**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 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)

**3. abstract is a---**

a. keyword

b. statement

c. int type

d. final keyword

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

1. floating-point number

2. concatenate strings

3. Floating point

4. widening (of types)

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

1. preincrement

2. logic error

3. identifier

4. compiler

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

1. dangling else ambiguity

2. selection statement

3. floating-point number

4. short circuit operator

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

1. postdecrement

2. preincrement

3. statement

4. postincrement

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

1. Conditional Expression ( ? :)

2. Integrated development environment

3. nextDouble

4. Augmented assignment operators

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

1. assignment statement

2. widening (of types)

3. Boolean Expression

4. assignment operator

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

1. Bytecode

2. bytecode verifier

3. widening (of types)

4. logic error

**11. 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

**12. do, else, and break**

1. keyword

2. keywords

3. operands

4. operators

**13. 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

**14. 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

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

1. syntax error

2. Bytecode

3. source code/program

4. interpreter

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

1. conditional operator

2. concatenate strings

3. primitive data type

4. fall-through behavior

**17. denotes names**

1. final

2. final keyword

3. Bit

4. Block

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

1. runtime error

2. logic error

3. input error

4. postincrement

**19. 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

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

1. assignment statement

2. operator precedence

3. selection statement

4. Application Program Interface ( API)

**21. 8 bits to 1 byte**

1. illegal identifier

2. operators

3. octa integer

4. Byte

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

1. Relational Operators (Boolean)

2. selection statement

3. Conditional Expression ( ? :)

4. conditional operator

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

1. dangling else ambiguity

2. selection statement

3. short circuit operator

4. primitive data type

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

1. constant

2. Literal

3. Bytecode

4. int type

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

1. literal

2. util

3. Assembler

4. casting

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

1. Boolean Expression

2. primitive data type

3. operator precedence

4. selection statement

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

1. operator associativity

2. Conditional Expression ( ? :)

3. interpreter

4. assignment statement

**28. scientific notation**

1. postincrement

2. Floating point

3. boolean operators

4. final keyword

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

1. bytecode verifier

2. source code/program

3. concatenate strings

4. interpreter

**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

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

1. concatenate strings

2. scope of a variable

3. operator precedence

4. Conditional Expression ( ? :)

**32. Variable**

1. preprocessor

2. identifier

3. directive

4. String

**33. 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)

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

1. dot pitch

2. operands

3. variable

4. overflow

**35. 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

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

1. wildcard import

2. overflow

3. input error

4. double type

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

1. operands

2. compiler

3. Literal

4. overflow

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

1. constant

2. compiler

3. input error

4. double type

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

1. Block

2. IPO

3. keyword

4. final

**40. 4thQtrSales**

1. identifier

2. floating point/pi

3. illegal identifier

4. preprocessor

41. Using a method without knowing how it is implemented

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

42. A loop statement that executes infinitely

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

43. 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

44. Keywords such as public, static, and final

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

45. Separating the use of a method from its implementation

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

46. Only can be invoked from a specific string instance

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

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

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

48. When there are 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

49. A simple, incomplete version of a method

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

50. 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

51. Immediately ends the innermost loop

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

52. 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

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

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

54. Only ends a current iteration

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

55. 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

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

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

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

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

58. 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

59. 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

60. The method name and the parameter list

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

61. 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

62. 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

63. A group of statements to perform a specific task

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

64. 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

63. 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**

64. 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

# 65. 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

66. 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

67. 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

68. 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

69. 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

70. 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**

71. 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

72. 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**

73. 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

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

A. 10

B. 11

C. 12

**D. 13**

E. Line 5 will be never reached.

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

A. super

B. static

**C. final**

D. private

E. None of these

**76. 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

# 77. 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

# 78. 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)

# 79. 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

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

A. ArithmeticException

B. Throwable

C. Object

D. Exception

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

A. java.util

B. java.file

C. java.io

D. java.lang

E. java.net

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

A. catch

B. throw

C. throws

D. finally

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

A. try

B. throwing

C. catch

D. throw

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

A. NumberFormatException

B. ArithmeticException

C. NullPointerException

D. None of these

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

A. catch

B. throws

C. finally

D. throw

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

A. finally

B. catch

C. throws

D. throw

**87. 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

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

A. ArrayElementOutOfBounds

B. ArrayIndexOutOfBoundsException

C. ArrayIndexOutOfBounds

D. None of these

# 89. 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