IRC_SKCT_Java2_MCQ_StreamAPI

Test Summary

- No. of Sections: 1
- No. of Questions: 15
- Total Duration: 30 min

Section 1 - MCQ

Section Summary

- No. of Questions: 15
- Duration: 30 min

Additional Instructions:

None

Q1. 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));
```

1.2 1.2

1.2 5.4

5.4 5.4

None of the above

Q2. How many of these collectors can fill in the blank to make this code compile?
1.toArrayList()
2.toList()
3.toMap()

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

None

One

Two

Three

Q3. 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"));
   }
}

```

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

Q4. On a DoubleStream, how many of the methods average(), count(), and sum() return an OptionalDouble?

- None
- One
- Two
- Three

Q5. choose the best option based on this program.

```
import java.util.OptionalInt;
import java.util.stream.IntStream;

public class FindMax {
   public static void main(String args[]) {
      maxMarks(IntStream.of(52,60,99,80,76));          // #1
}
   public static void maxMarks(IntStream marks) {
      OptionalInt max = marks.max();           // #2
      if(max.ifPresent()) {                    // #3
         System.out.print(max.getAsInt());          }
}
}

```

- This program results in a compiler error in line marked with comment #1
- This program results in a compiler error in line marked with comment #2
- This program results in a compiler error in line marked with comment #3
- This program prints: 99

```
IntStream stream = IntStream.of( 6, 10);
```

Fill in the blank so this code prints 8.0.

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

- longs.average(). get()
- longs.average(). getAsDouble()
- longs.getAverage(). get()
- longs.getAverage(). getAsDouble()

Q7. What is the result of the following?

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

- The code prints 0.
- The code prints 0.0.
- The code does not compile.
- The code compiles but throws an exception at runtime.

Q8. How many of the following can fill in the blank to have the code print 44?
1.map
2.mapToInt
3.mapToObject

```
Stream < String > stream = Stream.of(" base", "ball");

stream.____________( s -> s.length()). forEach( System.out:: print);
```

- None
- One
- Two

Three

Q9. 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());
   }
}
```

3

9

The code does not compile.

The code compiles but throws an exception at runtime.

Q10. Which of these stream pipeline operations takes a Predicate as a parameter and returns an Optional?

anyMatch()

filter()

findAny()

None of the above

Q11. 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);
```

anyMatch

allMatch

noneMatch

None of the above

Q12. Choose the best option based on this program.

```
import java.util.stream.Stream;
```

```
public class AllMatch {
   public static void main(String []args) {
      boolean result = Stream.of("do", "re", "mi", "fa", "so", "la", "ti")
         .filter(str -> str.length() > 5)       // #1
         .peek(System.out::println)             // #2
         .allMatch(str -> str.length() > 5);    // #3
      System.out.println(result);
      }
}
```

This program results in a compiler error in line marked with comment #1

This program results in a compiler error in line marked with comment #2

This program results in a compiler error in line marked with comment #3

This program prints: true

Q13. Choose the best option based on this program.

```
import java.util.function.IntPredicate;
import java.util.stream.IntStream;

public class MatchUse {
   public static void main(String []args) {
      IntStream temperatures = IntStream.of(-5, -6, -7, -5, 2, -8, -9);
      IntPredicate positiveTemperature = temp -> temp > 0;   // #1
      if(temperatures.anyMatch(positiveTemperature)) {       // #2
         int temp = temperatures
         .filter(positiveTemperature)
         .findAny()
         .getAsInt();                 // #3
         System.out.println(temp);
      }
}
}
```

this program results in a compiler error in line marked with comment #1

this program results in a compiler error in line marked with comment #2

this program results in a compiler error in line marked with comment #3

this program crashes by throwing java.lang.IllegalStateException

Q14. Choose the best option based on this program.

```
import java.util.regex.Pattern; import java.util.stream.Stream;

public class SumUse {
   public static void main(String []args) {
      Stream<String> words = Pattern.compile(" ").splitAsStream("a bb ccc");
      System.out.println(words.map(word -> word.length()).sum());
   }
}
```

Compiler error: Cannot find symbol “sum” in interface Stream<Integer>

this program prints: 3

this program prints: 5

this program prints: 6

Q15. Choose the best option based on this program.

```
import java.util.Optional;
import java.util.stream.Stream;

public class StringToUpper {
   public static void main(String args[]){
       Stream.of("eeny ","meeny ",null).forEach(StringToUpper::toUpper);
   }
   private static void toUpper(String str) {
      Optional <String> string = Optional.ofNullable(str);
      System.out.print(string.map(String::toUpperCase).orElse("dummy"));
      }
}

```

this program prints: eeNY meeNY dummy

this program prints: eeNY meeNY DUmmY

this program prints: eeNY meeNY null

this program prints: Optional[eeNY] Optional[meeNY] Optional[dummy]

Answer Key & Solution

Section 1 - MCQ

Q1

1.2 5.4

Solution

No Solution

Q2

One

Solution

No Solution

Q3

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

Solution

No Solution

Q4

One

Solution

No Solution

Q5

This program results in a compiler error in line marked with comment #3

Solution

No Solution

Q6

longs.average(). getAsDouble()

Solution

No Solution

Q7

The code compiles but throws an exception at runtime.

Solution

No Solution

Q8

Two

Solution

No Solution

Q9

The code does not compile.

Solution

No Solution

Q10

None of the above

Solution

No Solution

Q11

anyMatch

Solution

No Solution

Q12

This program prints: true

Solution

No Solution

Q13

this program crashes by throwing java.lang.IllegalStateException

Solution

No Solution

Q14

Compiler error: Cannot find symbol “sum” in interface Stream<Integer>

Solution

No Solution

Q15

this program prints: eeNY meeNY dummy

Solution

No Solution