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.

1. `IntConsumer`, `LongSupplier`
2. `IntSupplier`, `Function`
3. `Supplier`, `DoubleConsumer`
4. `UnaryOperator`, `Consumer`

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

1. `LongConsumer`
2. `LongFunction`
3. `LongPredicate`
4. `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 touris
      System.out.print(canSave.test(tourist) ? "Saved" : "Too far")
   }
   public final static void main(String... sand) {
      new Lifeguard().saveLife(s -> s.distance<4, new Tourist(2));
   }
}
```

1. `Saved`
2. `Too far`
3. The code does not compile because of line `y1`.
4. The code does not compile because of line `y2`.

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

1. Both are functional interfaces.
2. Lambdas for both can return a `double` value.
3. Lambdas for both cannot return a `null` value.
4. 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);
   }
}
```

1. `BiConsumer<Integer,Double>`
2. `BiFunction<Integer,Double,Double>`
3. `BiFunction<Integer,Integer,Double>`
4. `Function<Integer,Double>`

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

1. `(Integer a, Double b) -> {int c; return b;}`
2. `(h,i) -> (long)h`
3. `(String u, Object v) -> u.length()+v.length()`
4. `(x,y) -> {int z=2; return y/z;}`

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

1. `BiPredicate`
2. `DoubleUnaryOperator`
3. `ObjectDoubleConsumer`
4. `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
      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);
   }
}
```

1. `51`

2. The code does not compile because of lambda expression.
3. The code does not compile for a different reason.
4. The code compiles but prints an exception at runtime.

9. Which functional interface returns a primitive value?

1. `BiPredicate`
2. `CharSupplier`
3. `LongFunction`
4. `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);
            });
   }

}
```

1. `Consumer`
2. `DoubleConsumer`
3. `Supplier<Double>`
4. None of the above

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

1. `BiFunction<Double,Double,Double>`
2. `BinaryOperator<Double>`
3. `DoubleFunction<Double>`
4. 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);
   }
}
```

1. `magic((a) -> a)`
2. `magic((b,w) -> (long)w.intValue())`
3. `magic((c,m) -> {long c=4; return c+m;})`
4. `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!")); // d
   }
}
```

1. `Program started!`
2. The code does not compile because of line `d1` only.
3. The code does not compile because of line `d2` only.
4. 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?

1. The input type must be compatible with the return type.
2. Some of them take multiple arguments.
3. They each take a generic argument.
4. 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`	

1. `0`, `1`, `1`, `1`
2. `0`, `2`, `1`, `2`
3. `1`, `1`, `0`, `2`
4. `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`	

1. `double`, `R`, `long`, `R`
2. `R`, `int`, `long`, `R`
3. `void`, `int`, `R`, `void`
4. `void`, `R`, `long`, `void`

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

1. `Function`, `Function`
2. `Predicate`, `Consumer`
3. `Predicate`, `Function`
4. `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));
   }
}
```

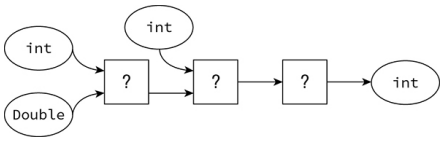
---

1. XXXXX
2. The code does not compile because of the lambda expression.
3. The code does not compile for a different reason.
4. The code compiles but produces an infinite loop at runtime.

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

1. `Consumer`
2. `Function`
3. `Supplier`
4. `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?



1. `DoubleBinaryOperator`

   `ToDoubleBiFunction<Integer,Double>`

   `UnaryOperator<Integer>`

2. `BinaryOperator<Double>`

   `BiFunction<Integer,Integer,Double>`

   `UnaryOperator<Integer>`

3. `Function<Double,Integer>`

   `BiFunction<Integer,Integer,Double>`

   `DoubleToIntFunction`

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

   `BinaryOperator<Integer>`

   `IntUnaryOperator`

21. Which statement about functional interfaces and lambda expressions is not true?

1. A lambda expression may be compatible with multiple functional interfaces.
2. A lambda expression must be assigned to a functional interface when it is declared.
3. A method can return a lambda expression in the form of a functional interface instance.
4. 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?

    1. `() -> 1<10 ? "3" : 4`
    2. `() -> {return 1/0;}`
    3. `() -> return 4`
    4. `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);
   }
}
```

    1. `[JENNY, GRACE]`
    2. `[tps.Boss@4218224c, tps.Boss@815f19a]`
    3. The code does not compile because of the lambda expression.
    4. 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?

    1. `ArrayList::new`
    2. `String::new`
    3. `System.out::println`

    1. I only
    2. I, II, and III
    3. I and III
    4. III only

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

    1. `FloatPredicate`
    2. `ToDoubleBiFunction`
    3. `UnaryIntOperator`
    4. `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);
   }
}
```

    1. `DoubleUnaryOperator`
    2. `Function<String,String>`
    3. `IntToLongFunction`
    4. `UnaryOperator`

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

    1. `BiFunction<T,U,R>`
    2. `DoubleFunction<T,R>`
    3. `ToDoubleFunction<T>`
    4. `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> tak
      return takeABreak.apply(3, 10.2);
   }
   public static void main(String[] participants) {
      rest(___________);
   }
}
```

    1. `(int n, double e) -> (int)(n+e)`
    2. `(n,w,e) -> System.out::print`
    3. `(s,w) -> 2*w`
    4. `(s,e) -> s.intValue()+e.intValue()`

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

    1. `BooleanSupplier`
    2. `CharPredicate`
    3. `FloatUnaryOperator`
    4. `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(depo
      System.out.print(creditUnion.savingsInCents);
   }
}
```

    1. `200`
    2. `250`
    3. The code does not compile because of line `j1`.
    4. None of the above

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

    1. `DoubleConsumer`
    2. `DoublePredicate`
    3. `DoubleUnaryOperator`
    4. `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: }
```

1. None. The code compiles as is.
2. One
3. Two
4. Three

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

1. `a -> null==null ? 1 : 2L`
2. `e -> (int)(10.0*e)`
3. `(double m) -> {long p = (long)m; return p;}`
4. `(Double s) -> s.longValue()`

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

1. `DoublePredicate`
2. `LongUnaryOperator`
3. `ShortSupplier`
4. `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);
   }
}
```

1. `Consumer<Sheep>`
2. `Function<Sheep,void>`
3. `UnaryOperator<Sheep>`
4. 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());
   }
}
```

---

1. 2
2. 3
3. The code does not compile because of lambda expression.
4. The code does not compile for a different reason.

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

1. `BiConsumer`
2. `BiFunction`
3. `BiPredicate`
4. `DoubleBinaryOperator`

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

1. `(Integer f) -> f`
2. `(v) -> null`
3. `s -> s`

1. I, II, and III
2. II and III only
3. III only
4. 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();
   }
}
```

---

1. `Plenty!`
2. `On Backorder!`
3. The code does not compile because of the `checkInventory()` method.
4. The code does not compile for a different reason.

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

1. It is possible to define a functional interface that returns two data types.
2. It is possible to define a primitive functional interface that uses `float`, `char`, or `short`.
3. It is not possible to define a functional interface that does not take any arguments nor return any value.
4. None of the primitive functional interfaces include generic arguments.

PREV
Chapter 13 Generics and Collections

NEXT
Chapter 15 Java Stream API