable. uses original variable in expression then increases by 1**

1. postdecrement

2. preincrement

3. statement

4. postincrement

**13. an environment for developing Java programs**

1. Conditional Expression ( ? :)

2. Integrated development environment

3. nextDouble

4. Augmented assignment operators

**14. instructions for a high-level program**

1. statement

2. nextDouble

3. postdecrement

4. data type

**15. evaluates to the value to be assigned to a variable (=)**

1. assignment statement

2. widening (of types)

3. Boolean Expression

4. assignment operator

**16. checks the validity of a bytecode**

1. Bytecode

2. bytecode verifier

3. widening (of types)

4. logic error

**17. 3.14159E1**

1. floating point/pi

2. octa integer

3. Boolean Value

4. illegal identifier

**18. ++**

1. name of type

2. escape character

3. Boolean Expression

4. increment operator

**19. -- placed before variable. decreases variable by one, then uses it in the expression**

1. runtime error

2. predecrement

3. wildcard import

4. postdecrement

**20. an operation that converts a value of one data type into a value of another data type**

1. statement

2. casting

3. keyword

4. comment

**21. an exact number, 1, 4 or 10**

1. Assembler

2. statement

3. int type

4. constant

**22. do, else, and break**

1. keyword

2. keywords

3. operands

4. operators

**23. An expression that evaluates a Boolean value to be true or false**

1. Boolean Expression

2. boolean operators

3. scope of a variable

4. dangling else ambiguity

**24. casting a data type from a large range to a smaller range - Java does this automatically**

1. narrowing (of types)

2. dangling else ambiguity

3. floating-point number

4. runtime error

**25. =**

1. assignment statement

2. increment operator

3. assignment operator

4. decrement operator

**26. result from errors in code construction, such as misspellings, wrong punctuation, etc.**

1. syntax error

2. Bytecode

3. source code/program

4. interpreter

**27. using no breaks in a switch**

1. conditional operator

2. concatenate strings

3. primitive data type

4. fall-through behavior

**28. denotes names**

1. final

2. final keyword

3. Bit

4. Block

**29. import**

1. directive

2. preprocessor

3. identifier

4. name of type

**30. errors that cause a program to terminate early, an impossible operation is detected**

1. runtime error

2. logic error

3. input error

4. postincrement

**Core Java, Quiz-2, Date: 08/09/2018**

**1. imports all the classes in a package by using a \* (import java.util.\*;)**

1. operator precedence

2. wildcard import

3. floating-point number

4. input error

**2. translates a Java source file into a Java bytecode file**

1. compiler

2. comment

3. variable

4. Assembler

**3. var++, + and -, casting, !, \* / %, + - concaction, (See page 105)**

1. assignment statement

2. operator precedence

3. selection statement

4. Application Program Interface ( API)

**4. --**

1. assignment

statement

2. decrement operator

3. postdecrement

4. name of type

**5. 8 bits to 1 byte**

1. illegal identifier

2. operators

3. octa integer

4. Byte

**6. \n**

1. variable name

2. name of type

3. decrement operator

4. escape character

**7. ? : for if statement shorthand**

1. Relational Operators (Boolean)

2. selection statement

3. Conditional Expression ( ? :)

4. conditional operator

**8. a type**

1. directive

2. String

3. operands

4. identifier

**9. the amount of space between pixels, measured in millimeters**

1. overflow

2. logic error

3. scope of a variable

4. dot pitch

**10. int**

1. int type

2. name of type

3. data type

4. final keyword

**11. denotes a value as a constant**

1. final keyword

2. final

3. input error

4. preprocessor

**12. Reserved words that have a specific meaning in java and cannot be used for variables**

1. preincrement

2. keywords

3. keyword

4. overflow

**13. -- placed after variable. uses original variable in expression then decreases by 1**

1. Block Comment

2. postdecrement

3. statement

4. postincrement

**14. represents a value stored in the computers memory**

1. operands

2. variable

3. data type

4. statement

**15. int, real numbers, characters and booleans**

1. dangling else ambiguity

2. selection statement

3. short circuit operator

4. primitive data type

**16. Similar to machine instructions, but can run on any platform with a JVM**

1. constant

2. Literal

3. Bytecode

4. int type

**17. a class name in the system library that contains different java functions**

1. literal

2. util

3. Assembler

4. casting

**18. !, &&, ||, ^**

1. Floating point

2. bytecode verifier

3. Boolean Expression

4. boolean operators

**19. 075**

1. floating point/pi

2. name of type

3. identifier

4. octa integer

**20. statements that let you choose actions with alternative choices**

1. Boolean Expression

2. primitive data type

3. operator precedence

4. selection statement

**21. on a program denoted by //xxxxx or /\*xxxx\*/**

1. comment

2. casting

3. double type

4. constant

**22. method that is applied to objects of Scanner**

1. Literal

2. Assembler

3. statement

4. nextDouble

**23. determine the order in which operators are evaluated**

1. operator associativity

2. Conditional Expression ( ? :)

3. interpreter

4. assignment statement

**24. scientific notation**

1. postincrement

2. Floating point

3. boolean operators

4. final keyword

**25. <, <=, ==, !=, >, >=**

1. Application Program Interface ( API)

2. operators

3. Floating point

4. Relational Operators (Boolean)

**26. a high-level program's code**

1. bytecode verifier

2. source code/program

3. concatenate strings

4. interpreter

**27. can be true or false**

1. double type

2. Boolean Value

3. bytecode verifier

4. long type

**28. ;**

1. widening (of types)

2. increment operator

3. statement terminator

4. assignment statement

**29. import statement**

1. preprocessor

2. preincrement

3. expression

4. operators

**30. If you try to store a value in a data type that cannot handle it.**

1. overflow

2. compiler

3. Assembler

4. dot pitch

**Core Java, Quiz-3, Date: 13/09/2018**

**1. consists of a set of separate programs, each invoked from a command line, for developing and** **testing Java programs**

1. identifier

2. Java Development Toolkit

3. postdecrement

4. expression

**2. using the (+) sign to combine strings**

1. concatenate strings

2. scope of a variable

3. operator precedence

4. Conditional Expression ( ? :)

**3. Evaluates an expression based on a condition (pg 103)**

1. Conditional Expression ( ? :)

2. assignment statement

3. scope of a variable

4. compiler

**4. char**

1. data type

2. name of type

3. runtime error

4. int type

**5. translates source code into machine code**

1. predecrement

2. interpreter

3. util

4. logic error

**6. a device used to translate assembly-language programs into machine code**

1. variable

2. Assembler

3. predecrement

4. overflow

**7. +, -, \*, /, %**

1. operands

2. preprocessor

3. long type

4. operators

**8. the kind of data stored in each variable**

1. data type

2. nextDouble

3. statement

4. dot pitch

**9. Variable**

1. preprocessor

2. identifier

3. directive

4. String

**10. a constant value that appears directly in a program**

1. variable

2. comment

3. input error

4. literal

**11. input, process, output - describes simple code**

1. IPO

2. variable

3. input error

4. comment

**12. A library in Java that contains predefined classes and interfaces**

1. Relational Operators (Boolean)

2. Integrated development environment

3. Application Program Interface ( API)

4. widening (of types)

**13. the values operated on by a operator**

1. dot pitch

2. operands

3. variable

4. overflow

**14. +=, -=, \*\*=, /= and %= (i+= 8 is i = i + 8)**

1. Boolean Expression

2. dangling else ambiguity

3. Augmented assignment operators

4. assignment statement

**15. The part of a program where the variable can be referenced**

1. dot pitch

2. selection statement

3. bytecode verifier

4. scope of a variable

**16. ++ placed before variable. increases variable by one, then uses it in the expression**

1. postincrement

2. predecrement

3. logic error

4. preincrement

**17. real numbers, decimal places, twice as precise as float**

1. long type

2. dot pitch

3. double type

4. int type

**18. Binary digits**

1. directive

2. Block

3. Literal

4. Bit

**19. represents a computation involving values, variables, and operators that, taking them together,** **evaluates to a value**

1. preincrement

2. expression

3. runtime error

4. identifier

**20. Occurs when the user inputs a value the program cannot handle**

1. wildcard import

2. overflow

3. input error

4. double type

**21. Bool**

1. variable name

2. preincrement

3. name of a type

4. predecrement

**22. Constant value directly in a program that stands for itself**

1. operands

2. compiler

3. Literal

4. overflow

**23. a number in the program that never changes, denoted by "final"**

1. constant

2. compiler

3. input error

4. double type

**24. -128 to 127**

1. long type

2. byte type

3. illegal identifier

4. int type

**25. uses a short descriptive word to represent each of the machine-language instructions**

1. Assembler

2. runtime error

3. wildcard import

4. Assembly Language

**26. anything inside of a {xxxxxx}**

1. Block

2. IPO

3. keyword

4. final

**27. 4thQtrSales**

1. identifier

2. floating point/pi

3. illegal identifier

4. preprocessor

**Core Java, Quiz-4, Date: 16/09/2018**

1. Used to execute a loop body a fixed number of times; has three parts: initial action that initializes a control variable, the loopcontinuation-condition,

and execution after each iteration and is often used to adjust the control variable

1. While Loop
2. Nested Loop
3. Do-While Loop
4. For Loop

2. The actual value that is passed into the method by a caller

1. Formal Parameter/Parameter
2. Sentinel Value
3. Scope of a Variable
4. Actual Parameter/Argument

3. Using a method without knowing how it is implemented

1. Information Hiding/Encapsulation
2. Instance Method
3. Off-by-One Error (Arrays)
4. Encoding

4. A loop statement that executes infinitely

1. Sentinel Value
2. Instance Method
3. Nested Loop
4. Infinite Loop

5. Controls how many times an operation or sequence of operations performed in succession

1. Index
2. Pre-Test Loop
3. Loop
4. While Loop

6. Consists of an outer loop and one or more inner loops

1. Instance Method
2. Nested Loop
3. Post-Test Loop
4. Pre-Test Loop

7. Keywords such as public, static, and final

1. Searching
2. Modifier
3. Encoding
4. Infinite Loop

8. Separating the use of a method from its implementation

1. Method Abstraction
2. Method Signature
3. Output Redirection
4. Scope of a Variable

9. Only can be invoked from a specific string instance

1. Continue Statement
2. Instance Method
3. Nested Loop
4. Infinite Loop

10. The process of looking for a specific element in an array

1. Modifier
2. Scope of a Variable
3. Searching
4. String

11. \ (the backslash)

1. Actual Parameter/Argument
2. Method Signature
3. Escape Character
4. Break Statement

12. The part of the loop that contains the statements to be repeated

1. Loop Body
2. Pass-By-Value
3. Char type
4. Searching

13. A numeric association to an element in a collection of data; the position of an object in an array

1. Index
2. Stub
3. Loop
4. Method Overloading

14. A special value that signifies the end of the loop

1. Nested Loop
2. Sentinel Value
3. Method Signature
4. Static Method

15. While & For Loops; the continuation condition is checked before the loop body is executed

1. Pre-Test Loop
2. Nested Loop
3. While Loop
4. Post-Test Loop

16. How characters are encoded

1. Escape Sequence
2. Encoding Scheme
3. Continue Statement
4. Indexed Variable

17. When there are two or more possible matches for the invocation of a method, but the compiler is unable to determine the

best match

1. Anonymous Array
2. Ambiguous Invocation
3. Input Redirection
4. While Loop

18. A simple, incomplete version of a method

1. Sentinel Value
2. Stub
3. Loop
4. Index

19. the variable inside the brackets, [ ]

1. Method Signature
2. Scope of a Variable
3. Anonymous Array
4. Indexed Variable

20. The value of the argument passed to the parameter when a method is invoked with an argument

1. Pre-Test Loop
2. Post-Test Loop
3. Loop Body
4. Pass-By-Value

21. Sends the output to a file rather than displaying it on the console

1. Input Redirection
2. Output Redirection
3. Garbage Collection
4. Method Abstraction

22. Immediately ends the innermost loop

1. Encoding Scheme
2. Break Statement
3. Continue Statement
4. Method Signature

23. A character data type used to represent a single character

1. Loop Body
2. Iteration
3. Static Method
4. Char type

24. Specifies how an item should be displayed, which may be a numeric value, character, Boolean value, or string. Begins with %

1. Input Redirection
2. Post-Test Loop
3. Format Specifier
4. Pre-Test Loop

25. A non-instant method that can be invoked without using an object

1. Nested Loop
2. Char type
3. Static Method
4. Loop Body

**Core Java, Quiz-5, Date: 18/09/2018**

1. Only ends a current iteration

1. Continue Statement
2. Output Redirection
3. Encoding Scheme
4. Break Statement

2. When there are two methods have the same name but different parameter lists within one class

1. Off-by-One Error (Arrays)
2. Pre-Test Loop
3. Method Signature
4. Method Overloading

3. \\ (two backslashes)

1. Escape Sequence
2. Encoding Scheme
3. Break Statement
4. Escape Character

4. Mapping a character to its binary representation

1. Information Hiding/Encapsulation
2. String
3. Nested Loop
4. Encoding

5. When there is no explicit reference variable for an array

1. Anonymous Array
2. Instance Method
3. Scope of a Variable
4. Ambiguous Invocation

6. A one-time execution of a loop body

1. Infinite Loop
2. Iteration
3. Nested Loop
4. Searching

7. Stores a fixed-size sequential collection of elements in the same type

1. Array
2. Anonymous Array
3. Loop Body
4. Index

8. combines the declaration, creation, and initialization of an array in one statement

1. Method Abstraction
2. Array Initializer
3. Pre-Test Loop
4. Pass-By-Value

9. A technique of writing software where you gradually add error checking and functionality

1. Off-by-One Error (Arrays)
2. Divide-and-Conquer / Stepwise Refinement
3. Pass-By-Value
4. Formal Parameter/Parameter

10. When one references the first element in an array with index [ 1 ], but it should be [ 0 ]

1. Formal Parameter/Parameter
2. Off-by-One Error (Arrays)
3. Anonymous Array
4. Off-by-One Error (Loops)

11. Executing a loop one more/less time accidentally

1. Off-by-One Error (Loops)
2. Method Abstraction
3. Actual Parameter/Argument
4. Off-by-One Error (Arrays)

12. similar to while loop; executes

the loop body first and then checks the loop-continuation-condition to

decide whether to continue or to terminate

1. While Loop
2. Post-Test Loop
3. Format Specifier
4. Do-While Loop

13. Do-While Loop; the condition is checked after the loop body is executed

1. Nested Loop
2. Pass-By-Value
3. Post-Test Loop
4. Pre-Test Loop

14. The part of the program where the variable is accessible

1. Anonymous Array
2. Scope of a Variable
3. Pass-By-Value
4. Method Abstraction

15. The method name and the parameter list

1. Method Signature
2. Method Abstraction
3. Sentinel Value
4. Indexed Variable

16. An automatic memory management feature that is collected by the Java Virtual Machine

1. Garbage Collection
2. Formal Parameter/Parameter
3. Output Redirection
4. Method Abstraction

17. The variable used in a method to stand for the value that is passed into the method by a caller

1. Pass-By-Value
2. Format Specifier
3. Formal Parameter/Parameter
4. Indexed Variable

18. A predefined class in the Java library, not a primitive type, a reference type, also an object

1. String
2. Index
3. Stub
4. Char type

19. The program takes the input from the file Input.TXT rather than having the user type the data from the keyword at runtime

1. Input Redirection
2. Scope of a Variable
3. Output Redirection
4. Format Specifier

20. A group of statements to perform a specific task

1. Static Method
2. Method
3. Iteration
4. Nested Loop

21. Checks the loop-continuation-condition first. If the condition

is true, its body is executed; if it is false, it terminates

1. Post-Test Loop
2. While Loop
3. Pre-Test Loop
4. For Loop

**Core Java, Quiz-6, Date: 20/09/2018**

1. Determine output:

**public class Test{**

**public static void main(String args[]){**

**int i;**

**for(i = 1; i < 6; i++){**

**if(i > 3) continue ;**

**}**

**System.out.println(i);**

**}**

**}**

A. 2

B. 3

C. 4

D. 5

**E. 6**

2. In java, ............ can only test for equality, whereas .......... can evaluate any type of the Boolean expression.

**A. switch, if**

B. if, switch

C. if, break

D. continue, if

3. What will be the output of the following program?

**public class Test{**

**public static void main(String args[]){**

**int i = 0, j = 5 ;**

**for( ; (i < 3) && (j++ < 10) ; i++ ){**

**System.out.print(" " + i + " " + j );**

**}**

**System.out.print(" " + i + " " + j );**

**}**

**}**

**A. 0 6 1 7 2 8 3 8**

B. 0 6 1 7 2 8 3 9

C. 0 6 1 5 2 5 3 5

D. Compilation Error

# 4. What is the output for the below code ?

**class A{**

**public A(){**

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

**}**

**public A(int i){**

**this();**

**System.out.println(i);**

**}**

**}**

**class B extends A{**

**public B(){**

**System.out.println("B");**

**}**

**public B(int i){**

**this();**

**System.out.println(i+3);**

**}**

**}**

**public class Test{**

**public static void main (String[] args){**

**new B(5);**

**}**

**}**

**A. A B 8**

B. A 5 B 8

C. A B 5

D. B 8 A 5

E. None of these

5. Determine output:

**public class Test{**

**public static void main(String args[]){**

**int i, j;**

**for(i=1, j=0;i<10;i++) j += i;**

**System.out.println(i);**

**}**

**}**

**A. 10**

B. 11

C. 9

D. 20

E. None of these

6. What will be the output?

**public class Test{**

**public static void main(String[] args){**

**int x=10, y=0;**

**if(x && y){**

**System.out.print("TRUE");**

**}**

**else{**

**System.out.print("FALSE");**

**}**

**}**

**}**

A. FALSE

B. TRUE

**C. Compilation Error**

D. Runtime Error

7. What will be the value of y after execution of switch statement?

**public class Test{**

**public static void main(String[] args){**

**int x = 3, y = 4;**

**switch(x + 3){**

**case 6: y = 0;**

**case 7: y = 1;**

**default: y += 1;**

**}**

**}**

**}**

A. 1

**B. 2**

C. 3

D. 4

E. 0

8. What is the printout of the following switch statement?

**char ch = 'a';**

**switch (ch){**

**case 'a':**

**case 'A': System.out.print(ch); break;**

**case 'b':**

**case 'B': System.out.print(ch); break;**

**case 'c':**

**case 'C': System.out.print(ch); break;**

**case 'd':**

**case 'D': System.out.print(ch);**

**}**

A. abcd

B. aa

**C. a**

D. ab

E. abc

9. How many times will the following code print "Welcome to Examveda"?

**int count = 0;**

**do {**

**System.out.println("Welcome to Examveda");**

**count++;**

**} while (count < 10);**

A. 8

B. 9

**C. 10**

D. 11

E. 0

10. Choose the correct statement in context of the following program code.

**public class Test{**

**public static void main(String[] args){**

**double sum = 0;**

**for(double d = 0; d < 10;){**

**d += 0.1;**

**sum += sum + d;**

**}**

**}**

**}**

A. The program has a compile error because the adjustment is missing in the for loop.

B. The program has a compile error because the control variable in the for loop cannot be of the double type.

C. The program runs in an infinite loop because d<10 would always be true.

**D. The program compiles and runs fine.**

11. Which of the following for loops will be an infinite loop?

A. for(; ;)

B. for(i=0 ; i<1; i--)

C. for(i=0; ; i++)

**D. All of the above**

12. What will be the result of the following code?

**public class Test{**

**static public void main(String args[]){ //line 2**

**int i, j;**

**for(i=0; i<3; i++){**

**for(j=1; j<4; j++){**

**i%=j;**

**System.out.println(j);**

**}**

**}**

**}**

**}**

A. 1 2 3 1

B. 1 2 3 2

**C. Repeatedly print 1 2 3 and cause infinite loop.**

D. Compilation fails because of line 2

E. None of these

13. What is the value of a[1] after the following code is executed?

**int[] a = {0, 2, 4, 1, 3};**

**for(int i = 0; i < a.length; i++)**

**a[i] = a[(a[i] + 3) % a.length];**

A. 0

**B. 1**

C. 2

D. 3

E. 4

14. What will be the result of compiling and runnig the following code:

**public class Test{**

**public static void main(String... args) throws Exception{**

**Integer i = 34;**

**int l = 34;**

**if(i.equals(l)){**

**System.out.println("true");**

**}else{**

**System.out.println("false");**

**}**

**}**

**}**

**A. true**

B. false

C. Compiler error

D. None of these

15. What all gets printed when the following program is compiled and run.

**public class Test{**

**public static void main(String args[]){**

**int i, j=1;**

**i = (j>1)?2:1;**

**switch(i){**

**case 0: System.out.println(0); break;**

**case 1: System.out.println(1);**

**case 2: System.out.println(2); break;**

**case 3: System.out.println(3); break;**

**}**

**}**

**}**

A. 0

B. 1

C. 2

D. 3

**E. 1 2**

16. What all gets printed when the following program is compiled and run?

**public class Test{**

**public static void main(String args[]){**

**int i=0, j=2;**

**do{**

**i=++i;**

**j--;**

**}while(j>0);**

**System.out.println(i);**

**}**

**}**

A. 0

B. 1

**C. 2**

D. The program does not compile because of statement "i=++i;"

E. None of these

17. What will be the output?

**public class Test{**

**public static void main(String args[]){**

**int i = 1;**

**do{**

**i--;**

**}while(i > 2);**

**System.out.println(i);**

**}**

**}**

A. 1

B. 2

C. -1

**D. 0**

E. None of these

18. Which option, inserted at line 4, produces the output 12?

**1. public class Test{**

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

**3. int x = 0;**

**4. // insert code here**

**5. do{ } while(x++ < y);**

**6. System.out.println(x);**

**7. }**

**8. }**

A. int y = x;

B. int y = 10;

**C. int y = 11;**

D. int y = 12;

E. None of the above will allow compilation to succeed.

19. What will be the result?

**1. int i = 10;**

**2. while(i++ <= 10){**

**3. i++;**

**4. }**

**5. System.out.print(i);**

A. 10

B. 11

C. 12

**D. 13**

E. Line 5 will be never reached.

**20. What is output of the program?**

**class Test{**

**public void display(int x, double y){**

**System.out.println(x+y);**

**}**

**public double display(int p, double q){**

**return (p+q);**

**}**

**public static void main(String args[]){**

**Test test = new Test();**

**test.display(4, 5.0);**

**System.out.println(test.display(4, 5.0));**

**}**

**}**

A. 9.0 9.0

B. 9 9

**C. Compilation Error**

D. None of these

# 21.what is the result of the following piece of code:

**public class Person{**

**public void talk(){**

**System.out.print("I am a Person");**

**}**

**}**

**public class Student extends Person{**

**public void talk(){**

**System.out.print("I am a Student");**

**}**

**}**

**public class Test{**

**public static void main(String args[]){**

**Person p = new Student();**

**p.talk();**

**}**

**}**

A. I am a Person

**B. I am a Student**

C. I am a Person I am a Student

D. I am a Student I am a Person

# **22.** What will be the output?

**interface A{**

**public void method1();**

**}**

**class One implements A{**

**public void method1(){**

**System.out.println("Class One method1");**

**}**

**}**

**class Two extends One{**

**public void method1(){**

**System.out.println("Class Two method1");**

**}**

**}**

**public class Test extends Two{**

**public static void main(String[] args){**

**A a = new Two();**

**a.method1();**

**}**

**}**

A. Class One method1

**B. Class Two method1**

C. Nothing will be printed

D. Compilation Error

23. What will be the output of the following program code?

**class Rectangle{**

**public int area(int length, int width){**

**return length\*width;**

**}**

**}**

**class Square extends Rectangle{**

**public int area(long length, long width){**

**return (int) Math.pow(length, 2);**

**}**

**}**

**public class Test{**

**public static void main(String args[]){**

**Square r = new Square();**

**System.out.println(r.area(5 , 4));**

**}**

**}**

A. Will not compile.

**B. Will compile and run printing out 20**

C. Runtime error

D. Will compile and run printing out 25

# 24. What will be printed after executing following program code?

**class Base{**

**int value = 0;**

**Base(){**

**addValue();**

**}**

**void addValue(){**

**value += 10;**

**}**

**int getValue(){**

**return value;**

**}**

**}**

**class Derived extends Base{**

**Derived(){**

**addValue();**

**}**

**void addValue(){**

**value += 20;**

**}**

**}**

**public class Test{**

**public static void main(String[] args){**

**Base b = new Derived();**

**System.out.println(b.getValue());**

**}**

**}**

A. 10

B. 20

C. 30

**D. 40**

E. None of these

# 25. What will be the output?

**class A{**

**static void method(){**

**System.out.println("Class A method");**

**}**

**}**

**class B extends A{**

**static void method(){**

**System.out.println("Class B method");**

**}**

**}**

**public class Test{**

**public static void main(String args[]){**

**A a = new B();**

**a.method();**

**}**

**}**

**A. Class A method**

B. Class B method

C. Compilation Error

D. Runtime Error

E. None of these

# 26. \_\_\_\_\_\_\_\_\_\_\_\_ method cannot be overridden.

A. super

B. static

**C. final**

D. private

E. None of these

# 27. What will be the output?

**class A{**

**int i = 10;**

**public void printValue(){**

**System.out.print("Value-A");**

**}**

**}**

**class B extends A{**

**int i = 12;**

**public void printValue(){**

**System.out.print("Value-B");**

**}**

**}**

**public class Test{**

**public static void main(String args[]){**

**A a = new B();**

**a.printValue();**

**System.out.print(a.i);**

**}**

**}**

A. Value-B 11

**B. Value-B 10**

C. Value-A 10

D. Value-A 11

E. None of these

# 28. What is the output for the below code?

**public class Test{**

**public static void printValue(int i, int j, int k){**

**System.out.println("int");**

**}**

**public static void printValue(byte...b){**

**System.out.println("long");**

**}**

**public static void main(String... args){**

**byte b = 9;**

**printValue(b,b,b);**

**}**

**}**

A. long

**B. int**

C. Compilation fails

D. Compilation clean but throws RuntimeException

E. None of these

# 29. Determine output:

**class A{**

**public void printValue(){**

**System.out.println("Value-A");**

**}**

**}**

**class B extends A{**

**public void printNameB(){**

**System.out.println("Name-B");**

**}**

**}**

**class C extends A{**

**public void printNameC(){**

**System.out.println("Name-C");**

**}**

**}**

**1. public class Test{**

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

**3. B b = new B();**

**4. C c = new C();**

**5. newPrint(b);**

**6. newPrint(c);**

**7. }**

**8. public static void newPrint(A a){**

**9. a.printValue();**

**10. }**

**11. }**

A. Value-A Name-B

**B. Value-A Value-A**

C. Value-A Name-C

D. Name-B Name-C

E. None of these

# 30. What is the output for the below code ?

**class A{**

**private void printName(){**

**System.out.println("Value-A");**

**}**

**}**

**class B extends A{**

**public void printName(){**

**System.out.println("Name-B");**

**}**

**}**

**public class Test{**

**public static void main (String[] args){**

**B b = new B();**

**b.printName();**

**}**

**}**

A. Value-A

**B. Name-B**

C. Value-A Name-B

D. Compilation fails - private methods can't be override

E. None of these

**Core Java, Quiz-8, Date: 24/09/2018**

**1. Which is true?**

A. "X extends Y" is correct if and only if X is a class and Y is an interface

B. "X extends Y" is correct if and only if X is an interface and Y is a class

C. "X extends Y" is correct if X and Y are either both classes or both interfaces

D. "X extends Y" is correct for all combinations of X and Y being classes and/or interfaces

# 2. Which of the following is true?

# 1. A class can extend more than one class. 2. A class can extend only one class but many interfaces. 3. An interface can extend many interfaces. 4. An interface can implement many interfaces. 5. A class can extend one class and implement many interfaces.

A. 1 and 2

B. 2 and 4

C. 3 and 5

D. 3 and 4

E. 2 and 5

# 3. What is the result of compiling and running the following code?

**class Base{**

**public Base(){**

**System.out.print("Base");**

**}**

**}**

**public class Derived extends Base{**

**public Derived(){**

**this("IDB");**

**System.out.print("Derived");**

**}**

**public Derived(String s){**

**System.out.print(s);**

**}**

**public static void main(String[] args){**

**new Derived();**

**}**

**}**

A. IDBDerived

B. IDBBaseDerived

C. Base**IDB**Derived

D. IDBDerivedBase

E. Compilation Error

# 4. What is the output of the following program code?

**abstract class** C1**{**

**public** C1()**{**

System.out.print(1);

**}**

**}**

**class** C2 **extends** C1**{**

**public** C2()**{**

System.out.print(2);

**}**

**}**

**class** C3 **extends** C2**{**

**public** C3()**{**

System.out.println(3);

**}**

**}**

**public class** Test**{**

**public static void** main(String[] a)**{**

**new** C3();

**}**

**}**

A. 12

B. 23

C. 123

D. 321

**Answer: Option C**

# 5. The concept of multiple inheritance is implemented in Java by I.   Extending two or more classes. II.  Extending one class and implementing one or more interfaces. III. Implementing two or more interfaces.

A. Only (II)

B. (I) and (II)

C. (II) and (III)

D. Only (I)

E. Only (III)

# 6. What will be the output?

**interface** A**{**

**public void** method1();

**}**

**class** One **implements** A**{**

**public void** method1()**{**

System.out.println("Class One method1");

**}**

**}**

**class** Two **extends** One**{**

**public void** method1()**{**

System.out.println("Class Two method1");

**}**

**}**

**public class** Test **extends** Two**{**

**public static void** main(String[] args)**{**

A a = **new** Two();

a.method1();

**}**

**}**

A. Compilation Error

B. Class One method1

C. Class Two method1

D. Throws a NoSuchMethodException at runtime.

E. None of these

# 7. What is the result of compiling and running this program?

**class** Mammal**{**

**void** eat(Mammal m)**{**

System.out.println("Mammal eats food");

**}**

**}**

**class** Cattle **extends** Mammal**{**

**void** eat(Cattle c)**{**

System.out.println("Cattle eats hay");

**}**

**}**

**class** Horse **extends** Cattle**{**

**void** eat(Horse h)**{**

System.out.println("Horse eats hay");

**}**

**}**

**public class** Test**{**

**public static void** main(String[] args)**{**

Mammal h = **new** Horse();

Cattle c = **new** Horse();

c.eat(h);

**}**

**}**

A. prints "Mammal eats food"

B. prints "Cattle eats hay"

C. prints "Horse eats hay"

D. Class cast Exception at runtime.

E. None of these

8.Determine output:

**class** A**{**

**public void** method1()**{**

System.out.print("Class A method1");

**}**

**}**

**class** B **extends** A**{**

**public void** method2()**{**

System.out.print("Class B method2");

**}**

**}**

**class** C **extends** B**{**

**public void** method2()**{**

System.out.print("Class C method2");

**}**

**public void** method3()**{**

System.out.print("Class C method3");

**}**

**}**

**public class** Test**{**

**public static void** main(String args[])**{**

A a = **new** A();

C c = **new** C();

c.method2();

a = c;

a.method3();

**}**

**}**

A. Class B method2 Class C method3

B. Class C method2 Class C method3

C. Compilation Error

D. Runtime exception

E. None of these

# 9. What will be printed after executing following program code?

**class** Base**{**

**int** value = 0;

Base()**{**

addValue();

**}**

**void** addValue()**{**

value += 10;

**}**

**int** getValue()**{**

**return** value;

**}**

**}**

**class** Derived **extends** Base**{**

Derived()**{**

addValue();

**}**

**void** addValue()**{**

value += 20;

**}**

**}**

**public class** Test**{**

**public static void** main(String[] args)**{**

Base b = **new** Derived();

System.out.println(b.getValue());

**}**

**}**

A. 30

B. 10

C. 40

D. 20

E. None of these

# 10. What will be the output?

**class** Parent**{**

**public void** method()**{**

System.out.println("Hi i am parent");

**}**

**}**

**public class** Child **extends** Parent**{**

**protected void** method()**{**

System.out.println("Hi i am Child");

**}**

**public static void** main(String args[])**{**

Child child = **new** Child();

child.method();

**}**

**}**

A. Compiles successfully and print

B. Compiles successfully and print

C. Compile time error

D. Run Time error

E. None of This

# 11. What will be the output?

**class** One**{**

**final int** a = 15;

**}**

**class** Two **extends** One**{**

**final int** a = 20;

**}**

**public class** Test **extends** Two**{**

**final int** a = 30;

**public static void** main(String args[])**{**

Test t = **new** One();

System.out.print(t.a);

**}**

**}**

A. 15

B. 20

C. 30

D. Compiler Error

E. None of these

# 12. What will be the output?

**class** A**{**

**int** i = 10;

**public void** printValue()**{**

System.out.print("Value-A");

**}**

**}**

**class** B **extends** A**{**

**int** i = 12;

**public void** printValue()**{**

System.out.print("Value-B");

**}**

**}**

**public class** Test**{**

**public static void** main(String args[])**{**

A a = **new** B();

a.printValue();

System.out.print(a.i);

**}**

**}**

A. Value-B 11

B. Value-B 10

C. Value-A 10

D. Value-A 11

E. None of these

# 13. What will be the result after compiling this code?

**class** SuperClass**{**

**public int** doIt(String str, Integer... data)**throws** Exception**{**

String signature = "(String, Integer[])";

System.out.println(str + " " + signature);

**return** 1;

**}**

**}**

**public class** Test **extends** SuperClass**{**

**public int** doIt(String str, Integer... data)**{**

String signature = "(String, Integer[])";

System.out.println("Overridden: " + str + " " +signature);

**return** 0;

**}**

**public static void** main(String... args)**{**

SuperClass sb = **new** Test();

sb.doIt("hello", 3);

**}**

**}**

A. Overridden: hello (String, Integer[])

B. hello (String, Integer[])

C. Compilation fails

D. None of these

**14. class** A**{**

A(String s)**{}**

A()**{}**

**}**

**class** B **extends** A**{**

B()**{}**

B(String s)**{**

super(s);

**}**

void test()**{**

**// insert code here**

**}**

**}**

# Which of the below code can be insert at line 7 to make clean compilation ?

A. A a = new B();

B. A a = new B(5);

C. A a = new A(String s);

D. All of the above

E. None of these

# 15. Determine output:

**class** A**{**

**public void** printValue()**{**

System.out.println("Value-A");

**}**

**}**

**class** B **extends** A**{**

**public void** printNameB()**{**

System.out.println("Name-B");

**}**

**}**

**class** C **extends** A**{**

**public void** printNameC()**{**

System.out.println("Name-C");

**}**

**}**

1. **public class** Test**{**

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

3. B b = **new** B();

4. C c = **new** C();

5. newPrint(b);

6. newPrint(c);

7. **}**

8. **public static void** newPrint(A a)**{**

9. a.printValue();

10. **}**

11. **}**

A. Value-A Name-B

B. Value-A Value-A

C. Value-A Name-C

D. Name-B Name-C

E. None of these

# 16. Determine output:

**class** A**{**

**public void** printName()**{**

System.out.println("Name-A");

**}**

**}**

**class** B **extends** A**{**

**public void** printName()**{**

System.out.println("Name-B");

**}**

**}**

**class** C **extends** A**{**

**public void** printName()**{**

System.out.println("Name-C");

**}**

**}**

1. **public class** Test**{**

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

3. B b = **new** B();

4. C c = **new** C();

5. b = c;

6. newPrint(b);

7. **}**

8. **public static void** newPrint(A a)**{**

9. a.printName();

10. **}**

11. **}**

A. Name B

B. Name C

C. Compilation fails due to an error on lines 5

D. Compilation fails due to an error on lines 9

E. None of these

# 17. What is the output for the below code ?

**class** A**{**

**private void** printName()**{**

System.out.println("Value-A");

**}**

**}**

**class** B **extends** A**{**

**public void** printName()**{**

System.out.println("Name-B");

**}**

**}**

**public class** Test**{**

**public static void** main (String[] args)**{**

B b = **new** B();

b.printName();

**}**

**}**

A. Value-A

B. Name-B

C. Value-A Name-B

D. Compilation fails - private methods can't be override

E. None of these

# 18. What will be the result of compiling and running the given code?

**class** A**{**

**int** b=10;

**private** A()**{**

**this**.b=7;

**}**

**int** f()**{**

return b;

**}**

**}**

**class** B **extends** A**{**

**int** b;

**}**

**public class** Test**{**

**public static void** main(String[] args)**{**

A a = **new** B();

System.out.println(a.f());

**}**

**}**

A. Compilation Fails

B. Prints 0

C. Prints 10

D. Prints 7

E. None of these

# 19. What will be the result of compiling and executing the following program code?

**class** Vehicle**{**

**public void** printSound()**{**

System.out.print("vehicle");

**}**

**}**

**class** Car **extends** Vehicle**{**

**public void** printSound()**{**

System.out.print("car");

**}**

**}**

**class** Bike **extends** Vehicle**{**

**public void** printSound()**{**

System.out.print("bike");

**}**

**}**

**public class** Test**{**

**public static void** main(String[] args)**{**

Vehicle v = **new** Car();

Bike b = (Bike) v;

v.printSound();

b.printSound();

**}**

**}**

A. Compilation fails.

B. ClassCastException exception is thrown at runtime.

C. "vehiclecar" is printed.

D. "vehiclebike" is printed.

E. "carcar" is printed.

# 20. Determine output:

**class** Small**{**

**public** Small()**{**

System.out.print("a ");

**}**

**}**

**class** Small2 **extends** Small**{**

**public** Small2()**{**

System.out.print("b ");

**}**

**}**

**class** Small3 **extends** Small2**{**

**public** Small3()**{**

System.out.print("c ");

**}**

**}**

**public class** Test**{**

**public static void** main(String args[])**{**

**new** Small3();

**}**

**}**

A. a

B. c

C. a b c

D. c b a

E. The code runs without output.

**Core Java, Quiz-9, Date: 25/09/2018**

**1. The class at the top of exception class hierarchy is .................**

A. ArithmeticException

B. Throwable

C. Object

D. Exception

**2. In which of the following package Exception class exist?**

A. java.util

B. java.file

C. java.io

D. java.lang

E. java.net

**3. Exception generated in try block is caught in ........... block.**

A. catch

B. throw

C. throws

D. finally

**4. Which keyword is used to explicitly throw an exception?**

A. try

B. throwing

C. catch

D. throw

**5. Which exception is thrown when divide by zero statement executes?**

A. NumberFormatException

B. ArithmeticException

C. NullPointerException

D. None of these

**6. Which keyword is used to specify the exception thrown by method?**

A. catch

B. throws

C. finally

D. throw

**7.**

**public class Test{**

**public static void main(String args[]){**

**try{**

**int a = Integer.parseInt("four");**

**}**

**}**

**}**

**Which exception could be handled by the catch block for above?**

A. IllegalStateException

B. NumberFormatException

C. ClassCastException

D. ArrayIndexOutOfBoundsException

E. None of these

**8.**

**public class** Test**{**

**public static void** main(String args[])**{**

**try{**

**int** i;

**return**;

**}**

**catch**(Exception e)**{**

System.out.print("inCatchBlock");

**}**

**finally{**

System.out.println("inFinallyBlock");

**}**

**}**

**}**

A. inCatchBlock

B. inCatchBlock inFinallyBlock

C. inFinallyBlock

D. The program will return without printing anything

**9.**

**public class** Test**{**

**public static void** main(String args[]){

**try{**

String arr[] = **new** String[10];

arr = null;

arr[0] = "one";

System.out.print(arr[0]);

**}**catch(Exception ex)**{**

System.out.print("exception");

**}**catch(NullPointerException nex)**{**

System.out.print("null pointer exception");

**}**

**}**

**}**

A. "one" is printed.

B. "exception" is printed.

C. "null pointer exception" is printed.

D. Compilation fails saying NullPointerException has already been caught.

E. None of these

10.

**try**{

File f = **new** File("a.txt");

**}catch**(Exception e)**{**

**}catch**(IOException io)**{**

**}**

Is this code create new file name a.txt ?

A. true

B. false

C. Compilation Error

D. None of these

**11. Which of the following blocks execute compulsorily whether exception is caught or not.**

A. finally

B. catch

C. throws

D. throw

**12. What happen in case of multiple catch blocks?**

A. Either super or subclass can be caught first.

B. The superclass exception must be caught first.

C. The superclass exception cannot caught first.

D. None of these

**13. Which exception is thrown when an array element is accessed beyond the array size?**

A. ArrayElementOutOfBounds

B. ArrayIndexOutOfBoundsException

C. ArrayIndexOutOfBounds

D. None of these

# 14. What is the output of the following program code?

**public class** Test**{**

**public static void** main(String args[])**{**

**try{**

**int** i;

**return**;

**}**

**catch**(Exception e)**{**

System.out.print("inCatchBlock");

**}**

**finally{**

System.out.println("inFinallyBlock");

**}**

**}**

**}**

A. inCatchBlock

B. inCatchBlock inFinallyBlock

C. inFinallyBlock

D. The program will return without printing anything

# 15. Given the following piece of code:

**class** SalaryCalculationException **extends** Exception**{}**

**class** Person**{**

**public void** calculateSalary() **throws** SalaryCalculationException**{**

//...

**throw new** SalaryCalculationException();

//...

**}**

**}**

**class** Company**{**

**public void** paySalaries()**{**

**new** Person().calculateSalary();

**}**

**}**

Which of the following statements is correct?  
1. This code will compile without any problems.  
2. This code will compile if in method paySalaries() we return a boolean in stead of void.  
3. This code will compile if we add a try-catch block in paySalaries().  
4. This code will compile if we add throws SalaryCalculationException in the signature of method paySalaries().

A. 1 and 4

B. 2 and 3

C. 2 and 4

D. 3 and 4

E. 1 and 2

# 16. What will be the output of the following piece of code:

**class** Person**{**

**public void** talk() **{}**

**}**

**public class** Test**{**

**public static void** main(String args[])**{**

Person p = null;

**try{**

p.talk();

**}**

**catch**(NullPointerException e)**{**

System.out.print("There is a NullPointerException. ");

**}**

**catch**(Exception e)**{**

System.out.print("There is an Exception. ");

**}**

System.out.print("Everything went fine. ");

**}**

**}**

A. There is a NullPointerException. Everything went fine.

B. There is a NullPointerException.

C. There is a NullPointerException. There is an Exception.

D. This code will not compile, because in Java there are no pointers.

# 17. Determine output of the following program code?

**public class** Test**{**

**public static void** main(String args[])**{**

**int** i;

**try{**

i = calculate();

System.out.println(i);

**}**catch(Exception e)**{**

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

**}**

**}**

**static int** calculate()**{**

**return** (7/2);

**}**

**}**

A. 3

B. 3.5

C. Error occured

D. Compilation Error

E. None of these

**18.**

**public class** Test**{**

**public static void** main(String args[])**{**

**try{**

**int** a = Integer.parseInt("four");

**}**

**}**

**}**

Which exception could be handled by the catch block for above?

A. IllegalStateException

B. NumberFormatException

C. ClassCastException

D. ArrayIndexOutOfBoundsException

E. None of these

**19. What will be the output? l be the output?**

**class** MyClass**{**

**public String** test()**{**

**try{**

System.out.print("One");

**return** "";

}

**finally{**

System.out.print("Two");

**}**

**}**

**}**

**public class** Test**{**

**public static void** main(String args[])**{**

MyClass m = **new** MyClass();

m.test();

**}**

**}**

A. One

B. Two

C. One Two

D. Compilation Error

E. None of these

**20. What will be the result after compiling this code?**

**class** SuperClass**{**

**public int** doIt(String str, Integer... data)**throws** Exception**{**

String signature = "(String, Integer[])";

System.out.println(str + " " + signature);

**return** 1;

**}**

**}**

**public class** Test **extends** SuperClass**{**

**public int** doIt(String str, Integer... data)**{**

String signature = "(String, Integer[])";

System.out.println("Overridden: " + str + " " +signature);

**return** 0;

**}**

**public static void** main(String... args)**{**

SuperClass sb = **new** Test();

sb.doIt("hello", 3);

**}**

**}**

A. Overridden: hello (String, Integer[])

B. hello (String, Integer[])

C. Compilation fails

D. None of these

**21. What is the output for the below code ?**

**import** java.io.FileNotFoundException;

**class** A**{**

**public void** printName() **throws** FileNotFoundException**{**

System.out.println("Value-A");

**}**

**}**

**class** B **extends** A**{**

**public void** printName() **throws** NullPointerException**{**

System.out.println("Name-B");

**}**

**}**

**public class** Test**{**

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

A a = **new** B();

a.printName();

**}**

**}**

A. Value-A

B. Compilation fails-Exception NullPointerException is not compatible with throws clause in A.printName()

C. Name-B

D. Compilation succeed but no output

E. None of these

**22.**

**try**{

File f = **new** File("a.txt");

**}catch**(Exception e)**{**

**}catch**(IOException io)**{**

**}**

Is this code create new file name a.txt ?

A. true

B. false

C. Compilation Error

D. None of these

**23. What is the output for the below code ?**

**import** java.io.FileNotFoundException;

**class** A**{**

**public void** printName() **throws** FileNotFoundException**{**

System.out.println("Value-A");

**}**

**}**

**class** B **extends** A**{**

**public void** printName() **throws** NullPointerException**{**

System.out.println("Name-B");

**}**

**}**

**public class** Test**{**

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

A a = **new** B();

a.printName();

**}**

**}**

**A.** Value-A

B. Compilation fails-Exception NullPointerException is not compatible with throws clause in A.printName()

C. Name-B

D. Compilation succeed but no output

E. None of these

**24. What will be the result of executing the following code?**

**public class** Test**{**

**public void** divide(**int** a, **int** b)**{**

**try{**

**int** c = a / b;

**}catch**(Exception e)**{**

System.out.print("Exception ");

**}finally{**

System.out.println("Finally");

**}**

**public static void** main(String args[])**{**

Test t = **new** Test();

t.divide(0,3);

**}**

**}**

**A.** Prints out: Exception

B. Prints out: Exception Finally

C. Compile with error

D. Prints out: Finally

E. None of these

**25. Which of the below statement is/are true about Error?**  
A. An Error is a subclass of Throwable.  
B. An Error is a subclass of Exception.  
C. Error indicates serious problems that a reasonable application should not try to catch.  
D. An Error is a subclass of IOException.

A. A and D

B. A and B

C. A and C

D. B and C

E. B and D

**26. Predict the output:**

**public class** Test**{**

**public static void** main(String args[]){

**try{**

String arr[] = **new** String[10];

arr = null;

arr[0] = "one";

System.out.print(arr[0]);

**}**catch(Exception ex)**{**

System.out.print("exception");

**}**catch(NullPointerException nex)**{**

System.out.print("null pointer exception");

**}**

**}**

**}**

**A**. "one" is printed.

B. "exception" is printed.

C. "null pointer exception" is printed.

D. Compilation fails saying NullPointerException has already been caught.

E. None of these

# 27. Given the code. What is the result when this program is executed?

**public class** Test**{**

**static int** x[];

**static{**

x[0] = 1;

**}**

**public static void** main(String args[])**{**

**}**

**}**

A. ArrayIndexOutOfBoundsException is thrown

B. ExceptionInInitializerError is thrown

C. IllegalStateException is thrown

D. StackOverflowException is thrown

E. None of these

**28. What will be the result if NullPointerException occurs at line 2?**

1. **try{**

2. **//some code goes here**

3. **}**

4. **catch**(NullPointerException ne)**{**

5. System.out.print("1 ");

6. **}**

7. **catch**(RuntimeException re)**{**

8. System.out.print("2 ");

9. **}**

10. **finally{**

11. System.out.print("3");

12. **}**

A. 1

B. 3

C. 2 3

D. 1 3

E. 1 2 3

**29. What will be the result after the class Test execution?**

**class** A**{**

**public void** doA()**{**

B b = **new** B();

b.dobB();

System.out.print("doA");

**}**

**}**

**class** B{

**public void** dobB()**{**

C c = **new** C();

c.doC();

System.out.print("doB");

**}**

**}**

**class** C**{**

**public void** doC()**{**

**if**(true)

**throw new** NullPointerException();

System.out.print("doC");

**}**

**}**

**public class** Test**{**

**public static void** main(String args[])**{**

**try{**

A a = **new** A();

a.doA();

**}catch**(Exception ex){

System.out.print("error");

**}**

**}**

**}**

**A**. "doCdoBdoA" is printed

B. "doAdoBdoC" is printed

C. "doBdoAerror" is printed

D. "error" is printed

E. nothing is printed