1. The newly introduced Streams API is available in which package of java 8:
   1. java.io.streams
   2. java.io.stream
   3. java.util.streams
   4. java.util.stream
2. Lambda expression represents an instance of \_\_\_\_\_\_\_\_\_\_\_\_\_.
   1. Abstract Method
   2. Concrete Class
   3. Functional interface
   4. Abstract class
3. Stream operations in java 8 can be divided into
   1. Terminal types
   2. Intermediate types
   3. All
   4. None
4. Identify the method inside Supplier Interface.
   1. void accept(T t);
   2. T get();
   3. boolean test(T t);
   4. R apply(T t);
5. Consider the below lambda expression, which call println method of System.out:

Consumer<String> consumer = (String str)-> System.out.println(str);

Identify the expression that can rewrite the above expression with method reference:

* 1. Consumer<String> consumer = System.out **::** println;
  2. Consumer<String> consumer = System**::** println;
  3. Consumer<String> consumer = System::out**::**println;
  4. Consumer<String> consumer = System.out **::** println();

1. Identify the argument the filter method of the Stream interface accepts.
   1. java.util.function.Function
   2. java.util.function.Predicate
   3. java.util.stream.Function
   4. java.util.stream.Predicate
2. The \_\_\_\_\_\_\_\_\_\_\_\_operation takes a sequence of input elements and combines them into a single summary result by repeated application of a combining operation.
   1. Sum
   2. Reduce
   3. Filter
   4. Map
3. Identify the interface that supports addition or removal of elements from either end of the data structure.
   1. Queue
   2. ArrayBlockingQueue
   3. ConcurrentLinkedQueue
   4. Deque
4. Identify the method of ArrayBlockingQueue that inserts the specified element at the tail of the queue if it is possible to do without exceeding the queue's capacity, returning true upon success and false if this queue is full.
   1. add(E e)
   2. offer(E e)
   3. put( E e)
   4. insert(E e)
5. Identify the method of ConcurrentMap that associates the given value with the key if the specified key is not already associated with a value.
   1. putIfAbsent(K key, V value)
   2. put (K key, V value)
   3. addIfAbsent(K key, V value)
   4. computeIfAbsent(K key, V value)
6. Identify the implementation of the task that cannot return a result and throw an exception
   1. Callable
   2. Runnable
   3. Executor
   4. ExecutorService
7. Given:

public class Letters extends Thread {

private String name;

public Letters(String name) { this.name = name; }

public void write() {

System.out.print(name);

System.out.print(name);}

public static void main(String[] args) {

new Letters("X").start();

new Letters("Y").start();

} }

We want to guarantee that the output can be either XXYY or YYXX, but never XYXY or any other combination. Which of the following method definitions could be added to the Letters class to make this guarantee? (Choose all that apply.)

* 1. public void run() { write(); }
  2. public synchronized void run() { write(); }
  3. public static synchronized void run() { write(); }
  4. public void run() { synchronized(this) { write(); } }
  5. public void run() { synchronized(Letters.class) { write(); } }

1. Which is true? (Choose all that apply.)
   1. "X extends Y" is correct if and only if X is a class and Y is an interface.
   2. "X extends Y" is correct if and only if X is an interface and Y is a class.
   3. "X extends Y" is correct if X and Y are either either classes or interfaces.
   4. "X extends Y" is correct for all combinations of X and Y being classes and/or interfaces.
2. Which are valid declarations? (Choose all that apply.)
   1. int $x;
   2. int 123;
   3. int \_123;
   4. int #dim;
   5. int %percent;
   6. int \*divide;
   7. int central\_sales\_region\_Summer\_2005\_gross\_sales;
3. Given:

1. enum Animals {

2. DOG("woof"), CAT("meow"), FISH("burble");

3. String sound;

4. Animals(String s) { sound = s; }

5. }

6. class TestEnum {

7. static Animals a;

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

9. System.out.println(a.DOG.sound + " " + a.FISH.sound);

10. }

11. }

What is the result?

* 1. woof burble
  2. Multiple compilation errors
  3. Compilation fails due to an error on line 2
  4. Compilation fails due to an error on line 3
  5. Compilation fails due to an error on line 4
  6. Compilation fails due to an error on line 9

1. Given the following,

1. class X { void do1() { } }

2. class Y extends X { void do2() { } }

3.

4. class Chrome {

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

6. X x1 = new X();

7. X x2 = new Y();

8. Y y1 = new Y();

9. // insert code here

10. }

11. }

Which, inserted at line 9, will compile? (Choose all that apply.)

* 1. x2.do2();
  2. (Y)x2.do2();
  3. ((Y)x2).do2();
  4. None of the above statements will compile.

1. Which statement(s) are true? (Choose all that apply.)
   1. Cohesion is the OO principle most closely associated with hiding implementation details.
   2. Cohesion is the OO principle most closely associated with making sure that classes know about other classes only through their APIs.
   3. Cohesion is the OO principle most closely associated with making sure that a class is designed with a single, well-focused purpose.
   4. Cohesion is the OO principle most closely associated with allowing a single object to be seen as having many types.
2. Given:

1. class Pro {

2. Pro debug() { return this; }

3. }

4. class JP extends Pro {

5. // insert code here

6. }

1. Which, inserted at line 5, will compile? (Choose all that apply.)
   1. Pro debug() { return this; }
   2. JP debug() { return this; }
   3. Object debug() { return this; }
   4. int debug() { return 1; }
   5. int debug(int x) { return 1; }
   6. Object debug(int x) { return this; }
2. Given:

class Scoop {

static int thrower() throws Exception { return 42; }

public static void main(String [] args) {

try {

int x = thrower();

} catch (Exception e) {

x++;

} finally {

System.out.println("x = " + ++x);

} } }

What is the result?

* 1. x = 42
  2. x = 43
  3. x = 44
  4. Compilation fails.
  5. The code runs with no output.

1. Given:

class Mixer {

Mixer() { }

Mixer(Mixer m) { m1 = m; }

Mixer m1;

public static void main(String[] args) {

Mixer m2 = new Mixer();

Mixer m3 = new Mixer(m2); m3.go();

Mixer m4 = m3.m1; m4.go();

Mixer m5 = m2.m1; m5.go();

}

void go() { System.out.print("hi "); }

}

What is the result?

* 1. hi
  2. hi hi
  3. hi hi hi
  4. Compilation fails
  5. hi, followed by an exception
  6. hi hi, followed by an exception

1. Given:

class Sixties {

public static void main(String[] args) {

int x = 5;

int y = 7;

System.out.print(((y \* 2) % x));

System.out.print(" " + (y % x));

}

}

What is the result?

* 1. 1 1
  2. 1 2
  3. 2 1
  4. 2 2
  5. 4 1
  6. 4 2
  7. Compilation fails.
  8. An exception is thrown at runtime.

1. Given:

class Titanic {

public static void main(String[] args) {

Boolean b1 = true;

boolean b2 = false;

boolean b3 = true;

if((b1 & b2) | (b2 & b3) & b3)

System.out.print("alpha ");

if((b1 = false) | (b1 & b3) | (b1 | b2))

System.out.print("beta ");

}

}

What is the result?

* 1. beta
  2. alpha
  3. alpha beta
  4. Compilation fails.
  5. No output is produced.
  6. An exception is thrown at runtime.

1. Given:

1. class Loopy {

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

3. int[] x = {7,6,5,4,3,2,1};

4. // insert code here

5. System.out.print(y + " ");

6. }

7. } }

Which, inserted independently at line 4, compiles? (Choose all that apply.)

* 1. for(int y : x) {
  2. for(x : int y) {
  3. int y = 0; for(y : x) {
  4. for(int y=0, z=0; z<x.length; z++) { y = x[z];
  5. for(int y=0, int z=0; z<x.length; z++) { y = x[z];
  6. int y = 0; for(int z=0; z<x.length; z++) { y = x[z];

1. Which about the three java.lang classes String, StringBuilder, and StringBuffer are true?

(Choose all that apply.)

* 1. All three classes have a length() method.
  2. Objects of type StringBuffer are thread-safe.
  3. All three classes have overloaded append() methods.
  4. The "+" is an overloaded operator for all three classes.
  5. According to the API, StringBuffer will be faster than StringBuilder under mostimplementations.
  6. The value of an instance of any of these three types can be modified through various methods in the API.

1. Given:

public static void before() {

Set set = new TreeSet();

set.add("2");

set.add(3);

set.add("1");

Iterator it = set.iterator();

while (it.hasNext())

System.out.print(it.next() + " ");

}

Which of the following statements are true?

* 1. The before() method will print 1 2
  2. The before() method will print 1 2 3
  3. The before() method will print three numbers, but the order cannot be determined.
  4. The before() method will not compile.
  5. The before() method will throw an exception at runtime.

1. Which collection class(es) allows you to grow or shrink its size and provides indexed access to its elements, but whose methods are not synchronized? (Choose all that apply.)
   1. java.util.HashSet
   2. java.util.LinkedHashSet
   3. java.util.List
   4. java.util.ArrayList
   5. java.util.Vector
   6. java.util.PriorityQueue
2. Given:

TreeSet map = new TreeSet();

map.add("one");

map.add("two");

map.add("three");

map.add("four");

map.add("one");

Iterator it = map.iterator();

while (it.hasNext() ) {

System.out.print( it.next() + " " );

}

What is the result?

* 1. Compilation fails.
  2. one two three four
  3. four three two one
  4. four one three two
  5. one two three four one
  6. one four three two one
  7. An exception is thrown at runtime.
  8. The print order is not guaranteed.

1. Which are true? (Choose all that apply.)
   1. The notifyAll() method must be called from a synchronized context.
   2. To call wait(), an object must own the lock on the thread.
   3. The notify() method is defined in class java.lang.Thread.
   4. When a thread is waiting as a result of wait(), it release its lock.
   5. The notify() method causes a thread to immediately release its lock.
   6. The difference between notify() and notifyAll() is that notifyAll() notifies all waiting threads, regardless of the object they're waiting on.
2. A ssume you have a class that holds two private variables: a and b. Which of the following

pairs can prevent concurrent access problems in that class? (Choose all that apply.)

* 1. public int read(){return a+b;}

public void set(int a, int b){this.a=a;this.b=b;}

* 1. public synchronized int read(){return a+b;}

public synchronized void set(int a, int b){this.a=a;this.b=b;}

* 1. public int read(){synchronized(a){return a+b;}}

public void set(int a, int b){synchronized(a){this.a=a;this.b=b;}}

* 1. public int read(){synchronized(a){return a+b;}}

public void set(int a, int b){synchronized(b){this.a=a;this.b=b;}}

* 1. public synchronized(this) int read(){return a+b;}

public synchronized(this) void set(int a, int b){this.a=a;this.b=b;}

* 1. public int read(){synchronized(this){return a+b;}}

public void set(int a, int b){synchronized(this){this.a=a;this.b=b;}}