OCP Java Test

1. **What is the result of executing the following code snippet?**

List<Integer> list1 = **new** ArrayList<>(Arrays.*asList*(1,2,3));

List<Integer> list2 = **new** CopyOnWriteArrayList<>(list1);

Map<Integer, Integer> map3 = **new** ConcurrentHashMap<>();

map3.put(1, 2);

map3.put(3, 3);

**for**(Integer item : list1) list1.add(10); // f1

**for**(Integer item : list2) list2.add(item); // f2

**for**(Integer key : map3.keySet()) map3.remove(key); // f3

System.***out***.println(list1.size()+" "+list2.size()+" "+map3.size());

○ A) It outputs 3 6 0.

○ B) It outputs 6 6 0.

○ C) It outputs 6 3 3.

○ D) The code does not compile.

○ E) It compiles but throws an exception at runtime on line f1.

○ F) It compiles but throws an exception at runtime on line f2.

○ G) It compiles but throws an exception at runtime on line f3.

○ H) It compiles but enters an infinite loop at runtime.

1. **Which functional interface has three generic types and an apply() method?**

○ A) BiConsumer

○ B) BinaryConsumer

○ C) BiPredicate

○ D) BinaryPredicate

○ E) BiFunction

○ F) BinaryFunction

1. **What is the result of the following code?**

**public** **class** WhichAnimal {

**enum** AnimalsInPark{

***SQUIRREL***, ***CHIPMUNK***, ***SPARROW***

}

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

AnimalsInPark a = AnimalsInPark.***CHIPMUNK***;

**switch** (a) {

**case** AnimalsInPark.***SQUIRREL***: System.***out***.print("S");

**case** AnimalsInPark.***CHIPMUNK***: System.***out***.print("C");

**default**: System.***out***.print("P");

} } }

○ A) C

○ B) CP

○ C) The enum doesn't compile.

○ D) The main method doesn't compile.

○ E) A runtime exception is thrown.

1. **What is the output of the following?**

**public** **class** SnowStorm {

**static** **class** Walk **implements** AutoCloseable {

**public** **void** close() {

**throw** **new** RuntimeException("snow");

}

}

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

**try** (Walk walk1 = **new** Walk()) {

**throw** **new** RuntimeException("rain");

} **catch** (Exception e) {

System.***out***.println(e.getMessage() + " "+ e.getSuppressed().length);

} } }

○ A) rain 0

○ B) rain 1

○ C) rain 2

○ D) show 0

○ E) snow 1

○ F) snow 3

○ G) The code does not compile.

1. **Which of the following are valid functional interfaces? (Choose all that apply.)**

**public** functional **interface** Chew {

**public** **void** chew();

}

**public** **interface** Eat {

**public** **default** isFed() {**return** **false**;}

}

**public** **interface** EatQuickly **extends** Eat {

**int** eat();

}

**public** **interface** EatEvenFaster **extends** EatQuickly {}

**public** **interface** Swallow **extends** EatEvenFaster {

**public** **abstract** **void** swallow(**int** count);

}

○ A) Eat

○ B) EatQuickly

○ C) Chew

○ D) EatEvenFaster

○ E) Swallow

○ F) None of these are valid functional interfaces.

1. **Which of the following are true statements about the following code?(Choose all that apply.)**

1: **public** **class** Outer {

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

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

4: **class** Inner {

5: **public** **static** **void** print () {

6: System.***out***.println("Roar!");

7: }

8: }

9: }

10: **new** Inner().print();

11: }

12: }

○ A) The code prints Roar! three times.

○ B) Line 4 does not compile.

○ C) Line 5 does not compile.

○ D) Line 10 does not compile.

○ E) A different line does not compile.

○ F) A runtime exception is thrown because an inner class is declared in a loop.

1. **What is the output of the following code if the default system time is US/Eastern?**

LocalDate date = LocalDate.*of*(2100, 5, 14);

LocalTime time = LocalTime.*of*(9, 15);

LocalDateTime dateTime = LocalDateTime.*of*(date, time);

Duration d = Duration.*of*(1, ChronoUnit.***WEEKS***);

System.***out***.println(dateTime.plus(d));

○ A) 2100-05-14T09:15

○ B) 2100-05-14T09:15[US/Eastern]

○ C) 2100-05-21T09:15

○ D) 2100-05-21T09:15[US/Eastern]

○ E) The code does not compile.

○ F) A runtime exception is thrown.

1. **What is the output of the following code?**

**import** java.io.\*;

**public** **class** AutocloseableFlow {

**static** **class** Door{

**public** **void** close() {

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

} }

**static** **class** Window {

**public** **void** close() {

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

} }

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

**try** (Door d = **new** Door(); Window w = **new** Window() ){

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

**throw** **new** RuntimeException();

} **catch** (Exception e) {

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

} **finally** {

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

} } }

○ A) TWF

○ B) TWDF

○ C) TWDEF

○ D) TWF followed by an exception.

○ E) TWDF followed by an exception.

○ F) TWEF followed by an exception.

○ G) The code does not compile.

1. **Assuming the current directory is /home and the directories and files referenced by the program exist, what is true about the following code snippet? (Choose all that apply.)**

Path path1 = Paths.*get*("/lemur/habitat/./party.txt");

Path path2 = path1.subpath(1, 4).toAbsolutePath();

System.***out***.println(Files.*lines*(path2)

.flatMap(p -> Stream.of(p.split(",")))

.filter(s -> s.trim().length()>0)

.allMatch(s -> s.length()>3));

○ A) It will access the file /lemur/habitat/party.txt.

○ B) It will access the file /habitat/party.txt.

○ C) It will access the file /home/habitat/party.txt.

○ D) If the String value ... is contained within the file, it will definitely print false.

○ E) If the String value, hat, is contained within the file, it will definitely print false.

○ F) It will always print false.

○ G) The code does not compile.

1. **Which of the following are properties of classes that are properly encapsulated as a JavaBean? (Choose all that apply)**

○ A) The first letter of the any getter/setter, after the get, set, or is prefix, must be uppercase.

○ B) Instance variables are accessed using public getter methods.

○ C) All instance variables are set by the constructor.

○ D) Boolean instance variables may be accessed with gimme.

○ E) Non-boolean instance variables are accessed with get.

○ F) All instance variables are marked static.

1. **Assuming that the file baboon.txt exists, and it is available for read/write access within the file system, what is the result of executing the following code?(Choose all that apply.)**

Path path = Paths.*get*("baboon.txt");

BasicFileAttributeView view = Files.*getFileAttributeView*(path, BasicFileAttributeView.**class**);

BasicFileAttributes data = view.readAttributes(); //x1

view.setTimes(**null**, **null**, **null**); //x2

○ A) It updates all of the file date/time values to the epoch.

○ B) It does not modify any of the file data/time values.

○ C) The code will not compile because of line x1.

○ D) The code will not compile because of line x2.

○ E) It compiles but throws an exception at runtime.

1. **Which of the following is a valid JDBC URL?**

○ A) jdbc-mysql-1234/db

○ B) jdbc-mysql-localhost:1234/db

○ C) jdbc-mysql-localhost-1234-db

○ D) jdbc:mysql:1234/db

○ E) jdbc:mysql:localhost:1234/db

1. **Which statements about the following code are true?(Choose all that apply.)**

1: Console console = System.*console*();

2: String password = console.readPassword("Enter a password: ");

3: System.***out***.println("You entered: " + password);

○ A) The code will not compile because of line 1.

○ B) The code will not compile because of line 2.

○ C) The code will not compile because of line 3.

○ D) The console reference on line 2 may be null.

○ E) The readPassword method does not use a format specifier.

○ F) It compiles but throws an exception at runtime.

1. **Which of the following calls of this method compile? (Choose all that apply.)**

**public** **static** <T **extends** Set> T work(T t) {

**return** t;

}

○ A) HashSet set = work(new HashSet());

○ B) ArrayDeque set = work(new ArrayDeque());

○ C) Set set = work(new HashSet());

○ D) Set set = work(new ArrayDeque ());

○ E) Set set = work(new Set());

1. **Suppose that you are asked to write a program for a zoo. Every animal has a unique identifier. The customer says the most important thing is for the user to be able to look up the animal quickly based on that identifier. Animals are unique. Which collection type should you use in this scenario?**

○ A) ArrayList

○ B) HashMap

○ C) HashSet

○ D) LinkedList

○ E) TreeMap

○ F) TreeSet

1. **Which of the answer choices are true given the following?(Choose all that apply.)**

2016-08-28T05:00-07:00

2016-08-28T05:00-06:00

○ A) The first date/time is earlier.

○ B) The second date/time is earlier.

○ C) Both date/times are the same.

○ D) The date/times are one hour apart.

○ E) The date/times are three hour apart.

○ F) The date/times are five hour apart.

1. **Which of the following are stored in a Duration object?**

○ A) Year

○ B) Month

○ C) Day

○ D) Hour

○ E) Minute

○ F) Second

1. **Which of the following are valid Callable expressions?(Choose all that apply.)**

○ A) void -> 10

○ B) () -> 10

○ C) (int quantity) -> {return 10;}

○ D) () -> {return;}

○ E) () -> Math.min(10,5)

○ F) () -> {for (int i=0; i<10; i++) {} return 10;}

○ G) () -> “Chipmunk”

1. **What is the result of the following statements?**

Set<Integer> set = **new** HashSet<>();

set.add(**new** Integer(6));

set.add(**new** Integer(6));

System.***out***.println(set.size() + " " + set.contains(6));

○ A) 1 false

○ B) 1 true

○ C) 2 false

○ D) 2 true

○ E) The code does not compile.

○ F) A runtime exception is thrown.

1. **Which of the following are properties of classes that define immutable objects?(Choose all that apply.)**

○ A) All of the instance variables are marked private and final.

○ B) They allow referenced mutable objects to be modified or accessed directly.

○ C) They extend the abstract Immutable class.

○ D) They don’t define any setter methods.

○ E) They have only a single getInstance() public method.

1. **Which of the following changes must be made in order for the following code to print out2?(Choose all that apply.)**

**import** java.util.\*;

**public** **class** Panda {

String name;

Panda(String name) { **this**.name = name; }

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

Set s = **new** HashSet<>();

s.add(**new** Panda("Bao Bao"));

s.add(**new** Panda("Mei Xiang"));

s.add(**new** Panda("Bao Bao"));

System.***out***.println(s.size());

} }

○ A) Have class implement Comparable

○ B) Have class implement Comparator

○ C) Override compare()

○ D) Override compareTo()

○ E) Override equals()

○ F) Override hashCode()

1. **Assume that today is June 1, 2016. What is the result of the following?**

Stream<LocalDate> s = Stream.*of*(LocalDate.*now*());

UnaryOperator<LocalDate> u = l -> l.get(Calendar.***DAY\_OF\_MONTH***);

**long** result = s.filter(l -> l !=**null**).map(u).count();

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

○ A) 0

○ B) 1

○ C) 5

○ D) 6

○ E) The code does not compile.

○ F) An exception is thrown.

1. **What is the result of executing the following application?(Choose all that apply.)**

**import** java.util.concurrent.\*;

**import** java.util.stream.\*;

**public** **class** PrintHolidays {

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

ExecutorService service = Executors.*newScheduledThreadPool*(10);

LongStream.*of*(101,704,1126) //p1

.forEach(() -> service.submit( //p2

c -> System.out.println(2+c))); //p3

service.submit(() -> System.***out***.println("Happy Holidays!")); //p4

}

}

○ A) It compiles and outputs the three numbers, followed by Happy Holidays!

○ B) The code will not compile because of line p1.

○ C) The code will not compile because of line p2.

○ D) The code will not compile because of line p3.

○ E) The code will not compile because of line p4.

○ F) It compiles but the output cannot be determined ahead of time.

○ G) It compiles but throws an exception at runtime.

○ H) It compiles but waits forever at runtime.

1. **What is the result of calling the following method?**

**public** **static** **void** updateRecords(Deque<Integer> deque) **throws** InterruptedException {

deque.offerLast(9, 4, TimeUnit.***DAYS***);

deque.offerFirst(8, 21, TimeUnit.***MICROSECONDS***);

deque.offer(3);

System.***out***.print(deque.pollFirst(3, TimeUnit.***MILLISECONDS***));

System.***out***.print(" "+ deque.pollLast(1, TimeUnit.***SECONDS***));

}

○ A) It outputs 8 9.

○ B) It outputs 3 9.

○ C) It outputs 8 3.

○ D) It outputs 3 8.

○ E) The code will not compile.

○ F) It compiles but throws an exception at runtime.

○ G) The output cannot be determined ahead of time.

1. **Which statements are true about the following code? (Choose all that apply.)**

**public** **interface** UsesBeak {}

**public** **class** Mammal {}

**public** **class** Bird **implements** UsesBeak {}

**public** **class** Platypus **extends** Mammal **implements** UsesBeak { }

○ A) Platypus has-a Beak.

○ B) Bird is-a UsesBeak.

○ C) Bird is-a Mammal.

○ D) Mammal has-a Bird.

○ E) Platypus is-a Bird.

○ F) Platypus is-a Mammal.

1. **Which of the following stream classes are low level?(Choose all that apply.)**

○ A) PrintStream

○ B) FileInputStream

○ C) ObjectOutputStream

○ D) Filereader

○ E) PrintReader

○ F) FileWriter

○ G) InputStream

1. **What is true about the following code snippet? (Choose all that apply.)**

String line;

Console c = System.*console*();

Writer w = c.writer();

**if**((line = c.read()) != **null**)

w.append(line);

w.flush();

○ A) The code runs without error but prints nothing.

○ B) The code prints what the user entered.

○ C) An ArrayIndexOutOfBoundsException might be thrown.

○ D) A NullPointerException might be thrown.

○ E) An IOException might be thrown.

○ F) The code does not compile.

1. **What is the result of the following code?**

**public** **class** FourLegged {

String walk = "walk,";

**public** **void** walk(){

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

}

**static** **class** BabyRhino **extends** FourLegged {

String walk = "toddle,";

}

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

FourLegged f = **new** BabyRhino();

BabyRhino b = **new** BabyRhino();

f.walk();

b.walk();

}

}

○ A) toddle, toddle,

○ B) toddle, walk,

○ C) walk, toddle,

○ D) walk, walk,

○ E) The code does not compile.

○ F) A runtime exception is thrown.

1. **What is the result of executing the following application?(Choose all that apply.)**

**public** **class** PerfectSquares {

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

ExecutorService service = Executors.*newSingleThreadExecutor*();

IntStream.*of*(1,4,9,16,25).parallel() //z1

.forEachOrdered(c -> service.submit( //z2

() -> System.***out***.println(10\*c))); //z3

service.submit(() -> System.***out***.println("complete!")); //z4

service.shutdown();

}

}

○ A) It compiles and outputs the five numbers, followed by Complete!

○ B) The code will not compile because of line z1.

○ C) The code will not compile because of line z2.

○ D) The code will not compile because of line z3.

○ E) The code will not compile because of line z4.

○ F) It compiles but the output cannot be determined ahead of time.

○ G) It compiles but throws an exception at runtime.

○ H) It compiles but waits forever at runtime.

1. **Which of the following methods are overrides of the methods implemented in java.lang.Object?(Choose all that apply.)**

○ A) public boolean equals(String string) { return false; }

○ B) public boolean equals(Object obj) { return false; }

○ C) public int hashcode() { return 1; }

○ D) public int hashCode() { return 1; }

○ E) public String asString { return ""; }

○ F) public String toString{ return ""; }

1. **What is the result of the following code?**

**public** **class** FourLegged {

String walk = "walk,";

**public** **void** walk(){

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

}

**static** **class** BabyRhino **extends** FourLegged {

String walk = "toddle,";

@Override **public** **void** walk() {

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

}

}

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

FourLegged f = **new** BabyRhino();

BabyRhino b = **new** BabyRhino();

f.walk();

b.walk();

}

}

○ A) toddle, toddle,

○ B) toddle, walk,

○ C) walk, toddle,

○ D) walk, walk,

○ E) The code does not compile.

○ F) A runtime exception is thrown.

1. **What is the result of this program?**

1: **public** **class** Color {

2: **private** **int** hue = 10;

3: **public** **class** Shade {

4: **public** **int** hue = Color.**this**.hue;

5: }

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

7: System.***out***.println(**new** Shade().hue );

8: } }

○ A) 0

○ B) 10

○ C) A compiler error occurs on line 3.

○ D) A compiler error occurs on line 4.

○ E) A compiler error occurs on line 7.

1. **Which statements are true about the following code?(Choose all that apply.)**

**public** **class** Fins {}

**public** **class** Gills {}

**public** **class** Fish {**public** Fins fins; }

**public** **class** Trout {

**public** Fish fish;

**public** Gills gills;

}

○ A) Fish has-a Fins.

○ B) Trout has-a Fins.

○ C) Trout is-a Fish.

○ D) Fish is-a Trout.

○ E) Trout has-a Fish.

○ F) Trout has-a gills.

1. **What are some ways a developer can ensure that there is at most one instance of a class within an application using the singleton pattern?(Choose all that apply.)**

○ A) Add a public static getInstance() method to the class.

○ B) Mark the singleton object final.

○ C) Mark all constructors private.

○ D) Name the singleton object instance.

○ E) Add synchronized to any method declaration that creates the singleton.

1. **Which of the following are true when this code is run with java Fail? (Choose all that apply.)**

**public** **class** Fail {

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

**int** x = 7;

**assert** x >= 1 && x <= 6;

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

}

}

○ A) There is a compiler error.

○ B) The code throws an AssertionError.

○ C) The code thrown an AssertionException.

○ D) The output is 7.

○ E) The assert statement is ignored.

1. **Which of the following are used by default?(Choose all that apply.)**

○ A) ResultSet.CONCUR\_READ\_ONLY

○ B) ResultSet.CONCUR\_UPDATABLE

○ C) ResultSet.TYPE\_FORWORD\_ONLY

○ D) ResultSet.TYPE\_SXROLL\_INSENSITIVE

○ E) ResultSet.TYPE\_SCROLL\_SENSITIVE

1. **What is the result of the following code?**

**public** **class** FruitStore {

**public** **enum** Fruit {

***APPLE***("red"), ***BANANA***("yellow"), ***ORANGE***("orange"),***PLUM***("purple");

**private** Fruit(String color) {**this**.color = color;}

**public** String color;

}

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

Fruit one = Fruit.***PLUM***;

System.***out***.println("a " + one + " is " + one.color);

}

}

○ A) A PLUM is purple.

○ B) A Fruit.PLUM is purple.

○ C) The enum does not compile.

○ D) The main method does not compile.

○ E) An exception is thrown.

1. **Suppose that you have a table animal with three rows. The names in those rows are Anna, Betty, and Cat. What does the following output?**

String sql = "select name from animal order by name";

**try** (Connection conn = DriverManager.*getConnection*("jdbc:derby:zoo");

Statement stmt = conn.createStatement(

ResultSet.***CONCUR\_READ\_ONLY***,ResultSet.***TYPE\_SCROLL\_INSENSITIVE***);

ResultSet rs = stmt.executeQuery(sql)){

rs.next();

rs.previous();

rs.previous();

rs.next();

rs.next();

rs.absolute(2);

System.***out***.println(rs.getString(1));

}

}

}

○ A) Anna

○ B) Betty

○ C) Cat

○ D) The code does not compile.

○ E) A SQLException is thrown.

1. **What is the result of the following program?**

**public** **class** StringSearch {

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

List<String> list = **new** ArrayList<String>();

list.add("ab"); list.add("ba");

list.add("bd"); list.add("aa");

Comparator<String> comparator = (a,b) -> b.compareToIgnoreCase(a);

Collections.*sort*(list, comparator);

**int** index = Collections.*binarySearch*(list, "ab", comparator);

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

}

}

○ A) 1

○ B) 2

○ C) 3

○ D) 4

○ E) The code does not compile.

○ F) A runtime exception is thrown.

1. **What are some differences between an interface and an abstract class? (Choose all that apply.)**

○ A) Interfaces cannot be marked abstract.

○ B) Only abstract classes can declare static methods.

○ C) Only interfaces can declare default methods.

○ D) Only abstract classes can have protected variables.

○ E) Only abstract classes can have non-final variables.

○ F) Only interfaces can have static final variables.

1. **What is the result of the following code?**

**public** **class** WhichBrowser {

**class** Browser { }

**class** Firefox **extends** Browser { }

**class** IE **extends** Browser { }

**private** **void** check() {

Browser ref = **new** IE();

**if**(ref **instanceof** Firefox) {

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

}**else** **if** (ref **instanceof** Browser) {

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

}**else** **if**(ref **instanceof** IE) {

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

}**else**{

System.***out***.println("None of the above");

}

}

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

**new** WhichBrowser().check();

}

}

○ A) Browser

○ B) Firefox

○ C) IE

○ D) None of the above.

○ E) The code does not compile.

○ F) A runtime exception is thrown.

1. **Assuming in and out are properly defined and instantiated InputStream and OutputStream objects, respectively, which are true statements about the following code snippet?(Choose all that apply.)**

5: **final** **byte**[] buffer = **new** **byte**[100];

6: **int** lengthRead;

7: **while** ((lengthRead = in.read(buffer)) > 0) {

8: out.flush();

9: out.write(buffer,lengthRead);

10: out.flush();

○ A) The code will not compile because the buffer reference is marked final.

○ B) The code will throw an exception t runtime because the size of the buffer is not a power of line 2.

○ C) The code will not compile due to line 7.

○ D) The code will not compile due to line 9.

○ E) The code will compile and throw an exception at runtime on line 10.

○ F) The code will compile and run without issue.

○ G) None of the above.

1. **Which statements about the following class definition are true? (Choose all that apply.)**

1: **public** **class** FoodStorage {

2: **private** **synchronized** **int** apples;

3: **public** **synchronized** **int** getAppleCount() { **return** apples; }

4: **public** **synchronized** **void** stock(**int** applesToStock) { apples += applesToStock; }

5: **public** **synchronized** **void** eatten(**int** eatenCount) {

6: **synchronized** (**this**) {

7: apples -= eatenCount;

8: }

9: }

10: }

○ A) It compiles without issue.

○ B) The code will not compile because of line 2.

○ C) The code will not compile because of line 4.

○ D) The code will not compile because of line 7.

○ E) The class could result in a deadlock due to line 6 and 7 synchronized on the same object.

○ F) The locks acquired on line 3 and 4 are on the same object.

1. **What statements about the following code snippet are true?**

8: Stream<Integer> prime = Stream.*of*(2,3,5,7);

9: Stream<Integer> composite = Stream.*of*(4,6,8);

10: ConcurrentMap<Boolean, List<Integer>> data = Stream.combine(prime,composite)

11: .flatMap(s -> s).parallelStream()

12: .collect(Collectors.*groupingByConcurrent*(n -> (n%2)==0));

13: System.***out***.println(data.get(**true**).size()+" "+data.get(**false**).size());

○ A) It outputs 3 4.

○ B) It outputs 4 3.

○ C) The code will not compile because of line 10.

○ D) The code will not compile because of line 11.

○ E) The code will not compile because of line 12.

○ F) It compiles but throws an exception at runtime.

○ G) The collect() operation is always executed concurrently.

1. **What is the result of executing the following application? (Choose all that apply.)**

**import** java.util.concurrent.\*;

**public** **class** SpinWheels **extends** RecursiveTask { //k1

**private** **int** a;

**private** **int** c;

**public** SpinWheels(**int** a, **int** c) {

**this**.a = a;

**this**.c = c;

}

**protected** Object compute() { //k2

**if** (a <= 0) **return** 10;

**else** {

**int** b = a + ((c - a) / 2);

*invokeAll*(**new** SpinWheels(a,b), **new** SpinWheels(b,c)); //k3

**return** ""; //k4

} }

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

ForkJoinTask task = **new** SpinWheels(0,100);

ForkJoinPool pool = **new** ForkJoinPool();

**int** x = (Integer)pool.invoke(task); //k4

} }

○ A) It compiles and runs without issue.

○ B) The code will not compile because of line k1.

○ C) The code will not compile because of line k2.

○ D) The code will not compile because of line k3.

○ E) The code will not compile because of line k4.

○ F) It compiles but throws an exception at runtime.

○ G) It compiles but hangs at runtime.

1. **What is the result of the following?**

Map<Integer, Integer>map = **new** HashMap<>();

map.put(1, 10);

map.put(2, 20);

map.put(3, **null**);

map.merge(1, 3, (a,b) -> **null**);

map.merge(3, 3, (a,b) -> **null**);

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

○ A) {}

○ B) {2=20}

○ C) {2=20, 3=null}

○ D) {2=20, 3=3}

○ E) The code does not compile.

○ F) An exception is thrown.

1. **Which of the following can fill in the blank to make the code compile?(Choose all that apply.)**

**import** java.util.stream.\*;

**public** **class** DragonBoat {

\_\_\_\_\_\_\_\_\_ DragonBoat boat;

**public** **static** **void** row() {}

**public** **static** **void** race() {

boat = **new** DragonBoat();

IntStream.range(1, 100).forEach(i -> boat.row());

}

}

○ A) private

○ B) private static

○ C) public

○ D) public static

○ E) secret

○ F) secret static

1. **What is the output of the following code?**

**import** java.io.\*;

**public** **class** AutocloseableFlow {

**static** **class** Door **implements** AutoCloseable {

**public** **void** close() {

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

} }

**static** **class** Window **implements** Closeable{

**public** **void** close() {

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

} }

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

**try** (Door d = **new** Door(); Window w = **new** Window() ){

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

**throw** **new** RuntimeException();

} **catch** (Exception e) {

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

} **finally** {

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

} } }

○ A) TWF

○ B) TWDF

○ C) TWDEF

○ D) TWF followed by an exception.

○ E) TWDF followed by an exception.

○ F) TWEF followed by an exception.

○ G) The code does not compile.

1. **What of the following are true statements about working with instances of the OutputStrema class?(Choose all that apply.)**

○ A) They must be bufferd.

○ B) They can be used to read input from the user.

○ C) They can be used to write byte data to a file.

○ D) They can be written forword and backward.

○ E) They can be used to write character data to a file.

○ F) They should be closed after use.

○ G) They must be periodically flushed.

1. **Which of the following can fill in the blank to make the code compile? (Choose all that apply.)**

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

**try** {

process();

}**catch** (\_\_\_\_\_\_\_\_\_\_\_) {}

}

**private** **static** **void** process() **throws** FileNotFoundException, IllegalArgumentException{}

○ A) FileNotFoundException | IOException e

○ B) FileNotFoundException | InterruptedException e

○ C) FileNotFoundException | IllegalArgumentException e

○ D) IllegalArgumentException | InterruptedException e

○ E) IOException | InterruptedException e

○ F) IOException | IllegalArgumentException e

1. **Which of the answer choices produce the same output as this traditional for loop?(Choose all that apply.)**

**for** (String s : strings)

**if** (!s.equals("gecko"))

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

○ A) strings

.filter(p -> p.equals("gecko"))

.forEach(System.out :: println);

○ B) strings

.filter(p -> !p.equals("gecko"))

.forEach(System.out :: println);

○ C) strings

.filter(!p -> p.equals("gecko"))

.forEach(System.out :: println);

○ D) strings.stream()

.filter(p -> p.equals("gecko"))

.forEach(System.out :: println);

○ E) strings.stream()

.filter(p -> !p.equals("gecko"))

.forEach(System.out :: println);

○ F) strings.stream()

.filter(!p -> p.equals("gecko"))

.forEach(System.out :: println);

1. **Which are true about the following code? (Choose all that apply.)**

1: **public** **enum** Suit {

2: ***SPADE***(Color.***BLACK***), ***HEART***(Color.***RED***),

3: ***DIMOND***(Color.***RED***),***CLUB***(Color.***BLACK***);

4: **private** **enum** Color { ***RED***, ***BLACK***; }

5: **private** Suit(Color c) { color = c; }

6: **public** Color color;

7: }

○ A) The Suit enum compiles successfully.

○ B) An enum cannot declare an inner enum.

○ C) The constructor on line 5 cannot be changed to public or protected.

○ D) Color.Black can be replaced by BLACK on line 2.

○ E) The Color enum on line 4 must be declared private.

1. **Assuming Donkey is an existing class that properly implements the Serializable interface and dataFile refers to a valid File object that exists within the file system,what statements about the following code snippet are true?(Choose all that apply.)**

1: List<Donkey> donkeys = **new** ArrayList<Donkey>();

2: **try** (ObjectInputStream in = **new** ObjectInputStream(

3: **new** BufferedInputStream(**new** FileReader(dataFile)))) {

4: **while**(**true**) {

5: Object object = in.readObject();

6: **if**(object **instanceof** Donkey)

7: donkeys.add(object);

8: }

9: }**catch**(EOFException e) { }

○ A) The code will not compile because of line 3.

○ B) The code will not compile because of line 6.

○ C) The code will not compile because of line 7.

○ D) The code will compile but throw an exception at runtime.

○ E) The code will compile and run without issue.

○ F) None of the above

1. **What is the output of the following method if props contains {veggies=brontosaurus, meat=velociraptor}?**

**public** **static** **void** print(Properties props) {

System.***out***.println(props.getProperty("veggies","none")

+ " " + props.getProperty("omni","none")); }

○ A) brontosaurus none

○ B) brontosaurus null

○ C) none none

○ D) none null

○ E) The code does not compile.

○ F) A runtime exception is thrown.

1. **Which of the following are valid lambda expressions?(Choose all that apply.)**

○ A) () -> {return 30;}

○ B) a,d -> d.quack()

○ C) (Duck d) -> {int x=1; return d;}

○ D) y -> {return y;}

○ E) Wolf w -> 39

○ F) (h, Ocelot o) -> 41

○ G) (int m, int m) -> 41

1. **What is the result of the following statements?**

Set points = **new** TreeSet();

points.add(7);

points.add(5);

points.add(-4);

points.add(6);

**for** (Number point : points)

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

○ A) 75-45

○ B) 765-4

○ C) -4567

○ D) The output is indeterminate.

○ E) The code does not compile.

○ F) An exception is thrown.

1. **Suppose the directory c:\temp exists, but no subdirectories of it exist. Which line of code would be the best way to create the directory c:\temp\book\java?**

○ A) File.mkdir("c:\\temp\\book\\java");

○ B) File.mkdirs("c:\\temp\\book\\java");

○ C) Files.mkdir("c:\\temp\\book\\java");

○ D) Files.mkdirs("c:\\temp\\book\\java");

○ E) new File("c:\\temp\\book\\java").mkdir();

○ F) new File("c:\\temp\\book\\java").mkdirs();

1. **What changes need to be made to make the following singleton pattern correct?(Choose all that apply.)**

**public** **class** ExhibitionManager {

**public** **final** **static** ExhibitionManager ***exhibitionManager***;

**static** {

***exhibitionManager*** = **new** ExhibitionManager();

}

**protected** ExhibitionManager() {}

**public** **static** ExhibitionManager getExhibitionManager() {

**return** **new** ExhibitionManager();

}

}

○ A) None, the singleton pattern is properly implemented.

○ B) Remove the final from the exhibitionManager variable declaration.

○ C) Change the ExhibitionManager constructor to private.

○ D) Synchronize the getExhibitionManager() method.

○ E) Rename exhibitionManager to be instance.

○ F) Rename the getExhibitionManager() to be getInstance().

○ G) Change the getExhibitionManager() method to return the static exhibitionManager object.

1. **Which functional interfaces complete the following code?(Choose all that apply.)**

\_\_\_\_\_\_\_ x = LocalDate::now;

\_\_\_\_\_\_\_ y = Files::exists;

○ A) Consumer<LocalDate>

○ B) Consumer<Path>

○ C) Predicate<LocalDate>

○ D) Predicate<Path>

○ E) Supplier<LocalDate>

○ F) Supplier<Path>

1. **Which of the following can fill in the blank correctly?(Choose all that apply.)**

ResultSet rs = stmt.executeQuery(sql); if(rs.next()) {

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

}

○ A) int num = rs.getInt(0);

○ B) int num = rs.getInt(1);

○ C) int num = rs.getInteger(0);

○ D) int num = rs.getInteger(1);

○ E) int num = rs.getObject(0);

○ F) int num = rs.getObject(1);

1. **What is true of the following code?**

**import** java.io.\*;

**public** **class** CustomException **extends** IOException **implements** AutoCloseble {

@Override

**public** **void** close() **throws** CustomException {

**throw** **new** CustomException ();

}

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

**try** (CustomException c = **new** CustomException()) {

**throw** **new** CustomException();

}

}

}

○ A) The code does not compile.

○ B) The code runs without throwing CustomException.

○ C) The code throws CustomException but no suppressed exceptions.

○ D) The code throws CustomException and CustomException as suppressed exceptions.

○ E) The code throws CustomException and also a CustomException as a suppressed exception.

1. **Which functional interfaces can fill in the blanks to make the code compile?(Choose all that apply.)**

**public** **void** doubles(\_\_\_\_\_ x, \_\_\_\_\_\_ y) {

**double** d1 = x.applyAsDouble("");

**double** d2 = y.applyAsDouble(1L);

z.accept(1.0);

}

○ A) consumer<Double>

○ B) DoubleConsumer

○ C) Function<Long, Double>

○ D) Function<String, Double>

○ E) LongToDoubleFunction

○ F) ToDoubleFunction<String>

1. **Which of the following statements can be inserted in the blank so that the code will compile successfully?(Choose all that apply.)**

**public** **class** Snake {}

**public** **class** Cobra **extends** Snake {}

**public** **class** GardenSnake {}

**public** **class** SnakeHandler {

**private** Snake snake;

**public** **void** setSnake(Snake snake) { **this**.snake = snake; }

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

**new** SnakeHandler().setSnake(\_\_\_\_\_\_);

}

}

○ A) new Cobra()

○ B) new GardenSnake()

○ C) new Snake()

○ D) new Object()

○ E) new String("Snake")

○ F) null

1. **Which of the following can be inserted in main? (Choose all that apply.)**

**public** **static** **void** call(Object obj) { }

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

//INSERT CODE HERE

}

○ A) call(new Runnable() {});

○ B) call(new Runnable() {

public void run(){ }

})

○ C) call(new Runnable() {

public void run(){ }

});

○ D) call(newThread() { } );

○ E) call(newThread() {

public void run(){ }

})

○ F) call(newThread() {

public void run(){ }

});

1. **What command line would cause this program to fail on the assertion?**

**public** **class** On {

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

String s = **null**;

**assert** s != **null**;

}

}

○ A) java -da On

○ B) java -ea On

○ C) java -Dda On

○ D) java -Dea On

○ E) The code does not compile.

1. **Which functional interfaces complete the following code? (Choose all that apply.)**

\_\_\_\_\_\_\_\_\_ x = a -> **true**;

\_\_\_\_\_\_\_\_\_ y = () -> System.out.println();

\_\_\_\_\_\_\_\_\_ z = (a,b) -> a - b;

○ A) BiPredicate<Integer, Integer>

○ B) Comparable<Integer>

○ C) Comparator<Integer>

○ D) Consumer<Integer>

○ E) Predicate<Integer>

○ F) Runnable

○ G) Runnable<Integer>

1. **What is true of the following code?(Choose all that apply.)**

2: **public** **class** School {

3: **private** List<Object> exceptions;

4: **private** List<?> names = **new** ArrayList<Object>();

5: **public** School() {

6: exceptions = **new** LinkedList<>();

7: }

8: }

○ A) There is a compiler error on line 4.

○ B) There is a compiler error on line 6.

○ C) We can add objects to exceptions.

○ D) We can add objects to names.

○ E) None of the above.

1. **Which of the following statements about Path.equals()and Files.issameFile() are correct? (Choose all that apply.)**

○ A) Both methods have the exact same function.

○ B) Path.equals() always throws an exception if either Path does not exist.

○ C) Files.isSameFile() always throws an exception if either Path does not exist.

○ D) Path.equals() requires a checked exception to be handled by the process that calls it.

○ E) Files.isSameFile() requires a checked exception to be handled by the process that calls it.

○ F) Both methods are reflexive; comparing a path with itself returns true for both.

1. **Assume that all bundles mentioned in the answers exist and define the same keys. Which resource bundle will be used to find the key in line 8?**

Locale.*setDefault*(**new** Locale("en", "US"));

ResourceBundle b = ResourceBundle("Dolphins");

b.getString("name");

○ A) Dolphins.properties

○ B) Dolphins\_fr.java

○ C) Dolphins\_fr.properties

○ D) Whales.properties

○ E) Whales\_en\_US.properties

○ F) The code does not compile.

1. **Which of the following fill in the blank on line 7 so that the program can compile and run without throwing an exception? (Choose all that apply.)**

5: DoubleStream ds = DoubleStream.*empty*();

6: OptionalDouble opt = ds.findAny();

7: System.***out***.println(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_);

○ A) opt.orElse(0)

○ B) opt.orElseGet(() -> 0)

○ C) opt.orElseThrow(RuntimeException :: new)

○ D) opt.get()

○ E) opt.getAsDouble()

○ F) Line 6 throws an exception at runtime.

**71. What is the result of executing the following application? (Choose all that apply.)**

**public** **class** SaladBarTracker {

**public** **static** **void** await(CyclicBarrier cb) {

**try** {

cb.await();

} **catch** (InterruptedException | BrokenBarrierException e) {

// Handle exception

}

}

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

CyclicBarrier cb = **new** CyclicBarrier(4,

() -> System.***out***.println("Salad bar empty")); //r1

ExecutorService service = Executors.*newFixedThreadPool*(4); //r2

**for**( **int** i=0; i<12; i++) {

service.submit(() -> *await*(cb)); //r3

}

service.shutdown();

}

}

○ A) It outputs Salad bar empty exactly once.

○ B) It outputs Salad bar empty multiple times.

○ C) The code will not compile because of line r1.

○ D) The code will not compile because of line r2.

○ E) The code will not compile because of line r3.

○ F) It compiles but throws an exception at runtime.

○ G) It compiles but waits forever at runtime.

1. **What is the output of the following?**

Stream<String> s = Stream.*empty*();

Stream<String> s2 = Stream.*empty*();

Predicate<String> condition = b -> b.startsWith("c");

Map<Boolean, List<String>> p = s.collect(

Collectors.*partitioningBy*(condition));

Map<Boolean, List<String>> g = s2.collect(

Collectors.*groupingBy*(condition));

System.***out***.println(p + " " + g);

○ A) {} {}

○ B) {} {false=[], true=[]}

○ C) {false=[], true=[]} {}

○ D) {false=[], true=[]} {false=[], true=[]}

○ E) The code does not compile.

○ F) An exception is thrown.

1. **Which of the following are valid functional interfaces? (Choose all that apply.)**

**public** **interface** Bark {

**public** **static** **int** bark() {**return** 10;}

}

**public** **interface** Sing {}

**public** **interface** Woof **extends** Bark {

Long wolf();

}

**public** **interface** Meow {

**public** **int** meow();

**public** **default** **void** purr() {}

}

**interface** Whistle **extends** Meow {}

○ A) Sing

○ B) Bark

○ C) Meow

○ D) Whistle

○ E) Woof

○ F) None of these are valid functional interfaces.

1. **What is the result of this program?**

1: **public** **class** PokerHand {

2: **public** **void** play() {

3: **int** numberOfPlayers = 6;

4: **class** Dealer {

5: **public** **void** deal() {

6: **for**(**int** i = 1; i <= numberOfPlayers; i++)

7: System.***out***.print(i);

8: }

9: }

10: Dealer dealer = **new** Dealer();

11: dealer.deal();

12: }

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

14: PokerHand hand = **new** PokerHand();

15: hand.play();

16: }

17: }

○ A) 123456

○ B) A compiler error occurs on line 4.

○ C) A compiler error occurs on line 6.

○ D) A compiler error occurs on line 10.

○ E) A compiler error occurs on a different line.

1. **What is the result of executing the following application?(Choose all that apply.)**

**import** java.util.concurrent.\*;

**public** **class** CheckWeather **extends** RecursiveTask { //j1

**private** **int** start;

**private** **int** end;

**public** CheckWeather(**int** start, **int** end) {

**this**.start = start;

**this**.end = end;

}

**protected** **void** compute() { //j2

**if**((end-start) < 2)

**return**;

**else**{

**int** middle = start + ((end-start) / 2);

*invokeAll*(**new** CheckWeather(start, middle), **new** CheckWeather(middle,end)); //j3

}

}

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

ForkJoinTask task = **new** CheckWeather(0, 4);

ForkJoinPool pool = **new** ForkJoinPool();

pool.invoke(task);

}

}

○ A) It compiles and runs without issue.

○ B) The code will not compile because of j1.

○ C) The code will not compile because of j2.

○ D) The code will not compile because of j3.

○ E) It compiles but throws an exception at runtime.

○ F) It compiles but hangs at runtime.

1. **Assume that all bundles mentioned in the answers exist and define the same keys. Which resource bundle will be used to find the key in line 8?**

6: Locale.*setDefault*(**new** Locale("en", "US"));

7: ResourceBundle b = ResourceBundle.*getBundle*("Dolphins");

8: b.getString("name");

○ A) Dolphins.properties

○ B) Dolphins\_fr.java

○ C) Dolphins\_fr.properties

○ D) Whales.properties

○ E) Whales\_en\_US.properties

○ F) The code does not compile.abstract

1. **Which are true statements about the following code? (Choose all that apply.)**

**public** **class** Numbers {

**int** num;

**public** Numbers(**int** n) {

num = n;

}

**public** **int** compareTo(Numbers n) {

**return** num - n.num;

}

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

Numbers x = **new** Numbers(\_\_\_);

Numbers y = **new** Numbers(\_\_\_);

List<Numbers>list = **new** ArrayList<Numbers>();

}

}

○ A) If x.compareTo(y) returns 0, the numbers passed to the constructors are guaranteed to have been the same.

○ B) If x.compareTo(y) returns 1, the numbers passed to the constructors are guaranteed to have been the same.

○ C) If x.compareTo(y) returns -1, the numbers passed to x's constructor is guaranteed to have been larger than the number passed to y's constructor.

○ D) If x.compareTo(y) returns -1, the numbers passed to x's constructor is guaranteed to have been smaller than the number passed to y's constructor.

○ E) If x and y are added to list, Collections.sort(list) will properly sort them.

1. **If the current working directory is / coralreef, then what is the output of the following code?**

Path userDirectory = Paths.*get*("/coralreef/../clown/fish ").normalize(); //m1

Path panterDirectory = userDirectory.getRoot().getParent().resolve("dolphin");//m2

System.***out***.println(pantherDirectory.toAbsolutePath()); //m3

○ A) /panther

○ B) /coralreef/dolphin

○ C) The code will not compile because of line m1.

○ D) The code will not compile because of line m2.

○ E) The code will not compile because of line m3.

○ F) It compiles but throws an exception at runtime.

1. **Which of the following are properties of classes that define immutable objects?(Choose all that apply.)**

○ A) They don’t allow referenced mutable objects to be modified or accessed directly.

○ B) They have a final constructor.

○ C) They must extend the IsImmutable interface.

○ D) They use a constructor to set all properties of the object.

○ E) They include a serialVersionUID variable.

○ F) They prevent methods from being overridden.

1. **Assume that you have an InputStream whose next bytes are DINO. What would the next char be that could be read from the stream after calling the following method on the stream, using a count value of 2?**

**public** **static** String pullBytes(InputStream is, **int** count) **throws** IOException {

**final** StringBuilder sb = **new** StringBuilder();

sb.append((**char**)is.read());

**if**(is.markSupported()){

is.mark(100);

**for**(**int** i=0; i<count; i++)

sb.append((**char**)is.read());

is.skip(20);

is.reset();

}

sb.append((**char**)is.read());

**return** sb.toString();

}

}

○ A) D

○ B) I

○ C) N

○ D) O

○ E) The code does not compile.

○ F) The code compiles but throws an exception at runtime.

○ G) The result cannot be determined with the information given.

1. **Which of the following are valid Callable expressions?(Choose all that apply.)**

○ A) () -> -1

○ B) () -> ()

○ C) () -> return 102.4

○ D) (int x) -> 19\*x

○ E) () -> {return null;}

○ F) () -> 10/0

○ G) () -> {retuen 1; System.out.println(“Hippo”);}

1. **Assume that today is June 1, 2016. What is the result of the following?**

Stream<LocalDate> s = Stream.*of*(LocalDate.*now*());

UnaryOperator<LocalDate> u = l -> l;

System.***out***.println(s.filter(l -> l != **null**).map(u));

○ A) 2016-05-01

○ B) 2016-06-01

○ C) There is no output.

○ D) The output is something other than 5 or 6.

○ E) The code does not compile.

○ F) An exception is thrown.

1. **Which is a true statement about the following code? (Choose all that apply.)**

**public** **class** WhatisIt {

**static** **interface** Furry { }

**static** **class** Chipmunk { }

**static** **class** FurryChipmunk **implements** Furry { }

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

Chipmunk c = **new** Chipmunk();

**int** result = 0;

**if**(c **instanceof** Furry) result += 1;

**if**(c **instanceof** Chipmunk) result += 2;

**if**( **null** **instanceof** FurryChipmunk) result +=4;

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

} }

○ A) The code compiles, and the output is 1.

○ B) The code compiles, and the output is 2.

○ C) The code compiles, and the output is 7.

○ D) c instanceof Chipmunk does not compile.

○ E) l instanceof Chipmunk does not compile.

○ F) r instanceof Chipmunk does not compile.

1. **Which of the following can be inserted to override the superclass method?(Choose all that apply.)**

**public** **class** LearnToWalk {

**public** **void** toddle() {}

**class** BabyRhino **extends** LearnToWalk {

// INSERT CODE HERE

}

}

○ A) private void toddle() {}

○ B) protected void toddle() {}

○ C) public int toddle() {}

○ D) public void toddle() {}

○ E) public void toddle() throws RuntimeException {}

○ F) void toddle() {}

1. **Which of the following can you add after line 5 for the code to run without error and not produce any output?(Choose all that apply.)**

4: IntegerStream ls = IntegerStream.of(1, 2, 3);

5: OptionalInteger opt = ls.map(n -> n \* 10).filter(n -> n <5).findFirst();

○ A) if (opt.isPresent()) System.out.println(opt.get());

○ B) if (opt.isPresent()) System.out.println(opt.getAsInt());

○ C) if (opt.isPresent()) System.out.println(opt.getAsInteger());

○ D) opt.ifPresent(System.out::println)

○ E) The code does not compile.

○ F) Line 5 throws an exception at runtime.

1. **What is the result of the following code?**

TreeMap<Integer, String> treeMap = **new** TreeMap<>();

treeMap.put(3, "a");

treeMap.put(3, "a");

treeMap.put(2, "b");

treeMap.put(5, "c");

**for** (Number number: treeMap.keySet())

System.***out***.print(treeMap.get(number));

○ A) abc

○ B) aac

○ C) bac

○ D) baac

○ E) The code does not compile.

○ F) A runtime exception is thrown.

1. **What is the value of name after an instance of Whale is serialized and then deserialized?**

**public** **class** Mammal {

**protected** **transient** String name = "Moby";

**public** **void** setName(String name) { **this**.name = name; }

**public** String grtName() { **return** name; }

**public** Mammal() {

**this**.name = "Monstro";

} }

**public** **class** Whale **extends** Mammal **implements** Serializable {

{ **this**.name = "Lisa"; }

**public** Whale() {

**this**.name = "Roger";

} }

○ A) Moby

○ B) Monstro

○ C) Lisa

○ D) Roger

○ E) null

○ F) The code does not compile.

○ G) The code compiles but throws an exception at runtime.

○ H) The value may not be known until runtime.

1. **Which of the following are true? (Choose all that apply.)**

List<Integer> l1 = Arrays.*asList*();

List<Integer> l2 = Arrays.*asList*(1, 2, 3);

List<Integer> l3 = Arrays.*asList*(4, 5, 6);

Stream.*of*(l1, l2, l3)

.limit(2)

.peek(System.***out***::println) //peek 1

.flatMap(x -> x.stream())

.peek(System.***out***::println) // peek 2

.map(x -> x + 1);

○ A) Zero lines are generated by the line marked by peek 1.

○ B) Two lines are generated by the line marked by peek 1.

○ C) Three lines are generated by the line marked by peek 1.

○ D) Zero lines are generated by the line marked by peek 2.

○ E) Two lines are generated by the line marked by peek 2.

○ F) Three lines are generated by the line marked by peek 2.

○ G) The code does not compile.

○ H) The code throws an exception.

1. **Which of the following is a valid JDBC URL?**

○ A) jdbc:oracle:123.123.123.123

○ B) jdbc:oracle:thin:123.123.123.123

○ C) jdbc:oracle:thin:123.123.123.123//fun

○ D) oracle:jdbc:123.123.123.123

○ E) oracle:jdbc:thin:123.123.123.123

○ F) oracle:jdbc:thin:123.123.123.123//fun

1. **Assuming that the following class has proper public getter/setter methods for all of its private fields, which of the following fields will always be null after an instance of the class is serialized and then deserialized?(Choose all that apply.)**

**public** **class** Dolphin {

**private** **transient** String name = "Flipper";

**private** **static** String *birthPlace* = "ocean";

**private** **transient** **int** age;

**private** java.util.List<Dolphin> friends = **new** java.util.ArrayList<>();

}

○ A) name

○ B) age

○ C) birthPlace

○ D) friends

○ E) The code does not compile.

○ F) The code compiles but throws an exception at runtime.

1. **Which of the following are true? (Choose all that apply.)**

**private** **static** **void** magic(Stream<Integer>s) {

Optional o = s.filter(x -> x < 5).max((x,y) -> x - y);

System.***out***.println(o.get());

}

○ A) magic(Stream.empty());runs infinitely.

○ B) magic(Stream.empty());throws an exception.

○ C) magic(Stream.iterate(1, x -> x++));runs infinitely.

○ D) magic(Stream.iterate(1, x -> x++));throws an exception.

○ E) magic(Stream.of(5, 10));runs infinitely.

○ F) magic(Stream.of(5, 10));throws an exception.

○ G) The method does not compile.

1. **Assume that you have an InputStream that supports the mark operation whose next bytes are LEMUR. What would be the next char that could be read from the stream after calling the following method on the stream, using a count value of 3?**

**public** **static** String pullBytes(InputStream is, **int** count) **throws** IOException {

**final** StringBuffer sb = **new** StringBuffer();

is.skip(2);

**if**(is.markSupported()) {

is.mark(1);

**for**(**int** i=0; i<count; i++)

sb.append((**char**)is.read());

is.reset();

}

sb.append(is.read()+is.read());

**return** sb.toString();

}

○ A) L

○ B) E

○ C) M

○ D) U

○ E) R

○ F) The code does not compile.

○ G) The code compiles but throws an exception at runtime.

○ H) The result cannot be determined with the information given.

1. **Which of the following can be inserted to overload the toddle() method? (Choose all that apply.)**

**public** **class** LearnToWalk {

**public** **void** toddle() {}

**class** BabyRhino **extends** LearnToWalk {

// INSERT CODE HERE

}

}

○ A) private void toddle(boolean fall) {}

○ B) private void Toddle(boolean fall) {}

○ C) public void toddle() {}

○ D) public void Toddle() {}

○ E) public void toddle(boolean fall) {}

○ F) public void Toddle(boolean fall) {}

1. **What is the output of the following?**

**public** **class** SnowStorm {

**static** **class** Walk **implements** AutoCloseable {

**public** **void** close() {

**throw** **new** RuntimeException("snow");

}

}

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

**try** (Walk walk1 = **new** Walk(); Walk walk2 = **new** Walk();) {

} **catch** (Exception e) {

System.***out***.println(e.getMessage() + " "+ e.getSuppressed().length);

} } }

○ A) rain 0

○ B) rain 1

○ C) rain 2

○ D) show 0

○ E) snow 1

○ F) snow 3

○ G) The code does not compile.

1. **What is the result of the following code?**

**public** **class** FourLegged {

String walk = "walk,";

**public** **void** walk(){

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

}

**static** **class** BabyRhino **extends** FourLegged {

@Override String walk = "toddle,";

@Override **public** **void** walk() {

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

}

}

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

FourLegged f = **new** BabyRhino();

BabyRhino b = **new** BabyRhino();

f.walk();

b.walk();

}

}

○ A) toddle, toddle,

○ B) toddle, walk,

○ C) walk, toddle,

○ D) walk, walk,

○ E) The code does not compile.

○ F) A runtime exception is thrown.

1. **Which of the following statements are true about the following code?(Choose all that apply.)**

**public** **class** Ticket {

**private** Stub stub = **new** Stub();

**private** **class** Stub {

**public** **int** number;

**public** String venue;

}

**public** Ticket(**int** number, String venue) {

stub.number = number;

stub.venue = venue;

}

@Override **public** **int** hashCode() {

**return** stub.number;

}

**public** **boolean** equals(Object other) {

**if**(!(other **instanceof** Ticket)) **return** **false**;

Ticket t = (Ticket) other;

**return** stub.number == t.stub.number && stub.venue.equals(t.stub.venue);

}

}

○ A) The Ticket class compiles successfully.

○ B) The equals() method contains a compiler error.

○ C) There is a compiler error related to the Stub class.

○ D) The equals() method is incorrect.

○ E) The hashCode() method is incorrect.

○ F) The equals() method overrides the one in Object.

1. **Which of the following prints out all of the keys in props?**

○ A) props.keys().stream().map(k -> k.forEach(System.out::println);

○ B) props.keys().stream().map(k -> props.get(k)).forEach(System.out::println);

○ C) props.keySet().stream().map(k -> k).forEach(System.out::println);

○ D) props.keySet().stream().map(k -> props.get(k)).forEach(System.out::println);

○ E) props.stream().map(k -> k).forEach(System.out::println);

○ F) props.stream().map(k -> props.get(k)).forEach(System.out::println);

1. **What statements about the following code snippet are true?(Choose all that apply.)**

8: Stream<String> primates = Stream.*of*("gorilla","lemur","monkey","orangutan");

9: Stream<String> turtles = Stream.*of*("leatherback","green","loggerhead");

10: ConcurrentMap<Boolean, List<String>> data = Stream.*of*(primates, turtles)

11: .flatMap(s -> s).parallel()

12: .collect(Collectors.*groupingByConcurrent*(s -> s.endsWith("n")));

13: System.***out***.println(data.get(**false**).size()+" "+data.get(**true**).size());

○ A) It outputs 2 5.

○ B) It outputs 5 2.

○ C) The code will not compile because of line 10.

○ D) The code will not compile because of line 11.

○ E) The code will not compile because of line 12.

○ F) It compiles but throw an exception at runtime.

○ G) The collect() operation is always executed in a single-threaded fashion.

1. **What is the result of executing the following application? (Choose all that apply.)**

**import** java.util.concurrent.CyclicBarrier;

**import** java.util.stream.\*;

**public** **class** RabbitPenManager {

**public** **static** **void** await(CyclicBarrier cb) { //k1

**try** { cb.await(); } **catch** (InterruptedException | BrokenBarrierException e) {

// Handle exception

}

}

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

CyclicBarrier cb = **new** CyclicBarrier(2,

() -> System.***out***.println("Pen is Full")); // k2

IntStream.*iterate*(1, i -> 1)

.limit(2).forEach(i -> *await*(cb)); //k3

}

}

○ A) It outputs Pen is Full exactly once.

○ B) It outputs Pen is Full multiple times.

○ C) The code will not compile because of line k1.

○ D) The code will not compile because of line k2.

○ E) The code will not compile because of line k3.

○ F) It compiles but throws an exception at runtime.

○ G) It compiles but waits forever at runtime.

1. **What is the result of the following?**

List<Integer> l = IntStream.*rangeClosed*(1, 5)

.mapToObj(i -> i).collect(Collectors.*toList*());

l.forEach(System.***out***::print);

○ A) 1234

○ B) 12345

○ C) There is no output.

○ D) The code does not compile.

○ E) An exception is thrown.

1. **How do you obtain a connection through JDBC?**

○ A) new Connection(url)

○ B) new ConnectionImpl(url)

○ C) Driver.getConnection(url)

○ D) new Drver().getConnection(url)

○ E) DriverManager.getConnection(url)

○ F) new Drivermanager().getConnection(url)

1. **Which of the following statements can be inserted in the blank so that the code will compile successfully? (Choose all that apply.)**

**public** **interface** WalksOn4Legs {}

**public** **abstract** **class** Mammal {

**public** **int** numberOfOffspring;

}

**public** **class** Antelope **extends** Mammal **implements** WalksOn4Legs {}

**public** **class** ParkRanger{

**public** **void** noteNewOffspring(Mammal mammal) {

mammal.numberOfOffspring++;

}

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

**new** ParkRanger().noteNewOffspring(\_\_\_\_\_\_\_\_\_);

}

}

○ A) new Mammal()

○ B) new Antelope()

○ C) new WalksOn4Legs()

○ D) (Mammal) new Object()

○ E) (Mammal) new String()

○ F) null

1. **Assuming 100 milliseconds is enough time for the tasks submitted to the thread executor to complete, what is the result of executing the following code snippet? (Choose all that apply.)**

AtomicInteger lion1 = **new** AtomicInteger(0);

**final** **int**[] lion2 = {0};

ExecutorService service = Executors.*newCachedThreadPool*(); //h1

**for**(**int** i = 0; i<1000; i++)

service.submit(() -> lion1++); //h2

**for**(**int** i = 0; i<1000; i++)

service.submit(() -> lion2[0]++); //h3

Thread.*sleep*(100);

System.***out***.println(lion1 + " "+ lion2);

○ A) It outputs 1000 999.

○ B) It outputs 1000 100.

○ C) The output cannot be determined ahead of time.

○ D) The code will not compile because of line h1.

○ E) The code will not compile because of line h2.

○ F) The code will not compile because of line h3.

○ G) It compiles but throws an exception at runtime.

1. **Which of the following are valid lambda expressions? (Choose all that apply.)**

○ A) a,b -> a

○ B) (Dolphin d, Fish f) -> {return new Object()}

○ C) Kangaroo w -> new Kangaroo()

○ D) (Cat c, d) -> {return 10;}

○ E) (Whale w) -> {int w = 3; return w;}

○ F) (int o) -> ""

○ G) () -> 10

1. **Which of the following can fill in the blank to make the code compile?(Choose all that apply.)**

**import** java.util.stream.IntStream;

**public** **class** SwanBoat {

\_\_\_\_\_\_\_\_ SwanBoat;

**public** **void** row() { }

**public** **void** paddle() {

IntStream.*range*(1,10).forEach(i -> boat.row());

}

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

boat = **new** SwanBoat();

}

}

○ A) final

○ B) public

○ C) public final

○ D) public final static

○ E) public static

○ F) static final

1. **What is the result of the following code?**

**public** **class** Sorting {

**static** **class** Lizard **implements** Comparable <Lizard> {

**int** weight;

Lizard(**int** w) { weight = w; }

**public** **int** compareTo(Lizard l) {

**return** weight-l.weight;

}

**public** String toString() { **return** "" + weight; }

}

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

List<Lizard> list = **new** ArrayList<>();

list.add(**new** Lizard(5));

list.add(**new** Lizard(4));

list.add(**new** Lizard(7));

Collections.*sort*(list);

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

}

}

○ A) [4, 5, 7]

○ B) [5, 4, 7]

○ C) [7, 5, 4]

○ D) The code does not compile.

○ E) A runtime exception is thrown.

1. **Which line is the first with a compiler error?**

List letters = Arrays.*asList*('a', 'b', 'c');

letters.stream() //c1

.sorted() //c2

.distinct() //c3

.forEach(System.***out***::println) //c4

.sorted(); //c5

○ A) c1

○ B) c2

○ C) c3

○ D) c4

○ E) c5

○ F) None of the above. The code compiles.

1. **Which of the following can fill in the blank so that the code prints out false? (Choose all that apply.)**

Stream<String> s = Stream.*generate*(() -> "");

**boolean** match = s.\_\_\_\_\_\_\_(String::isEmpty);

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

○ A) allMatch

○ B) anyMatch

○ C) findAny

○ D) findFirst

○ E) noneMatch

○ F) None of the above

1. **Which of following can fill in the blank to make the method successfully compile?(Choose all that apply.)**

**public** **static** **void** add(\_\_\_\_) {

list.add(123);

}

○ A) List list

○ B) List<?> list

○ C) List<Number> list

○ D) List<? Extends Object> list

○ E) List<? Super Number> list

○ F) List<? Implements Number> list

1. **Which of the following changes must be made in order for the following code to print out2?(Choose all that apply.)**

**import** java.util.\*;

**public** **class** Panda {

String name;

Panda(String name) { **this**.name = name; }

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

Set s = **new** TreeSet<>();

s.add(**new** Panda("Bao Bao"));

s.add(**new** Panda("Mei Xiang"));

s.add(**new** Panda("Bao Bao"));

System.***out***.println(s.size());

}

}

○ A) Have class implement Comparable

○ B) Have class implement Comparator

○ C) Override compare()

○ D) Override compareTo()

○ E) Override hashCode()

○ F) None of the above; it is impossible to output 2.

1. **Assuming the directory / capybara/ food exists, is accessible within the file system, and has at least one regular file in it, what are the possible results of executing the following code? (Choose all that apply.)**

**import** java.nio.file.\*;

**public** **class** DirStream {

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

Path one = Paths.*get*("/capybara/food","apples.txt");

Path two = Paths.*get*("/capybara/food","bananas.txt");

Files.*copy*(one,two);

}

}

○ A) The file is copied.

○ B) The code runs without printing anything.

○ C) The code does not compile.

○ D) The code compiles but throws an exception at runtime.

1. **What is the result of the following statements?**

Queue<Integer> q = **new** LinkedList<>();

q.add(**new** Integer(6));

q.add(**new** Integer(6));

System.***out***.println(q.size() + " " + q.contains(6L));

○ A) 1 false

○ B) 1 true

○ C) 2 false

○ D) 2 true

○ E) The code does not compile.

○ F) A runtime exception is thrown.

1. **What is the result of executing the following program? (Choose all that apply.)**

**import** java.util.concurrent.\*;

**public** **class** MonkeyCounter {

**private** **static** AtomicInteger *monkey1* = **new** AtomicInteger(); //m1

**private** **static** AtomicLong *monkey2* = **new** AtomicLong(0);

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

ExecutorService service = **null**;

**try** {

service = Executors.*newSingleThreadExecutor*(); //m2

**for**(**int** i = 0; i < 100; i++)

service.submit(() -> *monkey1*.getAndIncrement()); //m3

**for**(**int** i = 0; i < 100; i++)

service.submit(() -> *monkey2*.incrementAndGet());

System.***out***.println(*monkey1* + " " + *monkey2*); //m4

} **finally** {

**if**(service != **null**) service.shutdown();

}

}

}

○ A) It outputs 100 99.

○ B) It outputs 100 100.

○ C) The output cannot be determined ahead of time.

○ D) The code will not compile because of line m1.

○ E) The code will not compile because of line m2.

○ F) The code will not compile because of line m3.

○ G) The code will not compile because of line m4.

○ H) It compiles but throws an exception at runtime.

1. **Assuming an implementation of the calculateFuture() method is provided prior to runtime, which of the following are possible results of executing the following application? (Choose all that apply.)**

**import** java.util.\*;

**import** java.util.concurrent.\*;

**import** java.util.stream.\*;

**public** **class** ZooPredictor {

**public** **static** **void** calculateFuture(**int** seed) {

//IMPLEMENTATION OMITTED

}

**public** **static** **void** seeFuture(Future<?> f) {

**try** {

System.***out***.println(f.get(100, TimeUnit.***DAYS***)); //i1

} **catch** (Exception e) {

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

}

}

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

ExecutorService service = Executors.*newSingleThreadExecutor*();

**final** List<Future<?>> results = **new** ArrayList<>();

IntStream.*range*(10, 15)

.forEach(i -> results.add(

service.submit(() -> *calculateFuture*(i)))); //i2

results.stream().forEach(f -> *seeFuture*(f));

service.shutdown();

}

}

○ A) It outputs a number five times.

○ B) It outputs a String value five times.

○ C) It outputs a null value five times.

○ D) It outputs Problem five times.

○ E) It hangs indefinitely at runtime.

○ F) It throws an unhandled exception at runtime.

○ G) The code will not compile because of line i1.

○ H) The code will not compile because of line i2.

1. **What is the result of the following code?**

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

List list = **new** ArrayList();

list.add(1);

list.add("2");

list.add(.3);

*print*(list);

}

**private** **static** **void** print(List<String> list) {

**for** (Object object: list)

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

}

○ A) 1 followed by an exception.

○ B) 12 followed by an exception.

○ C) 120.3

○ D) The output is indeterminate.

○ E) The code does not compile.

○ F) An exception is thrown.

1. **Which of the following statements compile?(Choose all that apply.)**

○ A) java.util.List<? extends java.sql.Statement> list1 = new java.util.ArrayList<>();

○ B) java.util.List<? implements java.sql.Statement> list2 = new java.util.ArrayList<>();

○ C) java.util.List<? extends java.sql.Statement> list3 = new java.util.ArrayList()<>;

○ D) java.util.List<? implements java.sql.Statement> list4 = new java.util.ArrayList()<>;

○ E) java.util.List<? extends java.sql.Statement> list5 = new java.util.ArrayList();

○ F) java.util.List<? extends java.sql.Statement> list6 = new java.util.List();

1. **Suppose the directory c:\text exists within the file system but no subdirectories of it exist. Which line of code would be the best way to create the directory c:\text\book\java?**

○ A) File.mkdir("c:\\text\\book\\java");

○ B) File.mkdirs("c:\\text\\book\\java");

○ C) new File("c:\\text\\book\\java").mkdir();

○ D) new File("c:\\text\\book\\java").mkdirs();

○ E) None of the above

1. **There are currently 100 rows in the table species before inserting a new row. What is the output of the following code?**

**try** (Connection conn = DriverManager.*getConnection*("jdbc:derby:zoo");

Statement stmt = conn.createStatement()) {

ResultSet rs = stmt.executeQuery("select count(\*) from species");

**int** num = stmt.executeUpdate("INSERT INTO species VALUES (3,'Ant',

.05)");

rs = stmt.executeQuery("select count(\*) from species");

System.***out***.println(rs.getInt(1));

}

○ A) 100

○ B) 101

○ C) The code does not compile.

○ D) A SQLException is thrown.

○ E) A different exception is thrown.

1. **What is the result of the following code?(Choose all that apply.)**

1: **public** **abstract** **interface** CanSwim {

2: **int** ***MAX\_DISTANCE*** = 10;

3: **abstract** **void** swim();

4: **default** **static** **boolean** hasFlippers() { **return** **false**; }

5: **public** **static** **int** getMaxDistance() { **return** ***MAX\_DISTANCE***; }

6: **public** **final** **static** **void** checkSwimmer() {}

7: }

○ A) The code compiles without issue.

○ B) The code will not compile because of line 2.

○ C) The code will not compile because of line 3.

○ D) The code will not compile because of line 4.

○ E) The code will not compile because of line 5.

○ F) The code will not compile because of line 6.

1. **Assuming that the file /giraffe/food.csv exists within the file systemand that it is not empty, what is the result of executing the following code? (Choose all that apply.)**

Path path = Paths.*get*("/graffe/food.csv");

Files.*lines*(path)

.flatMap(p -> Stream.of(p.split(","))) //j1

.map(s -> s.length()) // j2

.collect(Collectors.*toList*())

.forEach(System.***out***::print); //j3

○ A) The code compiles but does not produce any output at runtime.

○ B) The code will not compile because of line j1.

○ C) The code will not compile because of line j2.

○ D) The code will not compile because of line j3.

○ E) The code prints the contents of the file without any commas.

○ F) The code prints a series of numbers.

○ G) The code produces an infinite loop at runtime.

1. **Given the following line of code, which of the following three statements must be true prior to execution for it to run without throwing a runtime exception?**

Files.move(path1,path2,StandardCopyOption.REPLACE\_EXISTING);

○ A) path1 refers to a file or directory that exist within the file syatem.

○ B) path1 refers to a file and not a directory.

○ C) path2 refers to a file or directory that does not exist within the fie system.

○ D) If path2 refers to a directory that exists within the file system, then it must be empty.

○ E) path1 refers to a symbolic link.

○ F) The file system is available and path objects are not hidden by the file system.

1. **Assuming that partrons.txt exists and it is readily accessible by the file system, what is the output of the following code?(Choose all that apply.)**

File file = **new** File("patrons.txt");

Path path = file.toPath().normalize(); //j1

Path.lines(path).map(s -> "Hello " + s).forEach(System.out::println); // j2

○ A) It prints nothing at runtime.

○ B) It prints one line for each line of the file at runtime.

○ C) The code will not compile because of line j1.

○ D) The code will not compile because of line j2.

○ E) It compiles but throws an exception at runtime.

1. **There are currently 100 rows in the table species before inserting a new row. What is the output of the following code?**

**try** (Connection conn = DriverManager.*getConnection*("jdbc:derby:zoo");

Statement stmt = conn.createStatement()) {

ResultSet rs = stmt.executeQuery("select count(\*) from species");

rs.next();

**int** num = stmt.executeUpdate("INSERT INTO species VALUES (3, 'Ant', .05)");

rs = stmt.executeQuery("select count(\*) from species");

rs.next();

System.***out***.println(rs.getInt(1));

}

○ A) 100

○ B) 101

○ C) The code does not compile.

○ D) A SQLException is thrown.

○ E) A different exception is thrown.

1. **Which option fills in the blank to make this code compile?**

Stream<String> s = Stream.*of*("bull","cow");

\_\_\_\_\_\_\_ r= s.collect(Colloctors.groupingBy(String::length));

○ A) List<String>

○ B) Map<Boolean, List<String>>

○ C) Map<Integer, List<String>>

○ D) Map<String, List<String>>

○ E) None of the above

1. **Assuming that the current directory is /mammals and the directories and files referenced by the program exist, what is true about the following code snippet?(Choose all that apply.)**

Path path = Paths.*get*("bear/polar/./environment").normalize().getRoot(); // w1

System.***out***.println(Files.*list*(path)

.filter(p -> !Files.*isDirectory*(p)) // w2

.map(p -> p) //w3

.collect(Collectors.*toSet*())

.size());

○ A) The code will not compile because of line w1.

○ B) The code will not compile because of line w2.

○ C) The code will not compile because of line w3.

○ D) The code prints the number of unique files in a directory.

○ E) The code prints the number of unique files in a directory tree.

○ F) The code compiles without issue but throws an exception an runtime.

1. **Which is a true statement about the following code? (Choose all that apply.)**

**public** **class** WhatisIt {

**interface** Furry { }

**class** Chipmunk { }

**class** FurryChipmunk **implements** Furry { }

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

Chipmunk c = **new** Chipmunk();

**int** result = 0;

**if**(c **instanceof** Furry) result += 1;

**if**(c **instanceof** Chipmunk) result += 2;

**if**( **null** **instanceof** FurryChipmunk) result +=4;

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

}

}

○ A) The code compiles, and the output is 1.

○ B) The code compiles, and the output is 2.

○ C) The code compiles, and the output is 7.

○ D) The code does not compile due to an instanceof statement.

○ E) The code does not compile for another reason.

○ F) A runtime exception is thrown.

1. **What is the result of executing the following code?(Choose all that apply.)**

String line;

Console c = System.*console*();

Writer w = c.writer();

**if**((line = c.readLine()) != **null**)

w.append(line);

○ A) The code runs without error but prints nothing.

○ B) The code prints what the user entered.

○ C) AnArryIndexOutOfBoundsException might be thrown.

○ D) A NullPointerException might be thrown.

○ E) An IOException might be thrown.

○ F) The code does not compile.

1. **What statements about the following code are true?(Choose all that apply.)**

Path path = Paths.*get*("/zoo/gorilla.txt");

**try** (BufferedReader reader = Files.*newBufferedReader*(path)) { //x1

String s = **null**;

**while**(reader.readLine() !=**null**) //x2

System.***out***.println(s); //x3

}

○ A) If the file exists and is non-empty,it is capable of printing all lines of the file.

○ B) The code will not compile because of line x1.

○ C) The code will not compile because of line x2.

○ D) The code will not compile because of line x3.

○ E) It compiles but may result in an infinite loop at runtime.

○ F) It compiles but may throw an exception at runtime.

1. **Which option fills in the blank to make this code compile?**

Stream<String> s = Stream.*of*("bull","cow");

\_\_\_\_\_\_\_ r= s.collect(Colloctors.partitioningBy(b -> b.startsWith("c")));

○ A) List<String>

○ B) Map<Boolean, List<String>>

○ C) Map<String, List<String>>

○ D) Map<Object, List<String>>

○ E) None of the above

1. **Which of the following can fill in the blanks in order to make this code compile?**

\_\_\_\_\_\_ a = \_\_\_\_\_.getConnection(url, userName, password);

\_\_\_\_\_\_ b = a.createStatement();

\_\_\_\_\_\_ c = b.executeQuery(sql);

**if** (c.next()) System.***out***.println(c.getString(1));

○ A) Connection,Driver,Statement,ResultSet

○ B) Connection,DriverManager,Statement,ResultSet

○ C) Connection,DataSource,Statement,ResultSet

○ D) Driver,Connection,Statement,ResultSet

○ E) DriverManager,Connection,Statement,ResultSet

○ F) DataSource,Connection,Statement,ResultSet

1. **What is the result of calling the following method?**

**public** **static** **void** printData(BlockingDeque<Boolean> deque) **throws** InterruptedException {

deque.offerLast(**true**, 4, TimeUnit.***MICROSECONDS***);

deque.offer(**true**);

deque.offerFirst(**false**, 25, TimeUnit.***SECONDS***);

System.***out***.print(deque.pollFirst(5, TimeUnit.***DAYS***));

System.***out***.print(" "+deque.pollLast(2, TimeUnit.***NANOSECONDS***));

}

○ A) It outputs true false.

○ B) It outputs false true.

○ C) It outputs true true.

○ D) It outputs false false.

○ E) The code will not compile.

○ F) It compiles but throws an exception at runtime.

○ G) The output cannot be determined ahead of time.

1. **Which of the following can fill in the blank correctly?(Choose all that apply.)**

ResultSet rs = stmt.executeQuery("select name from animal");

**if** (rs.next()) {

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

}

○ A) rs.getString(0);

○ B) rs.getString(1);

○ C) rs.getString("name");

○ D) None of the above-the code does not compile.

1. **Which of the following can fill in the blank to print just the hour, minutes, and seconds?**

ResultSet rs = stmt.executeQuery(sql);

**if**(rs.next()) {

System.***out***.println(\_\_\_\_\_\_\_\_\_\_\_);

}

○ A) rs.getDate("d");

○ B) rs.getLocalDate("d");

○ C) rs.getLocalDateTime("d");

○ D) rs.getLocalTime("d");

○ E) rs.getTime("d");

○ F) rs.getTimeStamp("d");

1. **Suppose that we have the following property files and code. Which bundle is used on lines 8 and 9?**

**Dolphins.properties**

name=The Dolphin

age=0

**Dolphins\_de.properties**

name=Dolly

age=4

**Dolphins\_en.properties**

name=Dolly

5: Locale fr = **new** Locale("fr");

6: Locale.*setDefault*(**new** Locale("en","US"));

7: ResourceBundle b = ResourceBundle.*getBundle*("Dolphins",fr);

8: b.getString("name");

9: b.getString("age");

○ A) Dolphins.properties and Dolphins.properties

○ B) Dolphins.properties and Dolphins\_en.properties

○ C) Dolphins\_en.properties and Dolphins.properties

○ D) Dolphins\_en.properties and Dolphins\_en.properties

○ E) Dolphins\_de.properties and Dolphins\_en.properties

○ F) The code does not compile.

1. **Assuming that the path values referenced here exist and are accessible within the file system, what is the result of executing the following code?(Choose all that apply.)**

Path path = Paths.*get*("/storage","toys").resolve("bird"); // n1

Files.*find*(path,(p,a) -> a.isDirectory()) // n2

.map(p -> p.toRealPath(LinkOption.NOFOLLOW\_LINKS).toString()) // n3

.map(p -> p.normalize()) // n4

.forEach(System.***out***::print);

○ A) The code will not compile because of line n1.

○ B) The code will not compile because of line n2.

○ C) The code will not compile because of line n3.

○ D) The code will not compile because of line n4.

○ E) The code compiles but may not produce any output at runtime.

○ F) The code compiles and may print a list of symbolic links at runtime.

1. **Which of the following statements are true given that a and b are of type Integer?(Choose all that apply.)**

○ A) If a.hashCode() == b.hashCode() is true, a.equals(b) is always true.

○ B) If a.hashCode() == b.hashCode() is true, a.equals(b) is sometimes but not always true.

○ C) If a.hashCode() == b.hashCode() is false, a.equals(b) is never true.

○ D) If a.hashCode() == b.hashCode() is false, a.equals(b) is sometimes true.

1. **What are some reasons to use a byte stream, such as InputStream/OutputStream, over a character stream, such as Reader/Writer?(Choose all that apply.)**

○ A) Improved performance.

○ B) More convenient code syntax when working with String data.

○ C) Built-in serialization and deserialization.

○ D) Reading and writing binary data.

○ E) Multi-threading support.

○ F) Byte streams are low-level streams.

1. **What statements about the following class definition are true? (Choose all that apply.)**

**public** **class** LandscapeTracker {

**private** **transient** **int** flowers;

**public** **int** getFlowers() { **return** flowers; }

**public** **synchronized** **void** addFlowers(**int** x) { // q1

flowers += x;

}

**public** **void** removeFlowers(**int** x) {

**synchronized** (**this**) { // q2

flowers -= x;

}

}

}

○ A) It compiles without issue.

○ B) The transient modifier prevents the code from being accessed by multiple threads.

○ C) This class implements the immutable object pattern.

○ D) The lock acquired on q1 and q2 is on the same object.

○ E) The class correctly prevents concurrency issues for the value of flowers when accessed by multiple threads.

○ F) The variable flowers would benefit from being changed to an AtomicInteger.

1. **Which of following is the best way to convert a stream of String objects to a stream of int primitives?**

○ A) IntStream is = stream.map(Integer::parseInt);

○ B) IntStream is = stream.mapToInt(Integer::parseInt);

○ C) Stream<Integer> is = stream.mapToInt(Integer::parseInt);

○ D) Stream<Integer> is = stream.mapToInt(Integer::parseInt);

○ E) None of the above. Primitives cannot be used in any type of stream.

1. **Which class is used to represent a directory in java.io?**

○ A) Dir

○ B) Directory

○ C) File

○ D) Files

○ E) Path

1. **What is the result of the following code?**

**public** **class** Sorting {

**static** **class** Lizard **implements** Comparable <Lizard> {

**int** weight;

Lizard(**int** w) { weight = w; }

**public** **int** compare(Lizard l, Lizard m) {

**return** l.weight-m.weight;

}

**public** String toString() { **return** "" + weight; }

}

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

List<Lizard> list = **new** ArrayList<>();

list.add(**new** Lizard(5));

list.add(**new** Lizard(4));

list.add(**new** Lizard(7));

Collections.*sort*(list);

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

}

}

○ A) [4, 5, 7]

○ B) [5, 4, 7]

○ C) [7, 5, 4]

○ D) The code does not compile.

○ E) A runtime exception is thrown.

1. **November 6,2016, is the date that clocks fall back from daylight saving time. What is the output of the following?**

LocalDate date = LocalDate.*of*(2016, Month.***NOVEMBER***, 6);

LocalTime time = LocalTime.*of*(1, 30);

ZoneId zone = ZoneId.*of*("US/Eastern");

ZonedDateTime dateTime1 = ZonedDateTime.*of*(date, time, zone);

ZonedDateTime dateTime2 = dateTime1.plus(1,ChronoUnit.***HOURS***);

**long** hours = ChronoUnit.***HOURS***.between(dateTime1, dateTime2);

**int** clock1 = dateTime1.getHour();

**int** clock2 = dateTime2.getHour();

System.***out***.println(hours + "," + clock1 + "," + clock2);

○ A) 0,1,1

○ B) 0,1,2

○ C) 1,1,1

○ D) 1,1,2

○ E) The code does not compile.

○ F) A runtime exception is thrown.

1. **Which of the following are true? (Choose all that apply.)**

List<Integer> l1 = Arrays.*asList*();

List<Integer> l2 = Arrays.*asList*(1, 2, 3);

List<Integer> l3 = Arrays.*asList*(4, 5, 6);

Stream.*of*(l1, l2, l3)

.limit(2)

.peek(System.***out***::println) //peek 1

.flatMap(x -> x.stream())

.peek(System.***out***::println) // peek 2

.map(x -> x + 1);

.forEach(System.out::println);

○ A) Zero lines are generated by the line marked by peek 1.

○ B) Two lines are generated by the line marked by peek 1.

○ C) Three lines are generated by the line marked by peek 1.

○ D) Zero lines are generated by the line marked by peek 2.

○ E) Two lines are generated by the line marked by peek 2.

○ F) Three lines are generated by the line marked by peek 2.

○ G) The code does not compile.

○ H) The code throws an exception.

1. **Which of the following is printed out by the following code?**

String d = Duration.*ofDays*(1);

String p = Period.*ofDays*(1);

**boolean** b1 = d == p;

**boolean** b2 = d.equals(p);

System.***out***.println(b1 + " " + b2);

○ A) false false

○ B) false true

○ C) true false

○ D) true true

○ E) The code does not compile.

○ F) A runtime exception is thrown.

1. **Suppose the table animal has five rows and this SQL statement updates all of them.What is the result of this code?**

Connection conn = DriverManager.*getConnection*("jdbc:derby:zoo");

Statement stmt = conn.createStatement();

**int** result = stmt.executeQuery("update animal set name = name");

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

○ A) 0

○ B) 1

○ C) 5

○ D) The code does not compile.

○ E) A SQLException is thrown.

○ F) A different exception is thrown.

1. **What is the result of the following code?**

**public** **class** Employee {

**public** **int** employeeId;

**public** String firstName, lastName;

**public** java.time.LocalDate hireDate;

@Override **public** **int** hashCode(**int** id) {

**return** employeeId;

}

**public** **boolean** equals(Employee e) {

**return** **this**.employeeId == e.employeeId;

}

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

Employee one = **new** Employee();

one.employeeId = 101;

Employee two = **new** Employee();

two.employeeId = 101;

**if**(one.equals(two)) System.***out***.println("Success");

**else** System.***out***.println("Failure");

}

// imagine getters and setters are here

}

○ A) Success

○ B) Failure

○ C) The hashCode() method fails to compile.

○ D) The equals() method fails to compile.

○ E) Another line of code fails to compile.

○ F) A runtime exception is thrown.

1. **Which are true statements?(Choose all that apply.)**

○ A) findAny() and findFirst() are guaranteed to return the same if the stream is empty.

○ B) findAny() and findFirst() are guaranteed to return the same if the stream has exactly one element.

○ C) findAny() and findFirst() are guaranteed to return the same if the stream has exactly two element.

○ D) findAny() returns a boolean.

○ E) findFirst() returns a boolean.

1. **What is the result of the following class?**

**public** **class** C<A> {

A a;

**public** **void** m(A a){

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

}

**public** **void** m(Object o) {

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

}

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

C<String> c = **new** C<>();

c.m("s");

} }

○ A) a

○ B) obj

○ C) s

○ D) The code does not compile.

○ E) A runtime exception is thrown.

1. **Which classes can be inserted in the blank to make this code compile? (Choose all that apply.)**

**import** \_\_\_\_\_\_\_\_\_\_\_\_\_\_;

**public** **class** StartOfSummer {

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

LocalDate date = LocalDate.*of*(2014, 6, 21);

} }

○ A) java.date.\*

○ B) java.date.LocalDate

○ C) java.time.\*

○ D) java.time.LocalDate

○ E) java.util.\*

○ F) java.util.LocalDate

1. **Assuming that an implementation of the trackParticipants() method is provided prior to runtime, which of the following are possible results of executing this application?(Choose all that apply.)**

**import** java.util.\*;

**import** java.util.concurrent.\*;

**import** java.util.stream.\*;

**public** **class** ContestApp {

**public** **static** Object trackParticipants(**int** count) {

// IMPLEMENTATION OMITTED

}

**public** **static** **void** viewWinners(Future<?> f) {

**try** {

System.***out***.println(f.get(100)); //a1

} **catch** (Exception e) {

System.***out***.println("No Winner");

}

}

**public** **static** **void** main(String[] args) **throws** InterruptedException, ExecutionException {

ExecutorService = Executors.*newSingleThreadExecutor*();

**final** List<Future<?>> results = **new** ArrayList<Future<?>>();

IntStream.*range*(100,105)

.forEach(i -> results.add(

service.submit(p -> trackParticipants(i)))); //a2

results.stream().forEach(f -> *viewWinners*(f));

service.shutdown();

}

}

○ A) It outputs a number five times.

○ B) It outputs a Boolean value five times.

○ C) It outputs a null value five times.

○ D) It outputs No Winner five times.

○ E) It hangs indefinitely at runtime.

○ F) It throws an unhandled exception at runtime.

○ G) The code will not compile because of line a1.

○ H) The code will not compile because of line a2.

1. **Which of the following fill in the blank to make the code compile? (Choose all that apply.)**

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

**try** {

process();

}**catch** (\_\_\_\_\_\_\_\_\_\_\_) {}

}

**private** **static** **void** process() **throws** InterruptedException, SQLException {}

}

○ A) InterruptedException | SQLException e

○ B) InterruptedException e | SQLException e

○ C) InterruptedException e1 | SQLException e2

○ D) SQLException | InterruptedException e

○ E) SQLException e | InterruptedException e

○ F) SQLException e1 | InterruptedException e2

1. **What is the result of the following code when run using the command java -ea PickAColor?**

**public** **class** PickAColor {

**enum** Color { ***RED***, ***BLUE***, ***GREEN***}

**public** **static** **void** go(Color c) {

**switch** (c) {

**case** ***RED***: System.***out***.println("red");

**case** ***BLUE***: System.***out***.println("blue"); **break**;

**case** ***GREEN***: System.***out***.println("green");

**default**: **assert** **false**;

}

}

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

*go*(Color.***RED***);

}

}

○ A) red

○ B) redblue

○ C) redblue, followed by an AssertionError

○ D) redbluegreen, followed by an AssertionError

○ E) The code does not compile.