absolutely must memorize Table 8.4, which lists the common functional interfaces.

Exam Essentials

Write simple lambda expressions. Look for the presence or absence of optional elements in lambda code. Parameter types are optional. Braces and the return keyword are optional when the body is a single statement. Parentheses are optional when only one parameter is specified and the type is implicit.

Determine whether a variable can be used in a lambda body. Local variables and method parameters must be final or effectively final to be referenced. This means the code must compile if you were to add the final keyword to these variables. Instance and class variables are always allowed.

Translate method references to the "long form" lambda. Be able to convert method references into regular lambda expressions and vice versa. For example, System.out::print and x -> System.out.print(x) are equivalent. Remember that the order of method parameters is inferred when using a method reference.

Determine whether an interface is a functional interface. Use the single abstract method (SAM) rule to determine whether an interface is a functional interface. Other interface method types (default, private, static, and private static) do not count toward the single abstract method count, nor do any public methods with signatures found in Object.

Identify the correct functional interface given the number of parameters, return type, and method name—and vice versa. The most common functional interfaces are Supplier, Consumer, Function, and Predicate. There are also binary versions and primitive versions of many of these methods. You can use the number of parameters and return type to tell them apart.

Review Questions

The answers to the chapter review questions can be found in the Appendix.

1. What is the result of the following class?

```
1: import java.util.function.*;
2:
3: public class Panda {
4:   int age;
5:   public static void main(String[] args) {
6:     Panda p1 = new Panda();
7:     p1.age = 1;
8:     check(p1, p -> p.age < 5);
9:   }
10:   private static void check(Panda panda,
11:     Predicate<Panda> pred) {
12:     String result =
13:       pred.test(panda) ? "match" : "not match";
14:     System.out.print(result);
15: } }
```

A. match

B. not match

C. Compiler error on line 8

D. Compiler error on lines 10 and 11

E. Compiler error on lines 12 and 13

F. A runtime exception is thrown.

2. What is the result of the following code?

```
1:  interface Climb {
2:    boolean isTooHigh(int height, int limit);
3:  }
4:
5:  public class Climber {
6:    public static void main(String[] args) {
7:      check((h, m) -> h.append(m).isEmpty(), 5);
8:    }
9:    private static void check(Climb climb, int height) {
10:     if (climb.isTooHigh(height, 10))
11:       System.out.println("too high");
12:     else
13:       System.out.println("ok");
14:   }
15: }
```

A. ok

B. too high

C. Compiler error on line 7

D. Compiler error on line 10

E. Compiler error on a different line

F. A runtime exception is thrown.

3. Which statements about functional interfaces are true? (Choose all that apply.)

A. A functional interface can contain default and private methods.

B. A functional interface can be defined as a class or an interface.

C. Abstract methods with signatures that are contained in public methods of java.lang.Object do not count toward the abstract method count for a functional interface.

D. A functional interface cannot contain static or private static methods.

E. A functional interface must be marked with the @FunctionalInterface annotation.

4. Which lambda can replace the MySecret class to return the same value? (Choose all that apply.)

```
interface Secret {
  String magic(double d);
}

class MySecret implements Secret {
  public String magic(double d) {
    return "Poof";
  }}
```

A. (e) -> "Poof"

B. (e) -> {"Poof"}

C. (e) -> { String e = ""; "Poof" }

D. (e) -> { String e = ""; return "Poof"; }

E. (e) -> { String e = ""; return "Poof" }

F. (e) -> { String f = ""; return "Poof"; }

5. Which of the following functional interfaces contain an abstract method that returns a primitive value? (Choose all that apply.)

A. BooleanSupplier

B. CharSupplier

C. DoubleSupplier

D. FloatSupplier

E. IntSupplier

F. StringSupplier

6. Which of the following lambda expressions can be passed to a function of Predicate<String> type? (Choose all that apply.)

A. s -> s.isEmpty()

B. s --> s.isEmpty()

C. (String s) -> s.isEmpty()

D. (String s) --> s.isEmpty()

E. (StringBuilder s) -> s.isEmpty()

F. (StringBuilder s) --> s.isEmpty()

7. Which of these statements is true about the following code?

```
public void method() {
  x((var x) -> {}, (var x, var y) -> false);
}
public void x(Consumer<String> x, BinaryOperator<Boolean> y) {}
```

A. The code does not compile because of one of the variables named x.

B. The code does not compile because of one of the variables named y.

C. The code does not compile for another reason.

D. The code compiles, and the x in each lambda refers to the same type.

E. The code compiles, and the x in each lambda refers to a different type.

8. Which of the following is equivalent to this code? (Choose all that apply.)

```
UnaryOperator<Integer> u = x -> x * x;
```

A. BiFunction<Integer> f = x -> x*x;

B. BiFunction<Integer, Integer> f = x -> x*x;

C. BinaryOperator<Integer, Integer> f = x -> x*x;

D. Function<Integer> f = x -> x*x;

E. Function<Integer, Integer> f = x -> x*x;

F. None of the above

9. Which statements are true? (Choose all that apply.)

A. The Consumer interface is good for printing out an existing value.

B. The Supplier interface is good for printing out an existing value.

C. The IntegerSupplier interface returns an int.

D. The Predicate interface returns an int.

E. The Function interface has a method named test().

F. The Predicate interface has a method named test().

10. Which of the following can be inserted without causing a compilation error? (Choose all that apply.)

```
public void remove(List<Character> chars) {
  char end = 'z';
  Predicate<Character> predicate = c -> {
    char start = 'a'; return start <= c && c <= end; };
  // INSERT LINE HERE
}
```

A. char start = 'a';

B. char c = 'x';

C. chars = null;

D. end = '1';

E. None of the above

11. How many times istrue printed out by this code?

```
import java.util.function.Predicate;
public class Fantasy {
  public static void scary(String animal) {
    var dino = s -> "dino".equals(animal);
    var dragon = s -> "dragon".equals(animal);
    var combined = dino.or(dragon);
    System.out.println(combined.test(animal));
  }
  public static void main(String[] args) {
    scary("dino");
    scary("dragon");
    scary("unicorn");
  }
}
```

A. One

B. Two

C. Three

D. The code does not compile.

E. A runtime exception is thrown.

12. What does the following code output?

```
Function<Integer, Integer> s = a -> a + 4;
Function<Integer, Integer> t = a -> a * 3;
Function<Integer, Integer> c = s.compose(t);
System.out.print(c.apply(1));
```

A. 7

B. 15

C. The code does not compile because of the data types in the lambda expressions.

D. The code does not compile because of thecompose() call.

E. The code does not compile for another reason.

13. Which is true of the following code?

```
int length = 3;

for (int i = 0; i<3; i++) {
  if (i%2 == 0) {
    Supplier<Integer> supplier = () -> length; // A
    System.out.println(supplier.get());      // B
  } else {
    int j = i;
    Supplier<Integer> supplier = () -> j;     // C
    System.out.println(supplier.get());      // D
  }
}
```

A. The first compiler error is on lineA.

B. The first compiler error is on lineB.

C. The first compiler error is on lineC.

D. The first compiler error is on lineD.

E. The code compiles successfully.

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

A. (Wolf w, var c) -> 39

B. (final Camel c) -> {}

C. (a,b,c) -> {int b = 3; return 2;}

D. (x,y) -> new RuntimeException()

E. (var y) -> return 0;

F. () -> {float r}

G. (Cat a, b) -> {}

15. Which lambda expression, when entered into the blank line in the following code, causes the program to printhahaha? (Choose all that apply.)

```
import java.util.function.Predicate;
public class Hyena {
  private int age = 1;
  public static void main(String[] args) {
    var p = new Hyena();
    double height = 10;
    int age = 1;
    testLaugh(p, ________________________);
    age = 2;
  }
  static void testLaugh(Hyena panda, Predicate<Hyena> joke) {
    var r = joke.test(panda) ? "hahaha" : "silence";
    System.out.print(r);
  }
}
```

A. var -> p.age <= 10

B. shenzi -> age==1

C. p -> true

D. age==1

E. shenzi -> age==2

F. h -> h.age < 5

G. None of the above, as the code does not compile

16. Which of the following can be inserted without causing a compilation error? (Choose all that apply.)

```
public void remove(List<Character> chars) {
  char end = 'z';

  // INSERT LINE HERE

  Predicate<Character> predicate =  c -> {
    char start = 'a'; return start <= c && c <= end; };
}
```

A. char start = 'a';

B. char c = 'x';

C. chars = null;

D. end = '1';

E. None of the above

17. What is the result of running the following class?

```
1:  import java.util.function.*;
2:
3:  public class Panda {
4:    int age;
5:    public static void main(String[] args) {
```

```
6:     Panda p1 = new Panda();
7:     p1.age = 1;
8:     check(p1, p -> {p.age < 5});
9:   }
10:  private static void check(Panda panda,
11:     Predicate<Panda> pred) {
12:     String result = pred.test(panda)
13:        ? "match" : "not match";
14:     System.out.print(result);
15: } }
```

A. match

B. not match

C. Compiler error on line 8

D. Compiler error on line 10

E. Compiler error on line 12

F. A runtime exception is thrown.

18. Which functional interfaces complete the following code? For line 7, assumem andn are instances of functional interfaces that exist and have the same type asy. (Choose three.)

```
6: _________ x = String::new;
7: _________ y = m.andThen(n);
8: _________ z = a -> a + a;
```

A. BinaryConsumer<String, String>

B. BiConsumer<String, String>

C. BinaryFunction<String, String>

D. BiFunction<String, String>

E. Predicate<String>

F. Supplier<String>

G. UnaryOperator<String>

H. UnaryOperator<String, String>

19. Which of the following compiles and prints out the entire set? (Choose all that apply.)

```
Set<?> set = Set.of("lion", "tiger", "bear");
var s = Set.copyOf(set);
Consumer<Object> consumer = ___________________;
s.forEach(consumer);
```

A. () -> System.out.println(s)

B. s -> System.out.println(s)

C. (s) -> System.out.println(s)

D. System.out.println(s)

E. System::out::println

F. System.out::println

20. Which lambdas can replace thenew Sloth() call in themain() method and produce the same output at runtime? (Choose all that apply.)

```
import java.util.List;
interface Yawn {
  String yawn(double d, List<Integer> time);
}
class Sloth implements Yawn {
  public String yawn(double zzz, List<Integer> time) {
    return "Sleep: " + zzz;
  }}
public class Vet {
  public static String takeNap(Yawn y) {
    return y.yawn(10, null);
  }
  public static void main(String… unused) {
    System.out.print(takeNap(new Sloth()));
  }}
```

A. (z,f) -> { String x = ""; return "Sleep: " + x }

B. (t,s) -> { String t = ""; return "Sleep: " + t; }

C. (w,q) -> {"Sleep: " + w}

D. (e,u) -> { String g = ""; "Sleep: " + e }

E. (a,b) -> "Sleep: " + (double)(b==null ? a : a)

F. (r,k) -> { String g = ""; return "Sleep:"; }

G. None of the above, as the program does not compile

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

```
public interface Transport {
  public int go();
  public boolean equals(Object o);
}
```

```
public abstract class Car {
  public abstract Object swim(double speed, int duration);
}

public interface Locomotive extends Train {
  public int getSpeed();
}

public interface Train extends Transport {}

abstract interface Spaceship extends Transport {
  default int blastOff();
}

public interface Boat {
  int hashCode();
  int hashCode(String input);
}
```

A. Boat

B. Car

C. Locomotive

D. Spaceship

E. Transport

F. Train

G. None of these is a valid functional interface.