Chapter 14
Lambda Built-in Functional Interfaces

THE OCP EXAM TOPICS COVERED IN THIS PRACTICE TEST INCLUDE THE FOLLOWING:

✓**Lambda Built-in Functional Interfaces**

- Use the built-in interfaces included in the java.util.function package such as Predicate, Consumer, Function, and Supplier
- Develop code that uses primitive versions of functional interfaces
- Develop code that uses binary versions of functional interfaces
- Develop code that uses the UnaryOperator interface

1. Fill in the blanks: The______________ functional interface does not take any inputs, while the______________ functional interface does not return any data.

   A. `IntConsumer`, `LongSupplier`

   B. `IntSupplier`, `Function`

   C. `Supplier`, `DoubleConsumer`

   D. `UnaryOperator`, `Consumer`

2. Which functional interface takes a `long` value as an input argument and has an `accept()` method?

   A. `LongConsumer`

   B. `LongFunction`

   C. `LongPredicate`

   D. `LongSupplier`

3. What is the output of the following application?

```
package beach;
import java.util.function.*;

class Tourist {
   public Tourist(double distance) {
      this.distance = distance;
   }
   public double distance;
}
public class Lifeguard {
   private void saveLife(Predicate<Tourist> canSave, Tourist
tourist) {
      System.out.print(canSave.test(tourist) ? "Saved" : "Too
far");  // y1
   }
   public final static void main(String... sand) {
      new Lifeguard().saveLife(s -> s.distance<4, new
Tourist(2));  // y2
   }
}
```

A. `Saved`

B. `Too far`

C. The code does not compile because of line `y1`.

D. The code does not compile because of line `y2`.

4. Which of the following statements about `DoubleSupplier` and `Supplier<Double>` is not true?

A. Both are functional interfaces.

B. Lambdas for both can return a `double` value.

C. Lambdas for both cannot return a `null` value.

D. One supports a generic type, the other does not.

5. Which functional interface, when filled into the blank, allows the class to compile?

```
package space;
import java.util.function.*;

public class Asteroid {
   public void mine(____________ lambda) {
      // TODO: Apply functional interface
   }
   public static void main(String[] debris) {
      new Asteroid().mine((s,p) -> s+p);
   }
}
```

A. `BiConsumer<Integer,Double>`

B. `BiFunction<Integer,Double,Double>`

C. `BiFunction<Integer,Integer,Double>`

D. `Function<Integer,Double>`

6. Assuming the proper generic types are used, which lambda expression cannot be assigned to a `ToDoubleBiFunction` functional interface reference?

A. `(Integer a, Double b) -> {int c; return b;}`

B. `(h,i) -> (long)h`

C. `(String u, Object v) -> u.length()+v.length()`

D. `(x,y) -> {int z=2; return y/z;}`

7. Which of the following is not a functional interface in the `java.util.function` package?

   A. `BiPredicate`

   B. `DoubleUnaryOperator`

   C. `ObjectDoubleConsumer`

   D. `ToLongFunction`

8. What is the output of the following application?

```
package zoo;
import java.util.function.*;

public class TicketTaker {
   private static int AT_CAPACITY = 100;
   public int takeTicket(int currentCount,
IntUnaryOperator<Integer> counter) {
      return counter.applyAsInt(currentCount);
   }
   public static void main(String...theater) {
      final TicketTaker bob = new TicketTaker();
      final int oldCount = 50;
      final int newCount = bob.takeTicket(oldCount,t -> {
         if(t>AT_CAPACITY) {
            throw new RuntimeException("Sorry, max has been
reached");
         }
         return t+1;
      });
      System.out.print(newCount);
   }
}
```

   A. `51`

   B. The code does not compile because of lambda expression.

   C. The code does not compile for a different reason.

   D. The code compiles but prints an exception at runtime.

9. Which functional interface returns a primitive value?

   A. `BiPredicate`

   B. `CharSupplier`

C. `LongFunction`

D. `UnaryOperator`

10. Which functional interface, when entered into the blank below, allows the class to compile?

```
package groceries;
import java.util.*;
import java.util.function.*;

public class Market {
   private static void checkPrices(List<Double> prices,
          ____________scanner) {
      prices.forEach(scanner);
   }
   public static void main(String[] right) {
      List<Double> prices = Arrays.asList(1.2, 6.5, 3.0);
      checkPrices(prices,
            p -> {
               String result = p<5 ? "Correct" : "Too high";
               System.out.println(result);
            });
   }

}
```

A. `Consumer`

B. `DoubleConsumer`

C. `Supplier<Double>`

D. None of the above

11. Which of the following three functional interfaces is not equivalent to the other two?

A. `BiFunction<Double,Double,Double>`

B. `BinaryOperator<Double>`

C. `DoubleFunction<Double>`

D. None of the above. All three are equivalent.

12. Which lambda expression can be passed to the `magic()` method?

```
package show;
import java.util.function.*;
```

```
public class Magician {
   public void magic(BinaryOperator<Long> lambda) {
      lambda.apply(3L, 7L);
   }
}
```

A. `magic((a) -> a)`

B. `magic((b,w) -> (long)w.intValue())`

C. `magic((c,m) -> {long c=4; return c+m;})`

D. `magic((Integer d, Integer r) -> (Long)r+d)`

13. What is the output of the following program?

```
package ai;
import java.util.function.*;

public class Android {
   public void wakeUp(Supplier supplier) { // d1
      supplier.get();
   }
   public static void main(String... electricSheep) {
      Android data = new Android();
      data.wakeUp(() -> System.out.print("Program started!"));
// d2
   }
}
```

A. `Program started!`

B. The code does not compile because of line `d1` only.

C. The code does not compile because of line `d2` only.

D. The code does not compile because of both lines `d1` and `d2`.

14. Which statement about all `UnaryOperator` functional interfaces (generic and primitive) is correct?

A. The input type must be compatible with the return type.

B. Some of them take multiple arguments.

C. They each take a generic argument.

D. They each return a primitive value.

15. Starting with `DoubleConsumer` and going downward, fill in the missing values for the table.

Functional Interface	**# Parameters**
`DoubleConsumer`	
`IntFunction`	
`LongSupplier`	
`ObjDoubleConsumer`	

A. `0`, `1`, `1`, `1`

B. `0`, `2`, `1`, `2`

C. `1`, `1`, `0`, `2`

D. `1`, `1`, `0`, `1`

16. Starting with `DoubleConsumer` and going downward, fill in the values for the table. For the choices below, assume `R` is a generic type.

Functional Interface	**Return Type**
`DoubleConsumer`	
`IntFunction`	
`LongSupplier`	
`ObjDoubleConsumer`	

A. `double`, `R`, `long`, `R`

B. `R`, `int`, `long`, `R`

C. `void`, `int`, `R`, `void`

D. `void`, `R`, `long`, `void`

17. Fill in the blanks: In the `Collection` interface, the method `removeIf()` takes a______________ , while the method `forEach()` takes a______________ .

A. `Function`, `Function`

B. `Predicate`, `Consumer`

C. `Predicate`, `Function`

D. `Predicate`, `UnaryOperator`

18. What is the output of the following application?

```
package nesting;
import java.util.function.*;
```

```
public class Doll {
   private int layer;
   public Doll(int layer) {
      super();
      this.layer = layer;
   }
   public static void open(UnaryOperator<Doll> task, Doll doll)
{
      while((doll = task.accept(doll)) != null) {
         System.out.print("X");
      }
   }
   public static void main(String[] wood) {
      open(s -> {
         if(s.layer<=0) return null;
         else return new Doll(s.layer--);
      }, new Doll(5));
   }
}
```

A. `XXXXX`

B. The code does not compile because of the lambda expression.

C. The code does not compile for a different reason.

D. The code compiles but produces an infinite loop at runtime.

19. Which functional interface has a `get()` method?

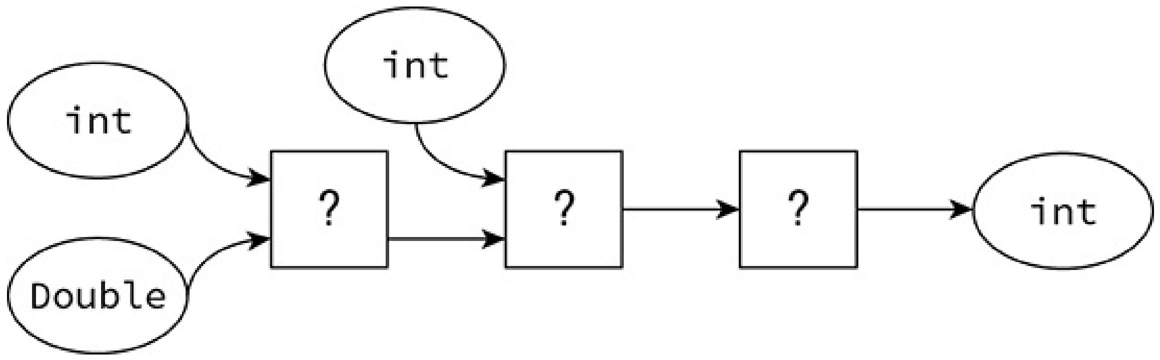
A. `Consumer`

B. `Function`

C. `Supplier`

D. `UnaryOperator`

20. The following diagram shows input arguments being applied to three functional interfaces of unknown type. Which three functional interfaces, inserted in order from left to right, could be used to complete the diagram?



A. `DoubleBinaryOperator`
`ToDoubleBiFunction<Integer,Double>`
`UnaryOperator<Integer>`

B. `BinaryOperator<Double>`
`BiFunction<Integer,Integer,Double>`
`UnaryOperator<Integer>`

C. `Function<Double,Integer>`
`BiFunction<Integer,Integer,Double>`
`DoubleToIntFunction`

D. `BiFunction<Integer,Double,Integer>`
`BinaryOperator<Integer>`
`IntUnaryOperator`

21. Which statement about functional interfaces and lambda expressions is not true?
    A. A lambda expression may be compatible with multiple functional interfaces.
    B. A lambda expression must be assigned to a functional interface when it is declared.
    C. A method can return a lambda expression in the form of a functional interface instance.
    D. The compiler uses deferred execution to skip determining whether a lambda expression compiles or not.

22. Which expression is compatible with the `IntSupplier` functional interface?

A. `() -> 1<10 ? "3" : 4`

B. `() -> {return 1/0;}`

C. `() -> return 4`

D. `System.out::print`

23. What is the output of the following application?

```
package tps;
import java.util.*;

class Boss {
   private String name;
   public Boss(String name) {
      this.name = name;
   }
   public String getName() {return name.toUpperCase();}
   public String toString() {return getName();}
}
public class Initech {
   public static void main(String[] reports) {
      final List<Boss> bosses = new ArrayList(8);
      bosses.add(new Boss("Jenny"));
      bosses.add(new Boss("Ted"));
      bosses.add(new Boss("Grace"));
      bosses.removeIf(s -> s.equalsIgnoreCase("ted"));
      System.out.print(bosses);
   }
}
```

A. `[JENNY, GRACE]`

B. `[tps.Boss@4218224c, tps.Boss@815f19a]`

C. The code does not compile because of the lambda expression.

D. The code does not compile for a different reason.

24. Which of the following method references can be passed to a method that takes `Consumer<Object>` as an argument?

I. `ArrayList::new`

II. `String::new`

III. `System.out::println`

A. I only

B. I, II, and III

C. I and III

D. III only

25. Which of the following is a valid functional interface in the `java.util.function` package?

A. `FloatPredicate`

B. `ToDoubleBiFunction`

C. `UnaryIntOperator`

D. `TriPredicate`

26. Which functional interface, when filled into the blank, prevents the class from compiling?

```
package morning;
import java.util.function.*;

public class Sun {
   public static void dawn(____________ sunrise) {}
   public void main(String... rays) {
      dawn(s -> s+1);
   }
}
```

A. `DoubleUnaryOperator`

B. `Function<String,String>`

C. `IntToLongFunction`

D. `UnaryOperator`

27. Which functional interface does not have the correct number of generic arguments?

A. `BiFunction<T,U,R>`

B. `DoubleFunction<T,R>`

C. `ToDoubleFunction<T>`

D. `ToIntBiFunction<T,U>`

28. Which lambda expression, when filled into the blank, allows the code to compile?

```
package ballroom;
import java.util.function.*;

public class Dance {
   public static Integer
rest(BiFunction<Integer,Double,Integer> takeABreak) {
      return takeABreak.apply(3, 10.2);
   }
   public static void main(String[] participants) {
      rest(____________);
   }
}
```

A. `(int n, double e) -> (int)(n+e)`

B. `(n,w,e) -> System.out::print`

C. `(s,w) -> 2*w`

D. `(s,e) -> s.intValue()+e.intValue()`

29. Fill in the blank: ________________is the only functional interface that does not involve `double`, `int`, or `long`.

A. `BooleanSupplier`

B. `CharPredicate`

C. `FloatUnaryOperator`

D. `ShortConsumer`

30. What is the output of the following application?

```
package savings;
import java.util.function.*;

public class Bank {
   private int savingsInCents;
   private static class ConvertToCents {
      static DoubleToIntFunction f = p -> p*100;
   }
   public static void main(String... currency) {
      Bank creditUnion = new Bank();
      creditUnion.savingsInCents = 100;
      double deposit = 1.5;

      creditUnion.savingsInCents +=
ConvertToCents.f.applyAsInt(deposit);  // j1
      System.out.print(creditUnion.savingsInCents);
   }
}
```

A. `200`

B. `250`

C. The code does not compile because of line `j1`.

D. None of the above

31. Which functional interface takes a `double` value and has a `test()` method?

A. `DoubleConsumer`

B. `DoublePredicate`

C. `DoubleUnaryOperator`

D. `ToDoubleFunction`

32. Given the following class, how many lines contain compilation errors?

```
1:  package showtimes;
2:  import java.util.*;
3:  import java.util.function.*;
4:  public class FindMovie {
5:     private Function<String> printer;
6:     protected FindMovie() {
7:         printer = s -> {System.out.println(s); return s;}
8:     }
9:     void printMovies(List<String> movies) {
10:       movies.forEach(printer);
11:    }
12:    public static void main(String[] screen) {
13:       List<String> movies = new ArrayList<>();
14:       movies.add("Stream 3");
15:       movies.add("Lord of the Recursion");
16:       movies.add("Silence of the Lambdas");
17:       new FindMovie().printMovies(movies);
18:    }
19: }
```

A. None. The code compiles as is.

B. One

C. Two

D. Three

33. Which lambda expression cannot be assigned to a `DoubleToLongFunction` functional interface?

   A. `a -> null==null ? 1 : 2L`

   B. `e -> (int)(10.0*e)`

   C. `(double m) -> {long p = (long)m; return p;}`

   D. `(Double s) -> s.longValue()`

34. Which of the following is not a functional interface in the `java.util.function` package?

   A. `DoublePredicate`

   B. `LongUnaryOperator`

   C. `ShortSupplier`

   D. `ToIntBiFunction`

35. Which functional interface, when filled into the blank, allows the class to compile?

```
package sleep;
import java.util.function.*;

class Sheep {}
public class Dream {
   int MAX_SHEEP = 10;
   int sheepCount;
   public void countSheep( ____________backToSleep) {
      while(sheepCount<MAX_SHEEP) {
         // TODO: Apply lambda
         sheepCount++;
      }
   }
   public static void main(String[] dark) {
      new Dream().countSheep(System.out::println);
   }
}
```

   A. `Consumer<Sheep>`

   B. `Function<Sheep,void>`

   C. `UnaryOperator<Sheep>`

   D. None of the above

36. What is the output of the following application?

```
package pet;
import java.util.*;
import java.util.function.*;

public class DogSearch {
   void reduceList(List<String> names, Predicate<String>
tester) {
      names.removeIf(tester);
   }
   public static void main(String[] treats) {
      int MAX_LENGTH = 2;
      DogSearch search = new DogSearch();
      List<String> names = new ArrayList<>();
      names.add("Lassie");
      names.add("Benji");
      names.add("Brian");
      MAX_LENGTH += names.size();
      search.reduceList(names, d -> d.length()>MAX_LENGTH);
      System.out.print(names.size());
   }
}
```

A. `2`

B. `3`

C. The code does not compile because of lambda expression.

D. The code does not compile for a different reason.

37. Which functional interface takes two values and has an `apply()` method?

A. `BiConsumer`

B. `BiFunction`

C. `BiPredicate`

D. `DoubleBinaryOperator`

38. Which of the following lambda expressions can be passed to a method that takes `IntFunction<Integer>` as an argument?

I. `(Integer f) -> f`

II. `(v) -> null`

III. `s -> s`

A. I, II, and III

B. II and III only

C. III only

D. None of the above

39. What is the output of the following application?

```
package lot;
import java.util.function.*;

public class Warehouse {
   private int quantity = 40;
   private final BooleanSupplier stock;
   {
      stock = () -> quantity>0;
   }
   public void checkInventory() {
      if(stock.get())
         System.out.print("Plenty!");
      else {
         System.out.print("On Backorder!");
      }
   }
   public static void main(String... widget) {
      final Warehouse w13 = new Warehouse();
      w13.checkInventory();
   }
}
```

A. `Plenty!`

B. `On Backorder!`

C. The code does not compile because of the `checkInventory()` method.

D. The code does not compile for a different reason.

40. Which of the following statements about functional interfaces is true?

A. It is possible to define a functional interface that returns two data types.

B. It is possible to define a primitive functional interface that uses `float`, `char`, or `short`.

C. It is not possible to define a functional interface that does not take any arguments nor return any value.

D. None of the primitive functional interfaces include generic arguments.

Chapter 15
Java Stream API

THE OCP EXAM TOPICS COVERED IN THIS PRACTICE TEST INCLUDE THE FOLLOWING:

✓**Java Stream API**

- Develop code to extract data from an object using peek() and map() methods including primitive versions of the map() method
- Search for data by using search methods of the Stream classes including findFirst, findAny, anyMatch, allMatch, noneMatch
- Develop code that uses the Optional class
- Develop code that uses Stream data methods and calculation methods
- Sort a collection using Stream API
- Save results to a collection using the collect method and group/partition data using the Collectors class
- Use flatMap() methods in the Stream API

1. Which of the following fills in the blank so that the code outputs one line but uses a poor practice?

```
import java.util.*;

public class Cheater {
   int count = 0;
   public void sneak(Collection<String> coll) {
      coll.stream()._______________________;
   }

   public static void main(String[] args) {
      Cheater c = new Cheater();
      c.sneak(Arrays.asList("weasel"));
  }
}
```

   A. `peek(System.out::println)`

   B. `peek(System.out::println).findFirst()`

   C. `peek(r -> System.out.println(r)).findFirst()`

   D. `peek(r -> {count++; System.out.println(r); }).findFirst()`

2. Which can fill in the blank to have the code print `true`?

```
Stream<Integer> stream = Stream.iterate(1, i -> i+1);
boolean b = stream.____________(i -> i > 5);
System.out.println(b);
```

   A. `anyMatch`

   B. `allMatch`

   C. `noneMatch`

   D. None of the above

3. On a `DoubleStream`, how many of the methods `average()`, `count()`, and `sum()` return an `OptionalDouble`?

   A. None

   B. One

   C. Two

   D. Three

4. How many of the following can fill in the blank to have the code print `44`?

```
Stream<String> stream = Stream.of("base", "ball");
stream.____________(s ->
s.length()).forEach(System.out::print);
```

   I. `map`
   II. `mapToInt`
   III. `mapToObject`

   A. None
   B. One
   C. Two
   D. Three

5. What is the result of the following?

```
IntStream s = IntStream.empty();
System.out.print(s.average().getAsDouble());
```

   A. The code prints `0`.
   B. The code prints `0.0`.
   C. The code does not compile.
   D. The code compiles but throws an exception at runtime.

6. Which of these stream pipeline operations takes a `Predicate` as a parameter and returns an `Optional`?

   A. `anyMatch()`
   B. `filter()`
   C. `findAny()`
   D. None of the above

7. What is the result of the following?

```
List<Double> list = new ArrayList<>();
list.add(5.4);
list.add(1.2);
Optional<Double> opt = list.stream().sorted().findFirst();
```

```
System.out.println(opt.get() + " " + list.get(0));
```

A. `1.2 1.2`

B. `1.2 5.4`

C. `5.4 5.4`

D. None of the above

8. Fill in the blank so this code prints `8.0`.

```
IntStream stream = IntStream.of(6, 10);
LongStream longs = stream.mapToLong(i -> i);
System.out.println(____________);
```

A. `longs.average().get()`

B. `longs.average().getAsDouble()`

C. `longs.getAverage().get()`

D. `longs.getAverage().getAsDouble()`

9. How many of these collectors can fill in the blank to make this code compile?

```
Stream<Character> chars = Stream.of(
   'o', 'b', 's', 't', 'a', 'c', 'l', 'e');
chars.map(c -> c).collect(Collectors.____________ );
```

I. `toArrayList()`

II. `toList()`

III. `toMap()`

A. None

B. One

C. Two

D. Three

10. What does the following output?

```
import java.util.*;

public class MapOfMaps {
   public static void main(String[] args) {
      Map<Integer, Integer> map = new HashMap<>();
      map.put(9, 3);
      Map<Integer, Integer> result = map.stream().map((k,v) ->
(v,k));
      System.out.println(result.keySet().iterator().next());
   }
}
```

A. `3`

B. `9`

C. The code does not compile.

D. The code compiles but throws an exception at runtime.

11. Which of the following creates an `Optional` that returns `true` when calling `opt.isPresent()`?

    I. `Optional<String> opt = Optional.empty();`

    II. `Optional<String> opt = Optional.of(null);`

    III. `Optional<String> opt = Optional.ofNullable(null);`

    A. I

    B. I and II

    C. I and III

    D. None of the above

12. What is the output of the following?

```
Stream<String> s = Stream.of("speak", "bark", "meow", "growl");
BinaryOperator<String> merge = (a, b) -> a;
Map<Integer, String> map = s.collect(toMap(String::length, k ->
k, merge));
System.out.println(map.size() + " " + map.get(4));
```

A. `2 bark`

B. `2 meow`

C. `4 bark`

D. None of the above

13. What is the output of the following?

```
package reader;
import java.util.stream.*;
```

```

public class Books {
   public static void main(String[] args) {
      IntegerStream pages = IntegerStream.of(200, 300);
      IntegerSummaryStatistics stats =
pages.summaryStatistics();
      long total = stats.getSum();
      long count = stats.getCount();
      System.out.println(total + "-" + count);
   }
}
```

A. `500-0`

B. `500-2`

C. The code does not compile.

D. The code compiles but throws an exception at runtime.

14. If this method is called with `Stream.of("hi")`, how many lines are printed?

```
public static void print(Stream<String> stream) {
   Consumer<String> print = System.out::println;
   stream.peek(print)
         .peek(print)
         .map(s -> s)
         .peek(print)
         .forEach(print);
}
```

A. Three

B. Four

C. The code compiles but does not output anything.

D. The code does not compile.

15. What is true of the following code?

```
Stream<Character> stream = Stream.of('c', 'b', 'a');       //
z1
stream.sorted().findAny().ifPresent(System.out::println);  //
z2
```

A. It is guaranteed to print the single character `a`.

B. It can print any single character of `a`, `b`, or `c`.

C. It does not compile because of line `z1`.

D. It does not compile because of line `z2`.

16. Suppose you have a stream pipeline where all the elements are of type `String`. Which of the following can be passed to the intermediate operation `sorted()`?

A. `(s,t) -> s.length() - t.length()`

B. `String::isEmpty`

C. Both of these

D. Neither of these

17. Fill in the blanks so that both methods produce the same output for all inputs.

```
private static void longer(Optional<Boolean> opt) {
   if (opt.____________())
      System.out.println("run: " + opt.get());
}
private static void shorter(Optional<Boolean> opt) {
   opt.map(x -> "run: " + x).____________(System.out::println);
}
```

A. `isNotNull`, `isPresent`

B. `ifPresent`, `isPresent`

C. `isPresent`, `forEach`

D. `isPresent`, `ifPresent`

18. What is the output of this code?

```
Stream<Boolean> bools = Stream.iterate(true, b -> !b);
Map<Boolean, List<Boolean>> map = bools.limit(1)
    .collect(partitioningBy(b -> b));
System.out.println(map);
```

A. `{true=[true]}`

B. `{false=null, true=[true]}`

C. `{false=[], true=[true]}`

D. None of the above

19. What does the following output?

```
Set<String> set = new HashSet<>();
set.add("tire-");
List<String> list = new LinkedList<>();
Deque<String> queue = new ArrayDeque<>();
queue.push("wheel-");
Stream.of(set, list, queue)
   .flatMap(x -> x.stream())
   .forEach(System.out::print);
```

A. `[tire-][wheel-]`

B. `tire-wheel-`

C. None of the above.

D. The code does not compile.

20. What is the output of the following?

```
Stream<String> s = Stream.of("over the river",
    "through the woods",
    "to grandmother's house we go");
s.filter(n -> n.startsWith("t"))
 .sorted(Comparator::reverseOrder)
 .findFirst()
 .ifPresent(System.out::println);
```

A. `over the river`

B. `through the woods`

C. `to grandmother's house we go`

D. None of the above

21. Which fills in the blank so the code is guaranteed to print `1`?

```
Stream<Integer> stream = Stream.of(1, 2, 3);
System.out.println(stream.____________);
```

A. `findAny()`

B. `first()`

C. `min()`

D. None of the above

22. Which of the following can be the type for `x`?

```
private static void spot(____________ x) {
      x.filter(y -> ! y.isEmpty())
        .map(y -> 8)
        .ifPresent(System.out::println);
   }
```

I. `List<String>`

II. `Optional<Collection>`

III. `Optional<String>`

IV. `Stream<Collection>`

A. I

B. IV

C. II and III

D. II and IV

23. Which can fill in the blank to have the code print `true`?

```
Stream<Integer> stream = Stream.iterate(1, i -> i);
boolean b = stream.____________(i -> i > 5);
System.out.println(b);
```

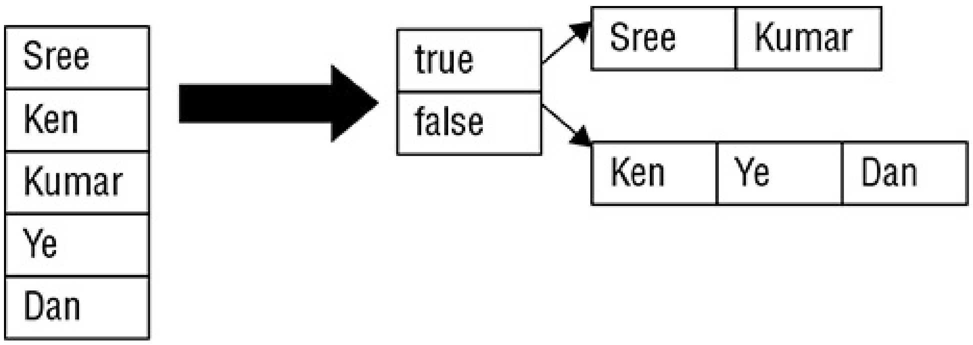
A. `anyMatch`

B. `allMatch`

C. `noneMatch`

D. None of the above

24. What collector turns the stream at left to the `Map` at right?



A. `grouping()`

B. `groupingBy()`

C. `partitioning()`

D. `partitioningBy()`

25. Which fills in the blank for this code to print `667788`?

```
IntStream ints = IntStream.empty();
IntStream moreInts = IntStream.of(66, 77, 88);
Stream.of(ints, moreInts).___________(x ->
x).forEach(System.out::print);
```

A. `flatMap`

B. `flatMapToInt`

C. `map`

D. None of the above

26. Fill in the blank so this code prints `8.0`. Note that it must not print `OptionalDouble[8.0]`.

```
LongStream stream = LongStream.of(6, 10);
LongSummaryStatistics stats = stream.summaryStatistics();
System.out.println(____________);
```

A. `stats.avg()`

B. `stats.average()`

C. `stats.average().get()`

D. `stats.getAverage()`

27. Which can independently fill in the blank to output `No dessert today`?

```
import java.util.*;
public class Dessert {
  public static void main(String[] yum) {
    eatDessert(Optional.of("Cupcake"));
  }
  private static void eatDessert(Optional<String> opt) {
     System.out.println(opt.____________);
  }
}
```

A. `get("No dessert today")`

B. `orElse("No dessert today")`

C. `orElseGet(() -> "No dessert today")`

D. None of the above

28. What does the following output?

```
Stream<Character> chars = Stream.generate(() -> 'a');
chars.filter(c -> c < 'b')
     .sorted()
     .findFirst()
     .ifPresent(System.out::print);
```

A. `a`

B. The code runs successfully without any output.

C. The code enters an infinite loop.

D. The code compiles but throws an exception at runtime.

29. How many of the following can fill in the blank to have the code print the single digit `9`?

```
LongStream stream = LongStream.of(9);
stream.____________(p -> p).forEach(System.out::print);
```

I. `mapToDouble`

II. `mapToInt`

III. `mapToLong`

A. None

B. One

C. Two

D. Three

30. Suppose you have a stream with one element and the code `stream.xxxx.forEach(System.out::println)`. Filling in `xxxx` from top to bottom in the table, how many elements can be printed out?

xxxx	**Number elements printed**
`filter()`	
`flatMap()`	
`map()`	

A. Zero or one, zero or more, exactly one

B. Zero or one, exactly one, zero or more

C. Zero or one, zero or more, zero or more

D. Exactly one, zero or more, zero or more

31. What is the output of the following?

```
Stream<Character> stream = Stream.of('c', 'b', 'a');
System.out.println(stream.sorted().findFirst());
```

A. It is guaranteed to print the single character `a`.

B. It can print any single character of `a`, `b`, or `c`.

C. The code does not compile.

D. None of the above

32. What is the output of the following?

```
public class Compete {
   public static void main(String[] args) {
      Stream<Integer> is = Stream.of(8, 6, 9);
      Comparator<Integer> c = (a, b) -> a - b;
      is.sort(c).forEach(System.out::print);
  }
}
```

A. `689`

B. `986`

C. The code does not compile

D. The code compiles but throws an exception at runtime.

33. What is the result of the following?

```
class Ballot {
   private String name;
   private int judgeNumber;
   private int score;

   public Ballot(String name, int judgeNumber, int score) {
      this.name = name;
      this.judgeNumber = judgeNumber;
      this.score = score;
   }
   // all getters and setters
}

public class Speaking {
   public static void main(String[] args) {
      Stream<Ballot> ballots = Stream.of(
         new Ballot("Mario", 1, 10),
         new Ballot("Christina", 1, 8),
         new Ballot("Mario", 2, 9),
         new Ballot("Christina", 2, 8)
      );

      Map<String, Integer> scores = ballots.collect(
         groupingBy(Ballot::getName,
summingInt(Ballot::getScore))); // w1
      System.out.println(scores.get("Mario"));
  }
}
```

A. The code prints `2`.

B. The code prints `19`.

C. The code does not compile due to line `w1`.

D. The code does not compile due to a different line.

34. Which can fill in the blank so this code outputs `true`?

```
import java.util.function.*;
import java.util.stream.*;

public class HideAndSeek {
   public static void main(String[] args) {
      Stream<Boolean> hide = Stream.of(true, false, true);
      boolean found = hide.filter(b -> b).__________();
      System.out.println(found);
   }
}
```

A. Only `anyMatch`

B. Only `allMatch`

C. Both `anyMatch` and `allMatch`

D. The code does not compile with any of these options.

35. What does the following output?

```
Set<String> set = new HashSet<>();
```

```
set.add("tire-");
List<String> list = new LinkedList<>();
Deque<String> queue = new ArrayDeque<>();
queue.push("wheel-");
Stream.of(set, list, queue)
   .flatMap(x -> x)
   .forEach(System.out::print);
```

A. `[tire-][wheel-]`

B. `tire-wheel-`

C. None of the above

D. The code does not compile.

36. When working with a `Stream<String>`, which of these types can be returned from the `collect()` terminal operator by passing arguments to `Collectors.groupingBy()`?

I. `Map<Integer, List<String>>`

II. `Map<Boolean, HashSet<String>>`

III. `List<String>`

A. I

B. II

C. I and II

D. I, II, and III

37. Which line can replace line 18 without changing the output of the program?

```
1:   class Runner {
2:      private int numberMinutes;
3:      public Runner(int n) {
4:         numberMinutes = n;
5:      }
6:      public int getNumberMinutes() {
7:         return numberMinutes;
8:      }
9:      public boolean isFourMinuteMile() {
10:        return numberMinutes < 4*60;
11:     }
12:  }
13:  public class Marathon {
14:     public static void main(String[] args) {
```

```
15:         Stream<Runner> runners = Stream.of(new Runner(250),
16:            new Runner(600), new Runner(201));
17:         long count = runners
18:              .filter(Runner::isFourMinuteMile)
19:              .count();
20:         System.out.println(count);
21:     }
22:  }
```

A. `.map(Runner::isFourMinuteMile)`

B. `.mapToBool(Runner::isFourMinuteMile)`
`.filter(b -> b == true)`

C. `.mapToBoolean(Runner::isFourMinuteMile)`
`.filter(b -> b == true)`

D. None of the above

38. Which method is not available on the `IntSummaryStatistics` class?

A. `getCountAsLong()`

B. `getMax()`

C. `toString()`

D. None of the above—all three methods are available.

39. Which can fill in the blank so this code outputs `Caught it`?

```
import java.util.*;
public class Catch {
   public static void main(String[] args) {
      Optional opt = Optional.empty();
      try {
         apply(opt);
      } catch (IllegalArgumentException e) {
         System.out.println("Caught it");
      }
   }
   private static void apply(Optional<Exception> opt) {
      opt.__________(IllegalArgumentException::new);
   }
}
```

A. `orElse`

B. `orElseGet`

C. `orElseThrow`

D. None of the above. The `main()` method does not compile.

40. A developer tries to rewrite a method that uses `flatMap()` without using that intermediate operator. Which pair of method calls shows the `withoutFlatMap()` method is not equivalent to the `withFlatMap()` method?

```
public static void main(String[] args) {
   List<String> list = new LinkedList<>();
   Deque<String> queue = new ArrayDeque<>();
   queue.push("all queued up");
   queue.push("last");
}

private static void withFlatMap(Collection<?> coll) {
   Stream.of(coll)
      .flatMap(x -> x.stream())
      .forEach(System.out::print);
   System.out.println();
}

private static void withoutFlatMap(Collection<?> coll) {
   Stream.of(coll)
      .filter(x -> !x.isEmpty())
      .map(x -> x)
      .forEach(System.out::print);
   System.out.println();
}
```

A. `withFlatMap(list); withoutFlatMap(list);`

B. `withFlatMap(queue); withoutFlatMap(queue);`

C. Both pairs disprove the claim.

D. Neither pair disproves this claim.