GitHub Java Code Search Engine

Have you ever written methods/functions where you realize that they must have been written before? Imagine being able to retrieve the exact code that you are looking for? On GitHub, millions of people are publishing their code for anyone to use, but there is no obvious way of finding the code snippet that you are looking for.

We have solved this problem by creating a search engine that retrieves the method that you are looking for (Java only). Right now, we are evaluating the results retrieved by the search engine and we need the help of experts for that.

In this form, we will propose a query search and you will need to punctuate the distinct results with the following criteria:

Criteria
(0) Irrelevant method.
(1) Marginally relevant method.
(2) Fairly relevant method.
(3) Highly relevant method.

It is not necessary to spend a lot of time in each question, just simulate the scenario where you're skimming through the responses to the search. The estimated time for completing the entire survey is less than 10 minutes.

Note: this is part of our course in Search Engines and Information Retrieval System (DD2476) at KTH. Thank you for helping us develop a really great project!

*Obrigatório

Query 1

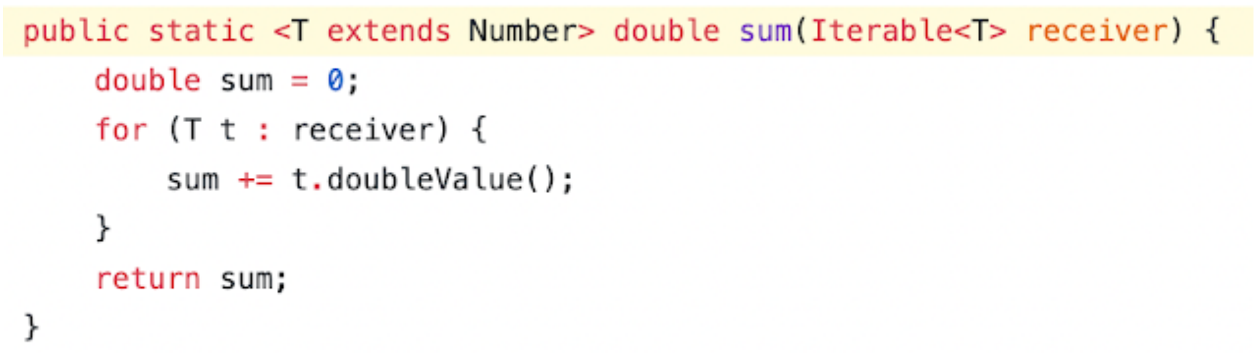
You are looking for a function that sums all the elements inside an array. You do a search for the method name "sum".

1. All the questions on this page refer to the above query. *

Marcar tudo o que for aplicável.

☐ I understand

2. 1. How relevant is the following code to the query: "sum"? *

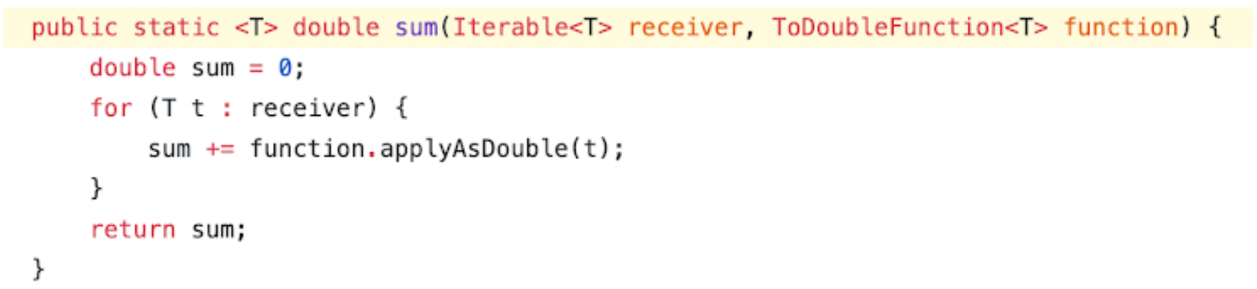


```
public static <T extends Number> double sum(Iterable<T> receiver) {
    double sum = 0;
    for (T t : receiver) {
        sum += t.doubleValue();
    }
    return sum;
}
```

Marcar apenas uma oval.

	0	1	2	3	
Irrelevant	◯	◯	◯	◯	Highly relevant

3. 2. How relevant is the following code to the query: "sum"? *

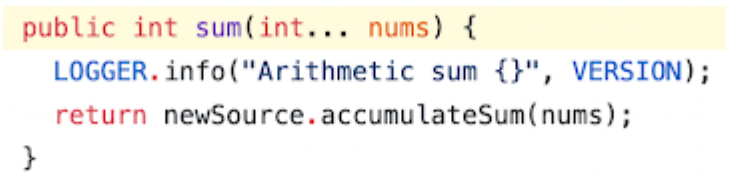


```
public static <T> double sum(Iterable<T> receiver, ToDoubleFunction<T> function) {
    double sum = 0;
    for (T t : receiver) {
        sum += function.applyAsDouble(t);
    }
    return sum;
}
```

Marcar apenas uma oval.

	0	1	2	3	
Irrelevant	◯	◯	◯	◯	Highly relevant

4. 3. How relevant is the following code to the query: "sum"? *

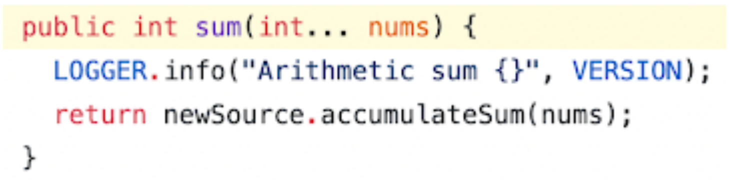


```
public int sum(int... nums) {
  LOGGER.info("Arithmetic sum {}", VERSION);
  return newSource.accumulateSum(nums);
}
```

Marcar apenas uma oval.

	0	1	2	3	
Irrelevant	⬭	⬭	⬭	⬭	Highly relevant

5. 4. How relevant is the following code to the query: "sum"? *

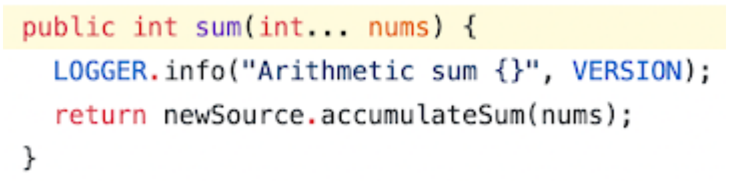


```
public int sum(int... nums) {
  LOGGER.info("Arithmetic sum {}", VERSION);
  return newSource.accumulateSum(nums);
}
```

Marcar apenas uma oval.

	0	1	2	3	
Irrelevant	⬭	⬭	⬭	⬭	Highly relevant

6. 5. How relevant is the following code to the query: "sum"? *

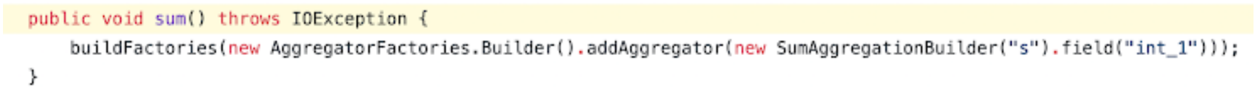


```
public int sum(int... nums) {
  LOGGER.info("Arithmetic sum {}", VERSION);
  return newSource.accumulateSum(nums);
}
```

Marcar apenas uma oval.

	0	1	2	3	
Irrelevant	⬭	⬭	⬭	⬭	Highly relevant

7. 6. How relevant is the following code to the query: "sum"? *

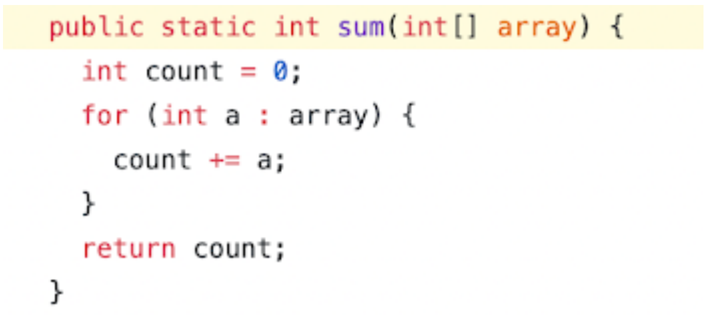


```java
public void sum() throws IOException {
    buildFactories(new AggregatorFactories.Builder().addAggregator(new SumAggregationBuilder("s").field("int_1")));
}
```

Marcar apenas uma oval.

	0	1	2	3	
Irrelevant	○	○	○	○	Highly relevant

8. 7. How relevant is the following code to the query: "sum"? *

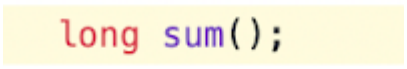


```java
public static int sum(int[] array) {
  int count = 0;
  for (int a : array) {
    count += a;
  }
  return count;
}
```

Marcar apenas uma oval.

	0	1	2	3	
Irrelevant	○	○	○	○	Highly relevant

9. 8. How relevant is the following code to the query: "sum"? *

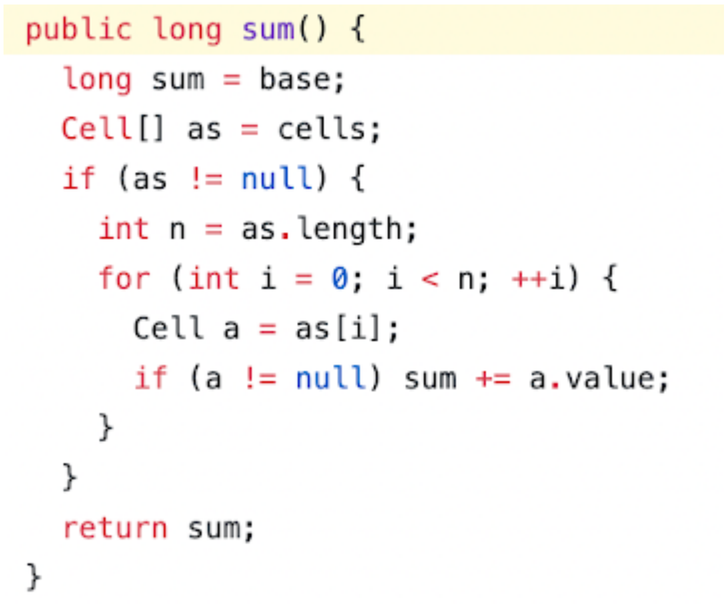


```java
long sum();
```

Marcar apenas uma oval.

	0	1	2	3	
Irrelevant	○	○	○	○	Highly relevant

10. 9. How relevant is the following code to the query: "sum"? *

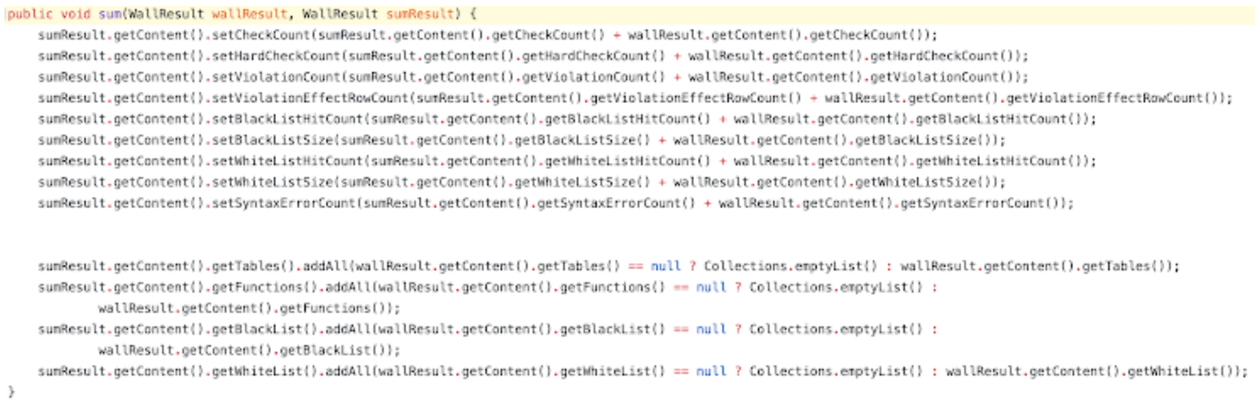


```
public long sum() {
  long sum = base;
  Cell[] as = cells;
  if (as != null) {
    int n = as.length;
    for (int i = 0; i < n; ++i) {
      Cell a = as[i];
      if (a != null) sum += a.value;
    }
  }
  return sum;
}
```

Marcar apenas uma oval.

	0	1	2	3	
Irrelevant	⬭	⬭	⬭	⬭	Highly relevant

11. 10. How relevant is the following code to the query: "sum"? *

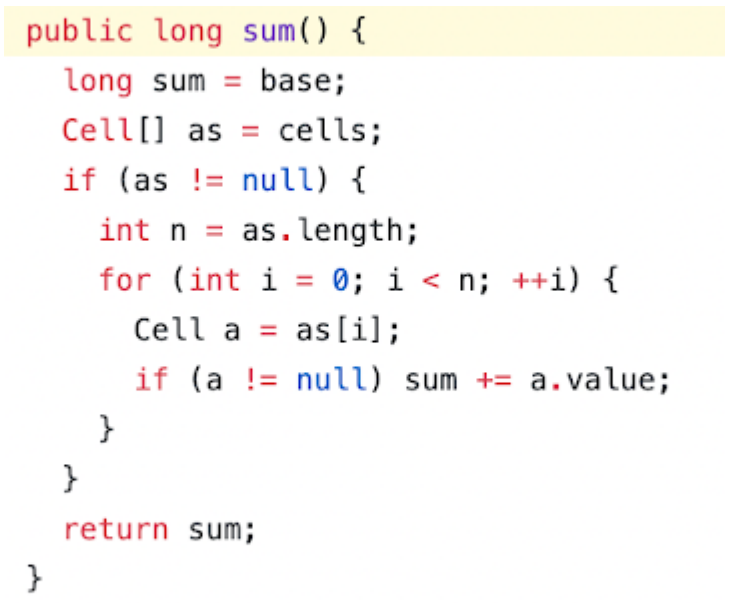


```java
public void sum(WallResult wallResult, WallResult sumResult) {
    sumResult.getContent().setCheckCount(sumResult.getContent().getCheckCount() + wallResult.getContent().getCheckCount());
    sumResult.getContent().setHardCheckCount(sumResult.getContent().getHardCheckCount() + wallResult.getContent().getHardCheckCount());
    sumResult.getContent().setViolationCount(sumResult.getContent().getViolationCount() + wallResult.getContent().getViolationCount());
    sumResult.getContent().setViolationEffectRowCount(sumResult.getContent().getViolationEffectRowCount() + wallResult.getContent().getViolationEffectRowCount());
    sumResult.getContent().setBlackListHitCount(sumResult.getContent().getBlackListHitCount() + wallResult.getContent().getBlackListHitCount());
    sumResult.getContent().setBlackListSize(sumResult.getContent().getBlackListSize() + wallResult.getContent().getBlackListSize());
    sumResult.getContent().setWhiteListHitCount(sumResult.getContent().getWhiteListHitCount() + wallResult.getContent().getWhiteListHitCount());
    sumResult.getContent().setWhiteListSize(sumResult.getContent().getWhiteListSize() + wallResult.getContent().getWhiteListSize());
    sumResult.getContent().setSyntaxErrorCount(sumResult.getContent().getSyntaxErrorCount() + wallResult.getContent().getSyntaxErrorCount());


    sumResult.getContent().getTables().addAll(wallResult.getContent().getTables() == null ? Collections.emptyList() : wallResult.getContent().getTables());
    sumResult.getContent().getFunctions().addAll(wallResult.getContent().getFunctions() == null ? Collections.emptyList() :
            wallResult.getContent().getFunctions());
    sumResult.getContent().getBlackList().addAll(wallResult.getContent().getBlackList() == null ? Collections.emptyList() :
            wallResult.getContent().getBlackList());
    sumResult.getContent().getWhiteList().addAll(wallResult.getContent().getWhiteList() == null ? Collections.emptyList() : wallResult.getContent().getWhiteList());
}
```

Marcar apenas uma oval.

	0	1	2	3	
Irrelevant	⬭	⬭	⬭	⬭	Highly relevant

12. 11. How relevant is the following code to the query: "sum"? *

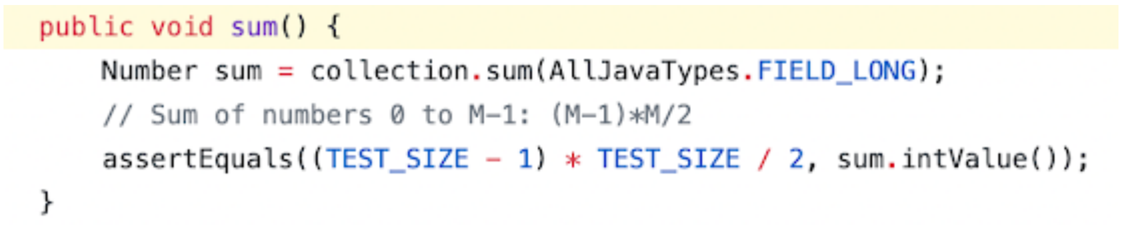


```java
public long sum() {
  long sum = base;
  Cell[] as = cells;
  if (as != null) {
    int n = as.length;
    for (int i = 0; i < n; ++i) {
      Cell a = as[i];
      if (a != null) sum += a.value;
    }
  }
  return sum;
}
```

Marcar apenas uma oval.

	0	1	2	3	
Irrelevant	⬭	⬭	⬭	⬭	Highly relevant

13. 12. How relevant is the following code to the query: "sum"? *

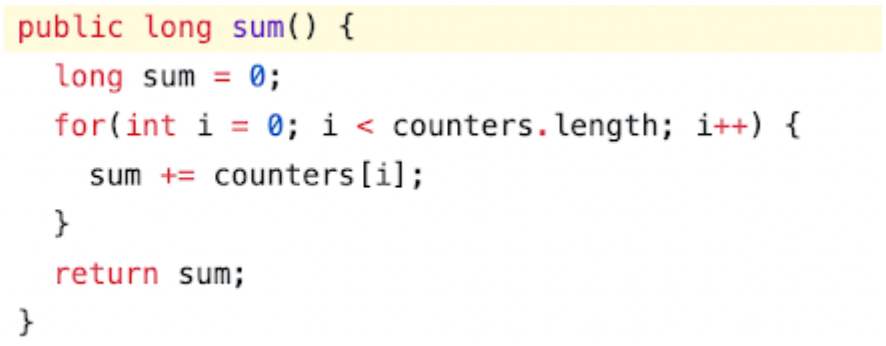


```java
public void sum() {
    Number sum = collection.sum(AllJavaTypes.FIELD_LONG);
    // Sum of numbers 0 to M-1: (M-1)*M/2
    assertEquals((TEST_SIZE - 1) * TEST_SIZE / 2, sum.intValue());
}
```

Marcar apenas uma oval.

	0	1	2	3	
Irrelevant	○	○	○	○	Highly relevant

14. 13. How relevant is the following code to the query: "sum"? *

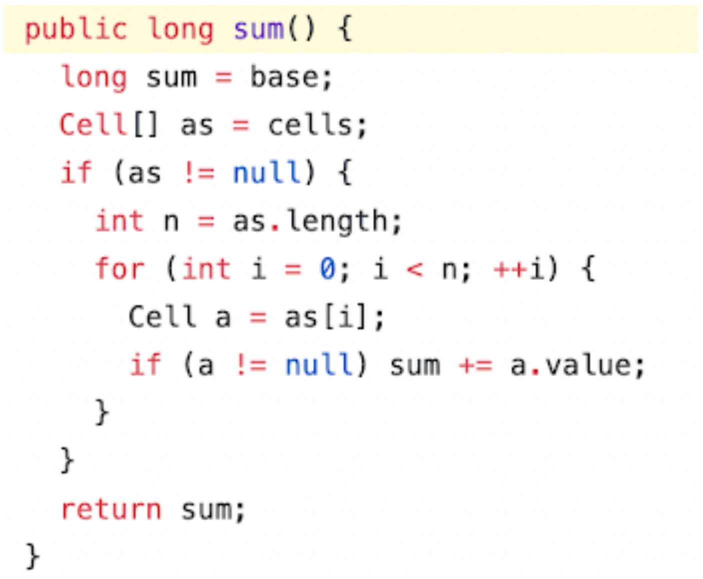


```java
public long sum() {
  long sum = 0;
  for(int i = 0; i < counters.length; i++) {
    sum += counters[i];
  }
  return sum;
}
```

Marcar apenas uma oval.

	0	1	2	3	
Irrelevant	○	○	○	○	Highly relevant

15. 14. How relevant is the following code to the query: "sum"? *



```java
public long sum() {
    long sum = base;
    Cell[] as = cells;
    if (as != null) {
        int n = as.length;
        for (int i = 0; i < n; ++i) {
            Cell a = as[i];
            if (a != null) sum += a.value;
        }
    }
    return sum;
}
```

Marcar apenas uma oval.

	0	1	2	3	
Irrelevant	◯	◯	◯	◯	Highly relevant

Query 2

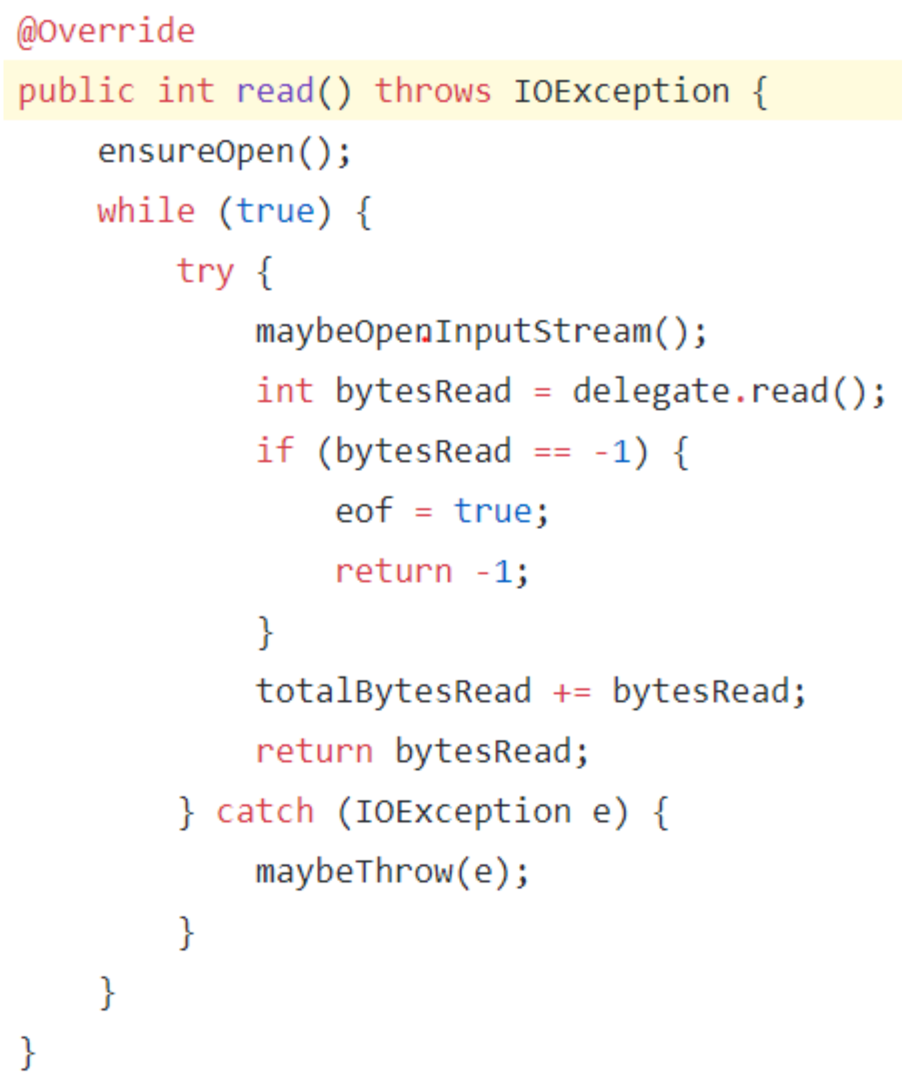
You are looking for a method that reads files. You search for the method name "read*" and the return type "int" (that should be the number of bytes read).
Note: The '*' at the end of the word means that you are looking for all the entries that start with that word. (i.e. you can also accept results such as readAll, readFile, etc.)

16. All the questions on this page refer to the above query *

Marcar tudo o que for aplicável.

☐ I understand

17. 1. How relevant is the following code to the query "read*"? *

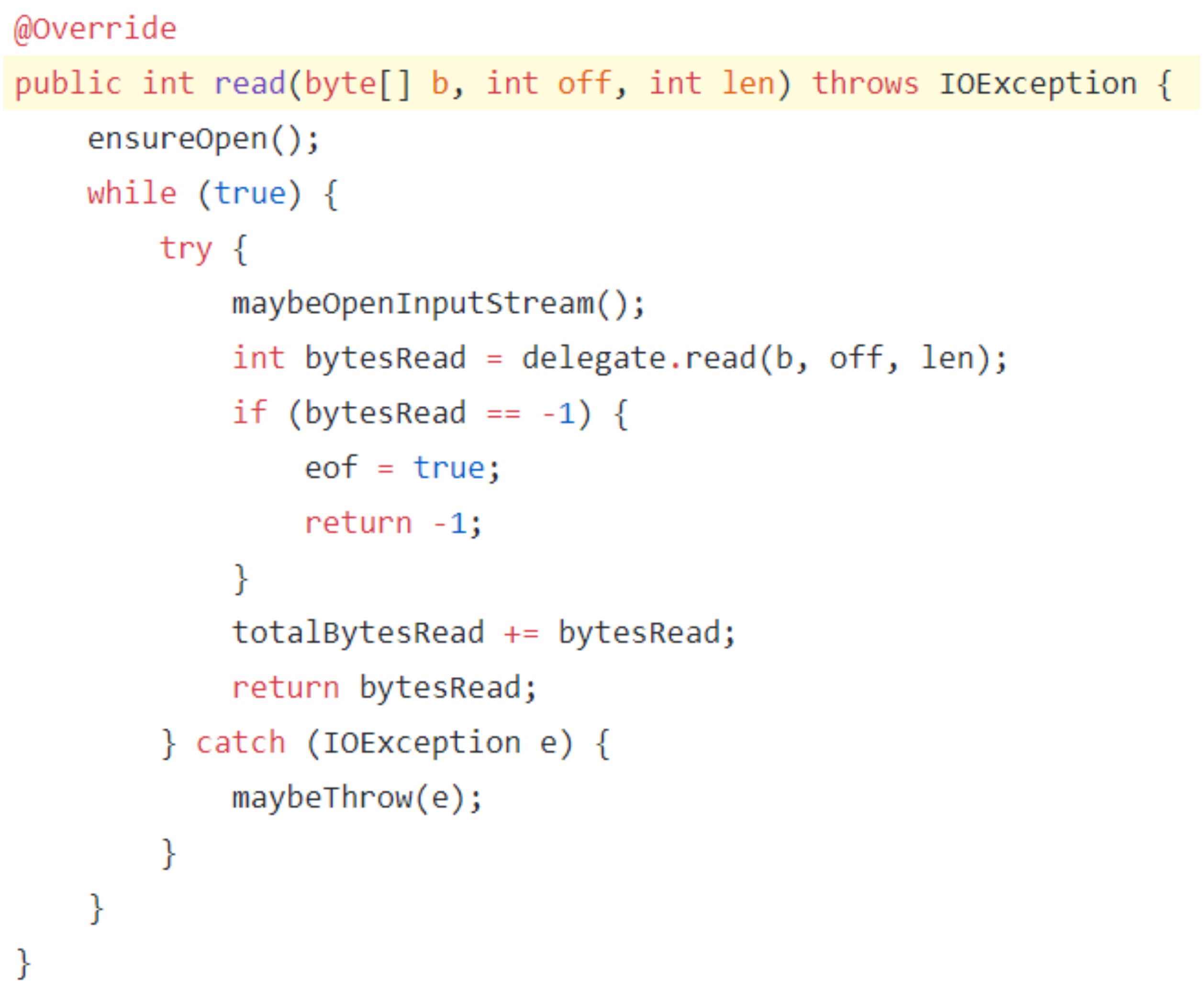


```
@Override
public int read() throws IOException {
    ensureOpen();
    while (true) {
        try {
            maybeOpenInputStream();
            int bytesRead = delegate.read();
            if (bytesRead == -1) {
                eof = true;
                return -1;
            }
            totalBytesRead += bytesRead;
            return bytesRead;
        } catch (IOException e) {
            maybeThrow(e);
        }
    }
}
```

Marcar apenas uma oval.

	0	1	2	3	
Irrelevant	◯	◯	◯	◯	Highly relevant

18. 2. How relevant is the following code to the query "read*"? *

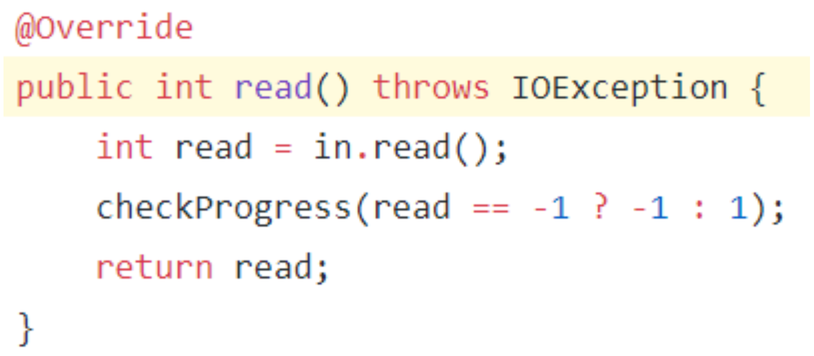


```java
@Override
public int read(byte[] b, int off, int len) throws IOException {
    ensureOpen();
    while (true) {
        try {
            maybeOpenInputStream();
            int bytesRead = delegate.read(b, off, len);
            if (bytesRead == -1) {
                eof = true;
                return -1;
            }
            totalBytesRead += bytesRead;
            return bytesRead;
        } catch (IOException e) {
            maybeThrow(e);
        }
    }
}
```

Marcar apenas uma oval.

	0	1	2	3	
Irrelevant	⬭	⬭	⬭	⬭	Highly relevant

19. 3. How relevant is the following code to the query "read*"? *

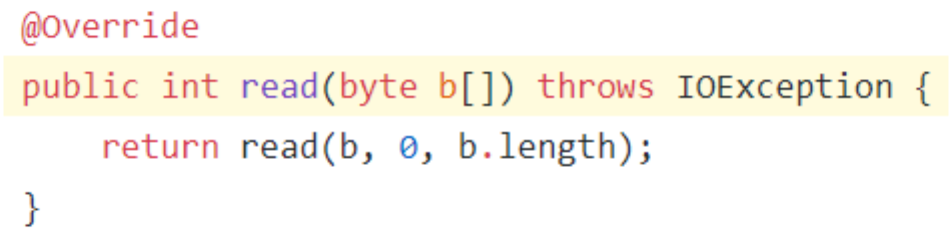


```
@Override
public int read() throws IOException {
    int read = in.read();
    checkProgress(read == -1 ? -1 : 1);
    return read;
}
```

Marcar apenas uma oval.

	0	1	2	3	
Irrelevant					Highly relevant

20. 4. How relevant is the following code to the query "read*"? *

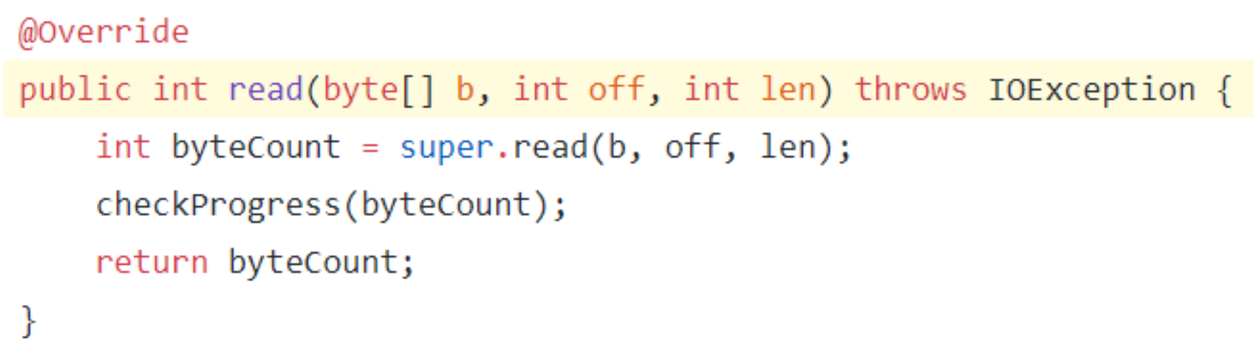


```
@Override
public int read(byte b[]) throws IOException {
    return read(b, 0, b.length);
}
```

Marcar apenas uma oval.

	0	1	2	3	
Irrelevant					Highly relevant

21. 5. How relevant is the following code to the query "read*"? *

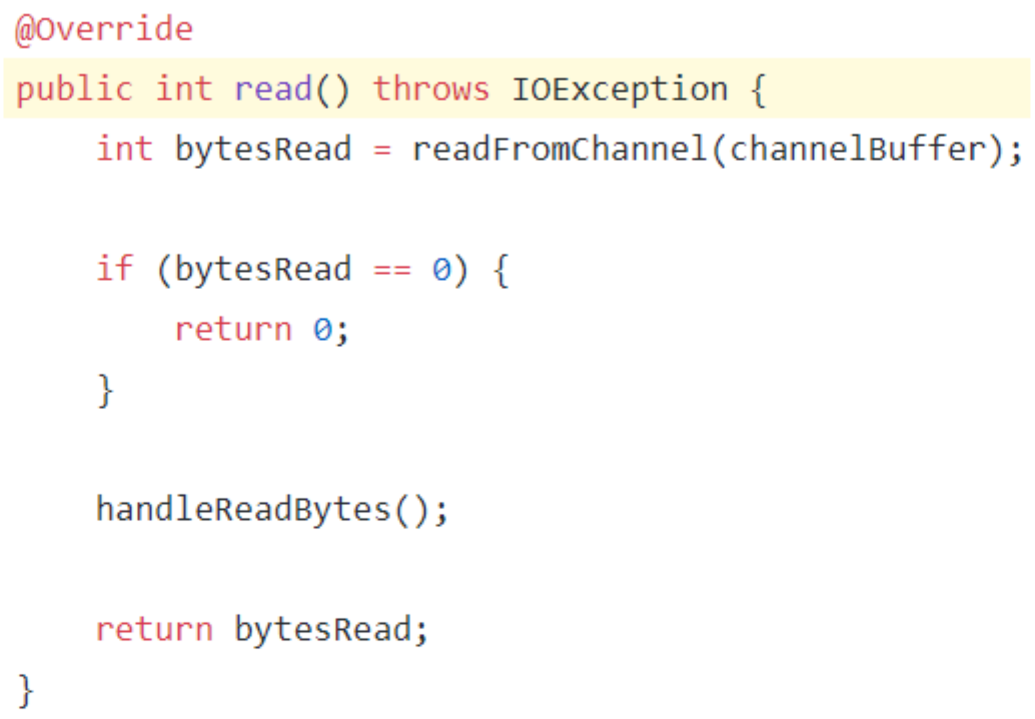


```java
@Override
public int read(byte[] b, int off, int len) throws IOException {
    int byteCount = super.read(b, off, len);
    checkProgress(byteCount);
    return byteCount;
}
```

Marcar apenas uma oval.

	0	1	2	3	
Irrelevant	◯	◯	◯	◯	Highly relevant

22. 6. How relevant is the following code to the query "read*"? *

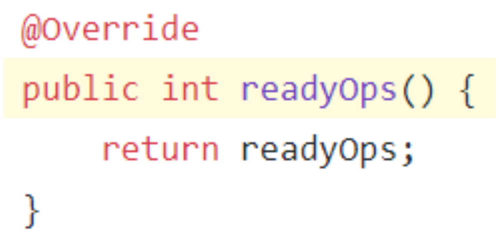


```java
@Override
public int read() throws IOException {
    int bytesRead = readFromChannel(channelBuffer);

    if (bytesRead == 0) {
        return 0;
    }

    handleReadBytes();

    return bytesRead;
}
```

Marcar apenas uma oval.

	0	1	2	3	
Irrelevant	◯	◯	◯	◯	Highly relevant

23. 7. How relevant is the following code to the query "read*"? *

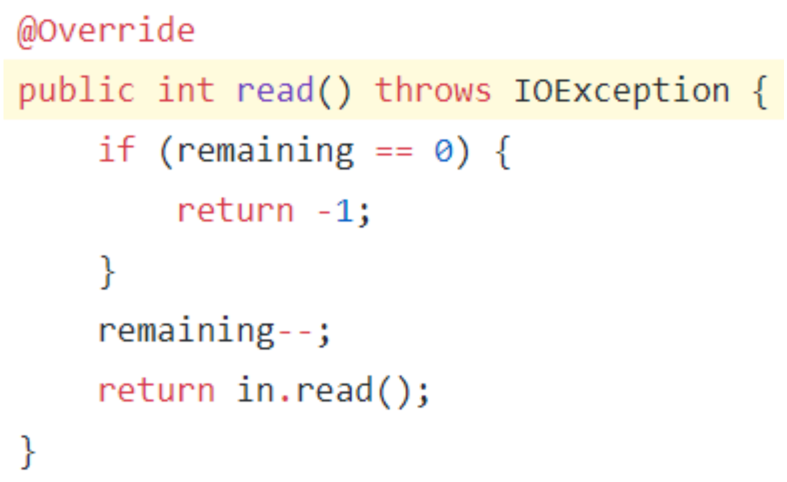


```java
@Override
public int readyOps() {
    return readyOps;
}
```

Marcar apenas uma oval.

	0	1	2	3	
Irrelevant	◯	◯	◯	◯	Highly relevant

24. 8. How relevant is the following code to the query "read*"? *

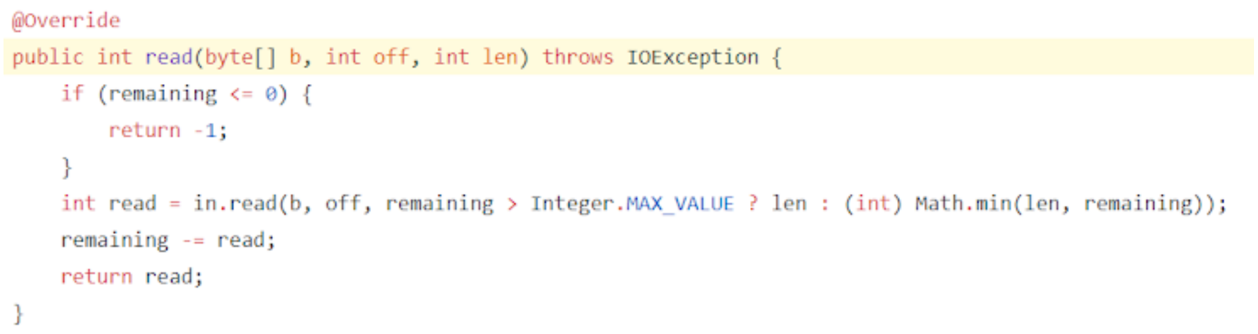


```
@Override
public int read() throws IOException {
    if (remaining == 0) {
        return -1;
    }
    remaining--;
    return in.read();
}
```

Marcar apenas uma oval.

	0	1	2	3	
Irrelevant	⬭	⬭	⬭	⬭	Highly relevant

25. 9. How relevant is the following code to the query "read*"? *

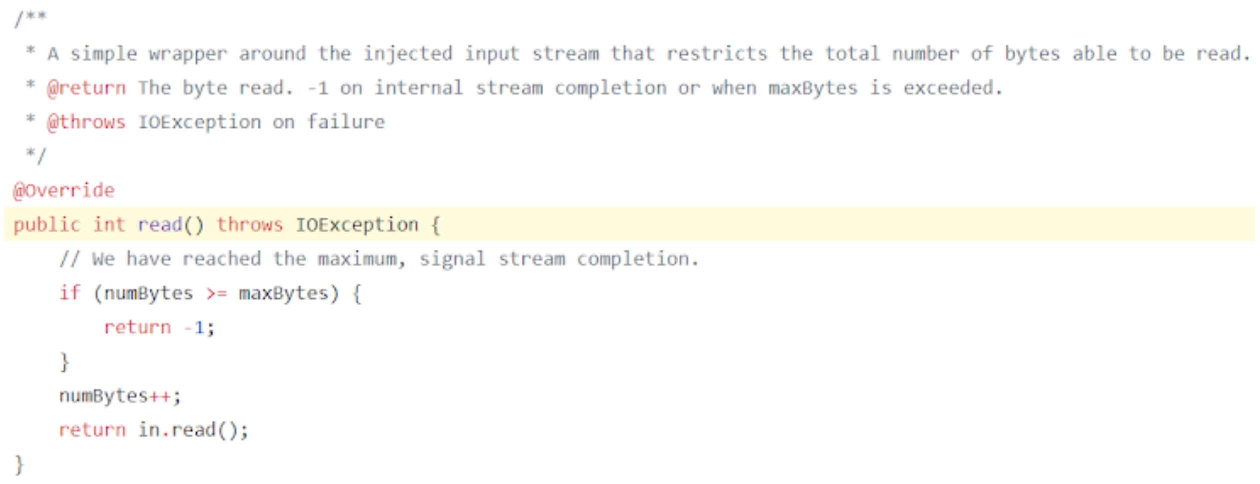


```
@Override
public int read(byte[] b, int off, int len) throws IOException {
    if (remaining <= 0) {
        return -1;
    }
    int read = in.read(b, off, remaining > Integer.MAX_VALUE ? len : (int) Math.min(len, remaining));
    remaining -= read;
    return read;
}
```

Marcar apenas uma oval.

	0	1	2	3	
Irrelevant	⬭	⬭	⬭	⬭	Highly relevant

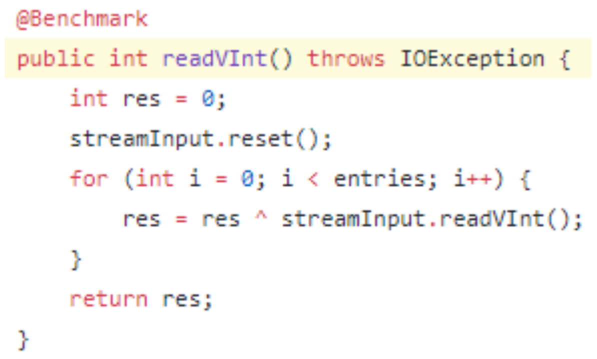
26. 10. How relevant is the following code to the query "read*"? *


```
/**
 * A simple wrapper around the injected input stream that restricts the total number of bytes able to be read.
 * @return The byte read. -1 on internal stream completion or when maxBytes is exceeded.
 * @throws IOException on failure
 */
@Override
public int read() throws IOException {
    // We have reached the maximum, signal stream completion.
    if (numBytes >= maxBytes) {
        return -1;
    }
    numBytes++;
    return in.read();
}
```

Marcar apenas uma oval.

	0	1	2	3	
Irrelevant	◯	◯	◯	◯	Highly relevant

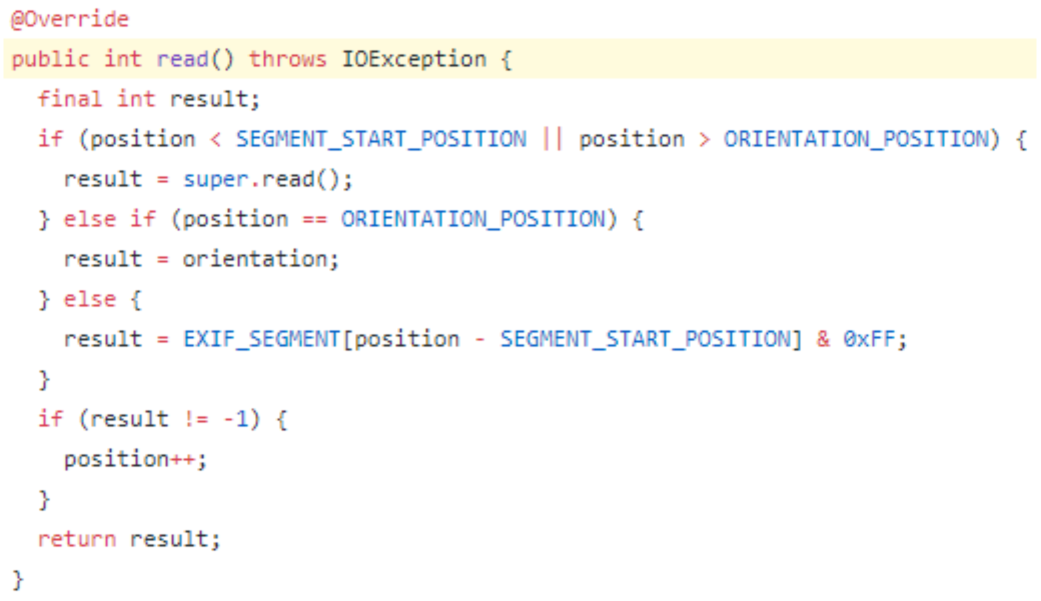
27. 11. How relevant is the following code to the query "read*"? *


```
@Benchmark
public int readVInt() throws IOException {
    int res = 0;
    streamInput.reset();
    for (int i = 0; i < entries; i++) {
        res = res ^ streamInput.readVInt();
    }
    return res;
}
```

Marcar apenas uma oval.

	0	1	2	3	
Irrelevant	◯	◯	◯	◯	Highly relevant

28. 12. How relevant is the following code to the query "read*"? *

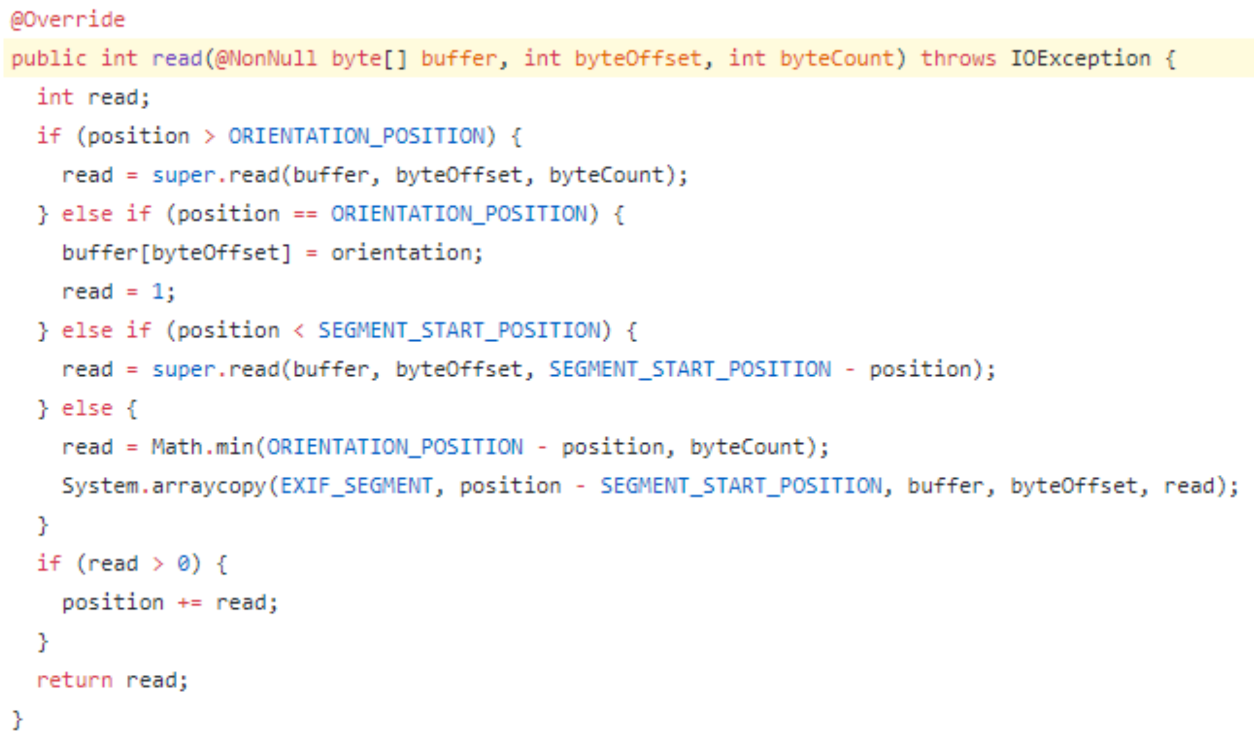


```
@Override
public int read() throws IOException {
  final int result;
  if (position < SEGMENT_START_POSITION || position > ORIENTATION_POSITION) {
    result = super.read();
  } else if (position == ORIENTATION_POSITION) {
    result = orientation;
  } else {
    result = EXIF_SEGMENT[position - SEGMENT_START_POSITION] & 0xFF;
  }
  if (result != -1) {
    position++;
  }
  return result;
}
```

Marcar apenas uma oval.

	0	1	2	3	
Irrelevant	○	○	○	○	Highly relevant

29. 13. How relevant is the following code to the query "read*"? *

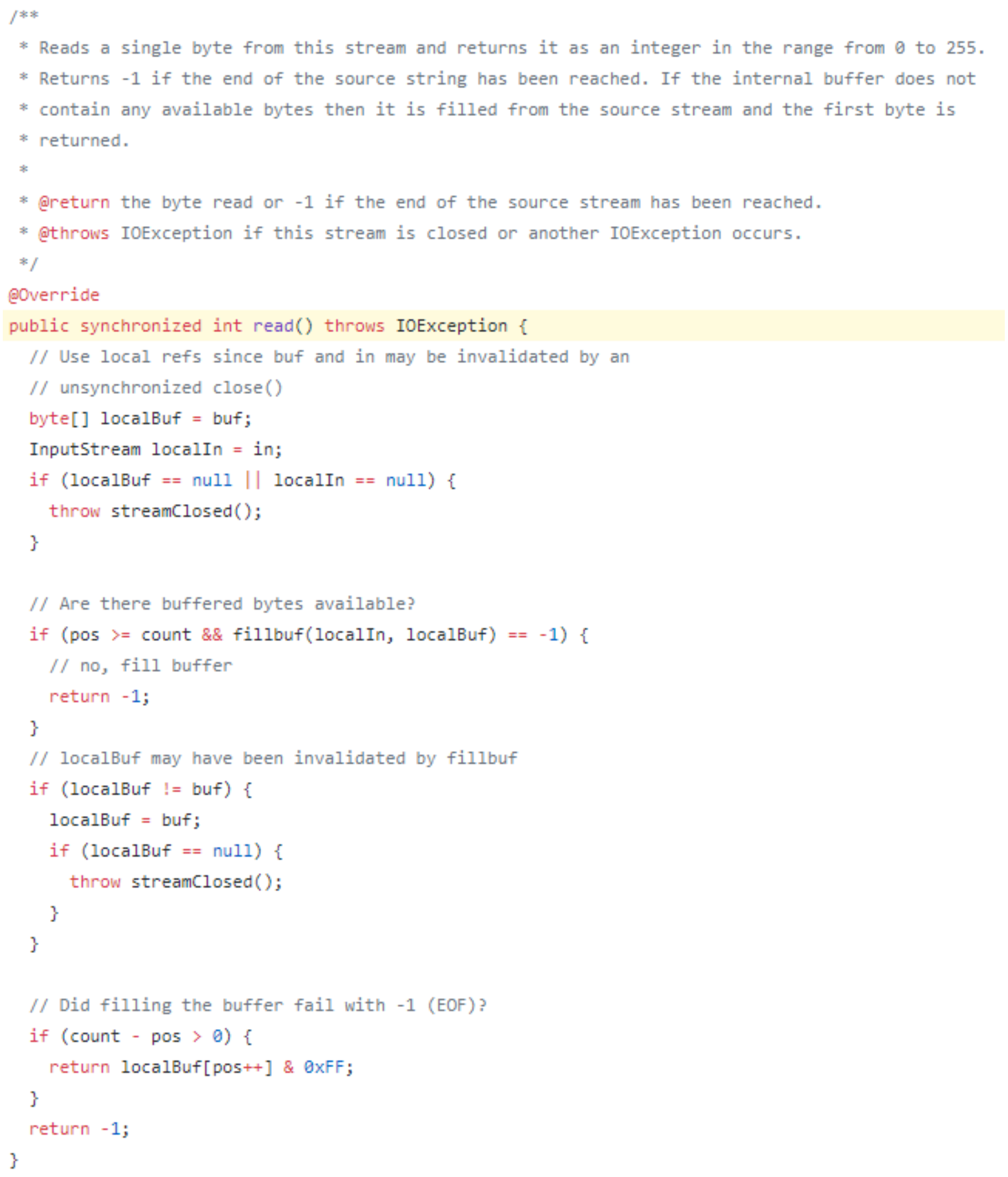


```java
@Override
public int read(@NonNull byte[] buffer, int byteOffset, int byteCount) throws IOException {
  int read;
  if (position > ORIENTATION_POSITION) {
    read = super.read(buffer, byteOffset, byteCount);
  } else if (position == ORIENTATION_POSITION) {
    buffer[byteOffset] = orientation;
    read = 1;
  } else if (position < SEGMENT_START_POSITION) {
    read = super.read(buffer, byteOffset, SEGMENT_START_POSITION - position);
  } else {
    read = Math.min(ORIENTATION_POSITION - position, byteCount);
    System.arraycopy(EXIF_SEGMENT, position - SEGMENT_START_POSITION, buffer, byteOffset, read);
  }
  if (read > 0) {
    position += read;
  }
  return read;
}
```

Marcar apenas uma oval.

	0	1	2	3	
Irrelevant	◯	◯	◯	◯	Highly relevant

30. 14. How relevant is the following code to the query "read*"? *



```
/**
 * Reads a single byte from this stream and returns it as an integer in the range from 0 to 255.
 * Returns -1 if the end of the source string has been reached. If the internal buffer does not
 * contain any available bytes then it is filled from the source stream and the first byte is
 * returned.
 *
 * @return the byte read or -1 if the end of the source stream has been reached.
 * @throws IOException if this stream is closed or another IOException occurs.
 */
@Override
public synchronized int read() throws IOException {
  // Use local refs since buf and in may be invalidated by an
  // unsynchronized close()
  byte[] localBuf = buf;
  InputStream localIn = in;
  if (localBuf == null || localIn == null) {
    throw streamClosed();
  }

  // Are there buffered bytes available?
  if (pos >= count && fillbuf(localIn, localBuf) == -1) {
    // no, fill buffer
    return -1;
  }
  // localBuf may have been invalidated by fillbuf
  if (localBuf != buf) {
    localBuf = buf;
    if (localBuf == null) {
      throw streamClosed();
    }
  }

  // Did filling the buffer fail with -1 (EOF)?
  if (count - pos > 0) {
    return localBuf[pos++] & 0xFF;
  }
  return -1;
}
```

Marcar apenas uma oval.

	0	1	2	3	
Irrelevant	◯	◯	◯	◯	Highly relevant

31. 15. How relevant is the following code to the query "read*"? *

```java
@Override
public synchronized int read() throws IOException {
  int value = super.read();
  checkReadSoFarOrThrow(value >= 0 ? 1 : -1);
  return value;
}
```

Marcar apenas uma oval.

	0	1	2	3	
Irrelevant	◯	◯	◯	◯	Highly relevant

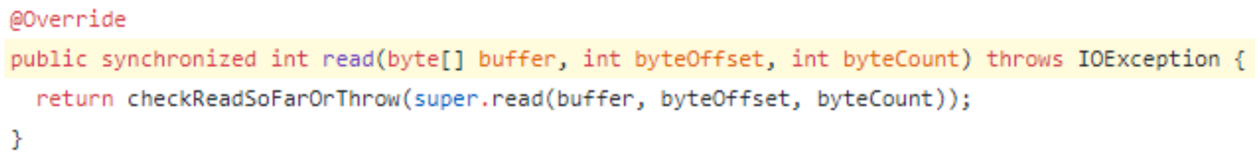
32. 16. How relevant is the following code to the query "read*"? *

```java
@Override
public int read(byte[] buffer) throws IOException {
  return read(buffer, 0 /*byteOffset*/, buffer.length /*byteCount*/);
}
```

Marcar apenas uma oval.

	0	1	2	3	
Irrelevant	◯	◯	◯	◯	Highly relevant

33. 17. How relevant is the following code to the query "read*"? *

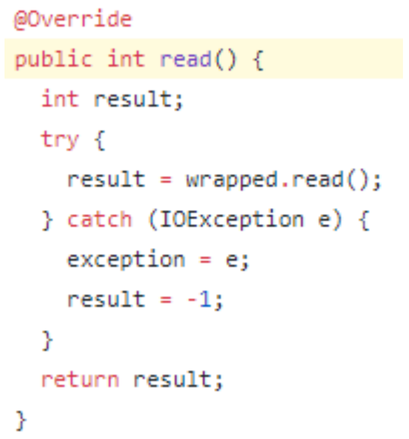


```
@Override
public synchronized int read(byte[] buffer, int byteOffset, int byteCount) throws IOException {
  return checkReadSoFarOrThrow(super.read(buffer, byteOffset, byteCount));
}
```

Marcar apenas uma oval.

	0	1	2	3	
Irrelevant	◯	◯	◯	◯	Highly relevant

34. 18. How relevant is the following code to the query "read*"? *

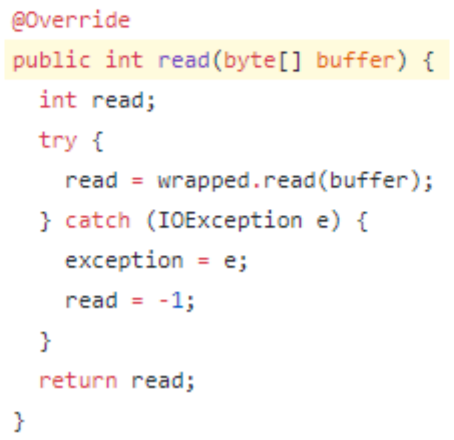


```
@Override
public int read() {
  int result;
  try {
    result = wrapped.read();
  } catch (IOException e) {
    exception = e;
    result = -1;
  }
  return result;
}
```

Marcar apenas uma oval.

	0	1	2	3	
Irrelevant	◯	◯	◯	◯	Highly relevant

35. 19. How relevant is the following code to the query "read*"? *

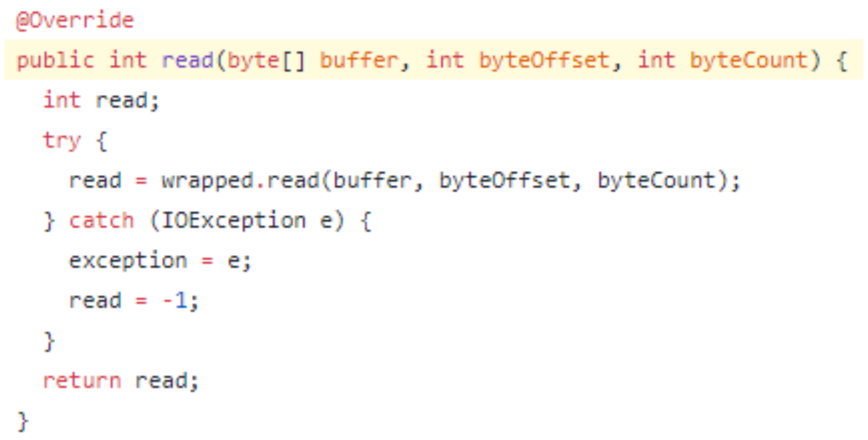


```
@Override
public int read(byte[] buffer) {
  int read;
  try {
    read = wrapped.read(buffer);
  } catch (IOException e) {
    exception = e;
    read = -1;
  }
  return read;
}
```

Marcar apenas uma oval.

	0	1	2	3	
Irrelevant	⬭	⬭	⬭	⬭	Highly relevant

36. 20. How relevant is the following code to the query "read*"? *



```
@Override
public int read(byte[] buffer, int byteOffset, int byteCount) {
  int read;
  try {
    read = wrapped.read(buffer, byteOffset, byteCount);
  } catch (IOException e) {
    exception = e;
    read = -1;
  }
  return read;
}
```

Marcar apenas uma oval.

	0	1	2	3	
Irrelevant	⬭	⬭	⬭	⬭	Highly relevant

Query	You are looking for a method that returns you the max within an array of ints. You look for the method name "max*" with arguments type: "int" and return type "int". Note: The '*' at the end of the word means that you are looking for all the entries that start

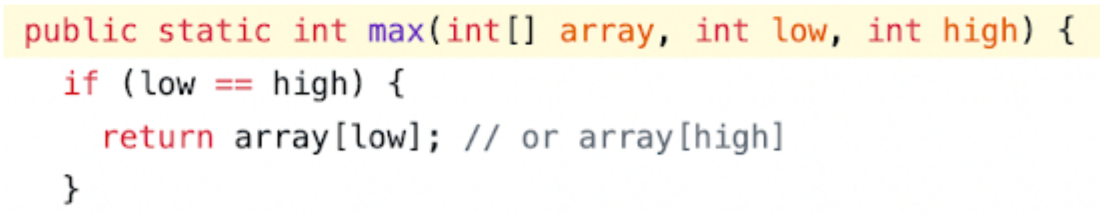
3 with that word. (i.e. you can also accept results such as maximum, maximin, etc.)

37. All the questions on this page refer to the above query. *

Marcar tudo o que for aplicável.

☐ I understand

38. 1. How relevant is the following code to the query "max*" with arguments type: "int" and return type "int". *

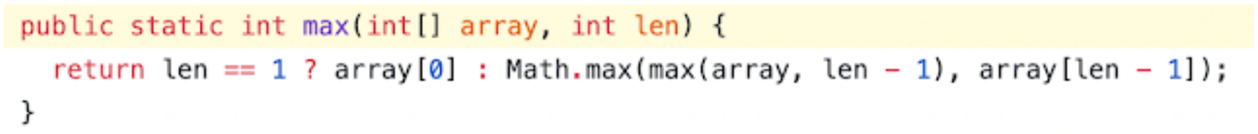


```
public static int max(int[] array, int low, int high) {
  if (low == high) {
    return array[low]; // or array[high]
  }
```

Marcar apenas uma oval.

	0	1	2	3	
Irrelevant	◯	◯	◯	◯	Highly relevant

39. 2. How relevant is the following code to the query "max*" with arguments type: "int" and return type "int". *

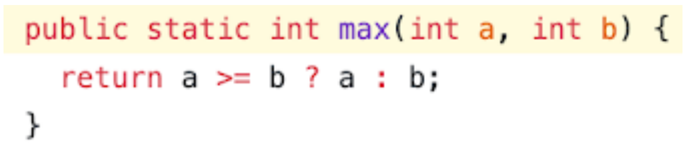


```
public static int max(int[] array, int len) {
   return len == 1 ? array[0] : Math.max(max(array, len - 1), array[len - 1]);
}
```

Marcar apenas uma oval.

	0	1	2	3	
Irrelevant	◯	◯	◯	◯	Highly relevant

40. 3. How relevant is the following code to the query "max*" with arguments type: "int" and return type "int". *

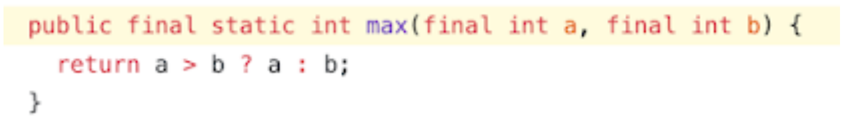


```
public static int max(int a, int b) {
   return a >= b ? a : b;
}
```

Marcar apenas uma oval.

	0	1	2	3	
Irrelevant	⬭	⬭	⬭	⬭	Highly relevant

41. 4. How relevant is the following code to the query "max*" with arguments type: "int" and return type "int". *

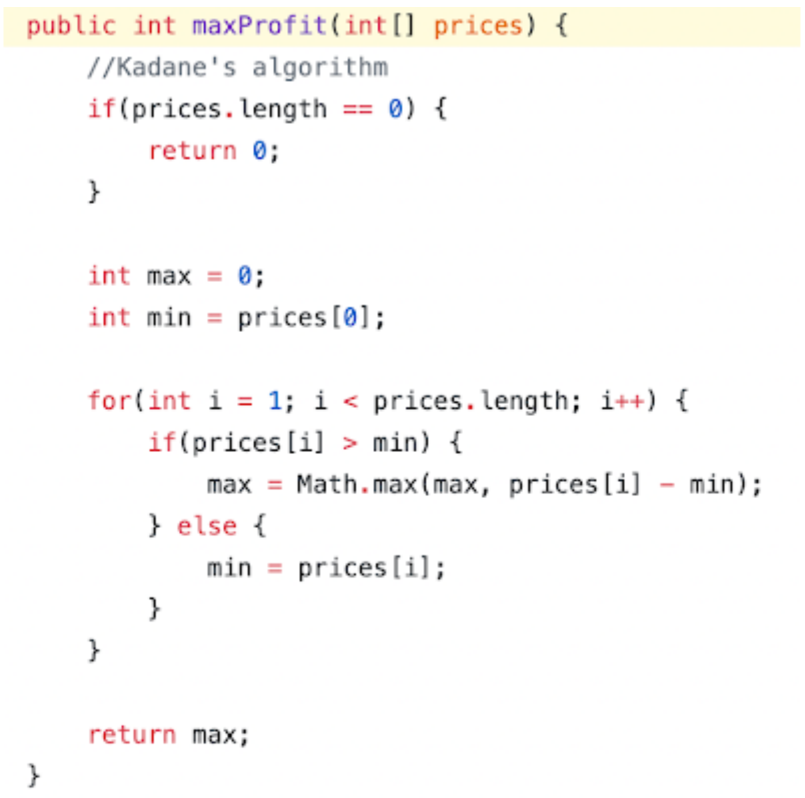


```
public final static int max(final int a, final int b) {
   return a > b ? a : b;
}
```

Marcar apenas uma oval.

	0	1	2	3	
Irrelevant	⬭	⬭	⬭	⬭	Highly relevant

42. 5. How relevant is the following code to the query "max*" with arguments type: "int" and return type "int". *

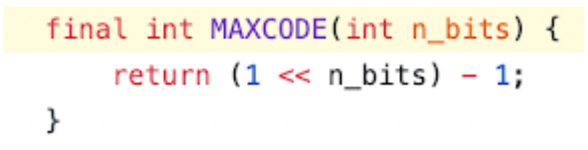


```
public int maxProfit(int[] prices) {
    //Kadane's algorithm
    if(prices.length == 0) {
        return 0;
    }

    int max = 0;
    int min = prices[0];

    for(int i = 1; i < prices.length; i++) {
        if(prices[i] > min) {
            max = Math.max(max, prices[i] - min);
        } else {
            min = prices[i];
        }
    }

    return max;
}
```

Marcar apenas uma oval.

	0	1	2	3	
Irrelevant	◯	◯	◯	◯	Highly relevant

43. 6. How relevant is the following code to the query "max*" with arguments type: "int" and return type "int". *

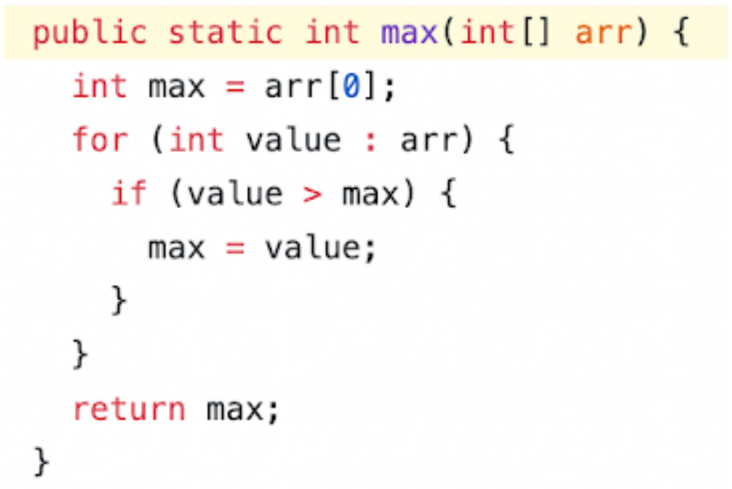


```
final int MAXCODE(int n_bits) {
    return (1 << n_bits) - 1;
}
```

Marcar apenas uma oval.

	0	1	2	3	
Irrelevant	◯	◯	◯	◯	Highly relevant

44. 7. How relevant is the following code to the query "max*" with arguments type: "int" and return type "int". *

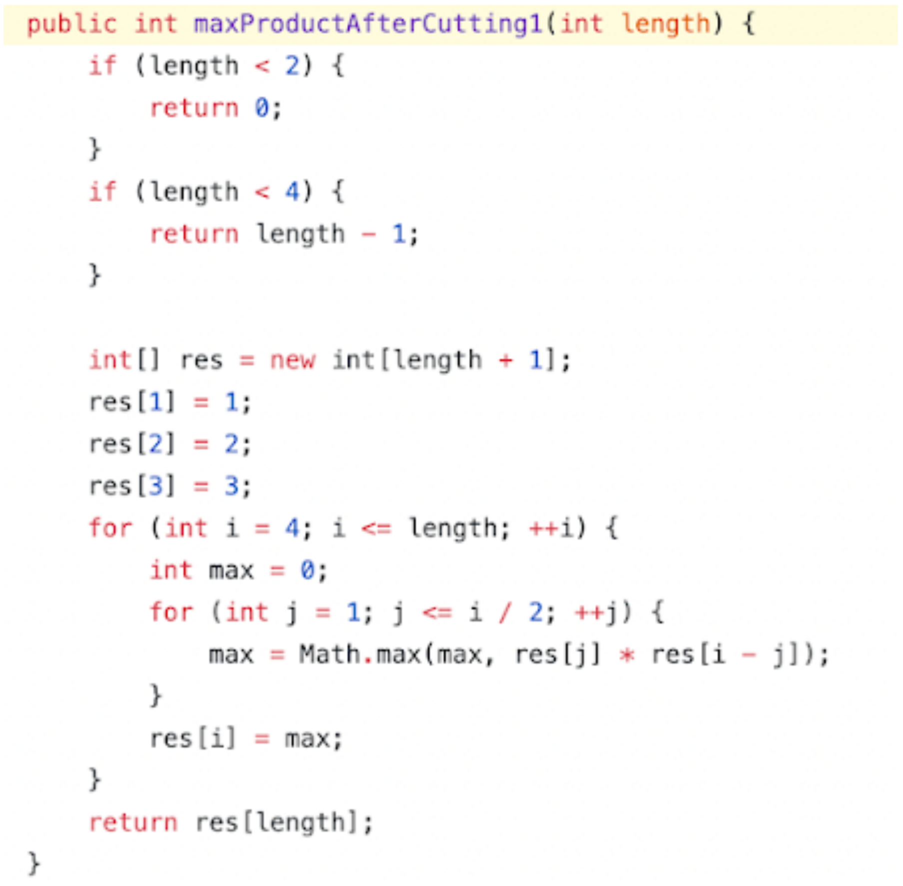


```
public static int max(int[] arr) {
    int max = arr[0];
    for (int value : arr) {
      if (value > max) {
        max = value;
      }
    }
    return max;
}
```

Marcar apenas uma oval.

	0	1	2	3	
Irrelevant	○	○	○	○	Highly relevant

45. 8. How relevant is the following code to the query "max*" with arguments type: "int" and return type "int". *

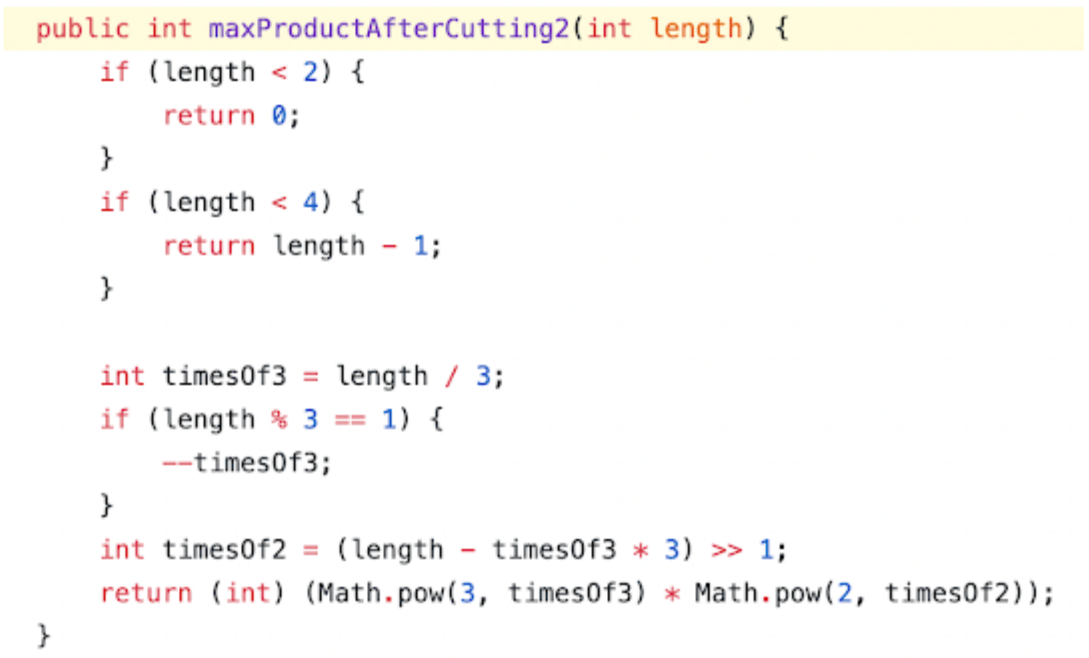


```java
public int maxProductAfterCutting1(int length) {
    if (length < 2) {
        return 0;
    }
    if (length < 4) {
        return length - 1;
    }

    int[] res = new int[length + 1];
    res[1] = 1;
    res[2] = 2;
    res[3] = 3;
    for (int i = 4; i <= length; ++i) {
        int max = 0;
        for (int j = 1; j <= i / 2; ++j) {
            max = Math.max(max, res[j] * res[i - j]);
        }
        res[i] = max;
    }
    return res[length];
}
```

Marcar apenas uma oval.

	0	1	2	3	
Irrelevant	◯	◯	◯	◯	Highly relevant

46. 9. How relevant is the following code to the query "max*" with arguments type: "int" and return type "int". *

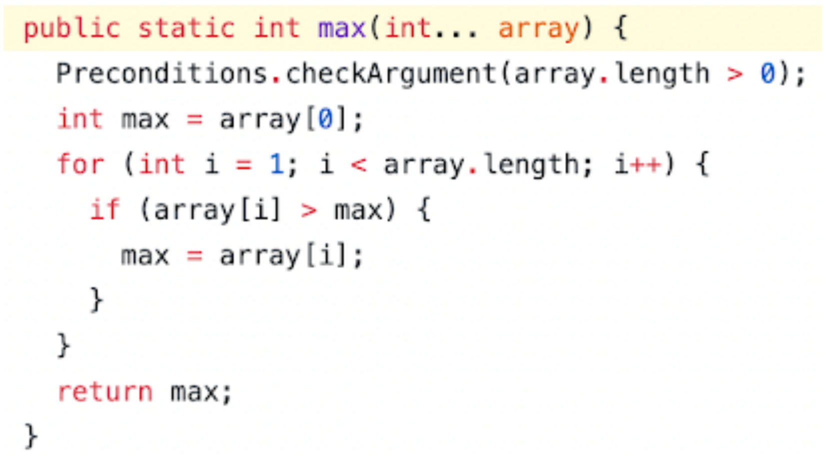


```java
public int maxProductAfterCutting2(int length) {
    if (length < 2) {
        return 0;
    }
    if (length < 4) {
        return length - 1;
    }

    int timesOf3 = length / 3;
    if (length % 3 == 1) {
        --timesOf3;
    }
    int timesOf2 = (length - timesOf3 * 3) >> 1;
    return (int) (Math.pow(3, timesOf3) * Math.pow(2, timesOf2));
}
```

Marcar apenas uma oval.

	0	1	2	3	
Irrelevant	◯	◯	◯	◯	Highly relevant

47. 10. How relevant is the following code to the query "max*" with arguments type: "int" and return type "int". *

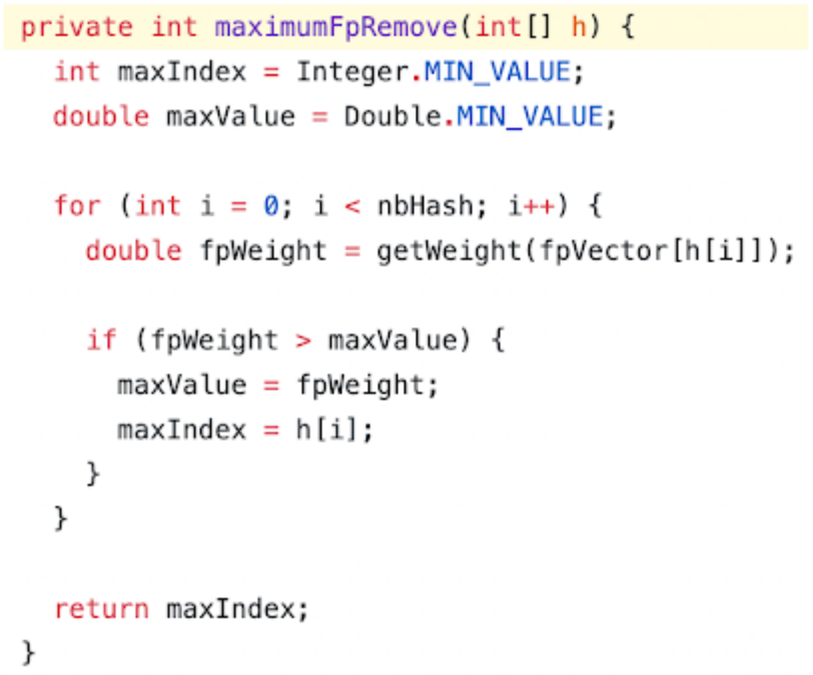


```java
public static int max(int... array) {
  Preconditions.checkArgument(array.length > 0);
  int max = array[0];
  for (int i = 1; i < array.length; i++) {
    if (array[i] > max) {
      max = array[i];
    }
  }
  return max;
}
```

Marcar apenas uma oval.

	0	1	2	3	
Irrelevant	◯	◯	◯	◯	Highly relevant

48. 11. How relevant is the following code to the query "max*" with arguments type: "int" and return type "int". *

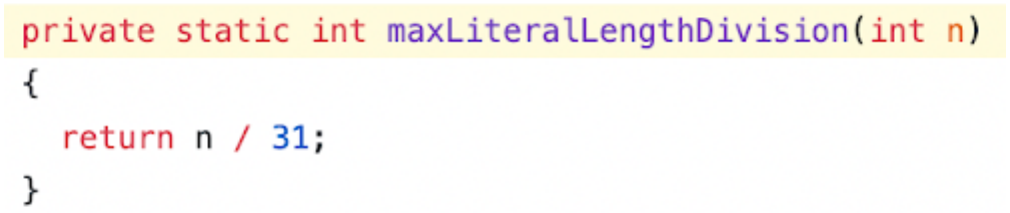


```java
private int maximumFpRemove(int[] h) {
  int maxIndex = Integer.MIN_VALUE;
  double maxValue = Double.MIN_VALUE;

  for (int i = 0; i < nbHash; i++) {
    double fpWeight = getWeight(fpVector[h[i]]);

    if (fpWeight > maxValue) {
      maxValue = fpWeight;
      maxIndex = h[i];
    }
  }

  return maxIndex;
}
```

Marcar apenas uma oval.

	0	1	2	3	
Irrelevant	◯	◯	◯	◯	Highly relevant

49. 12. How relevant is the following code to the query "max*" with arguments type: "int" and return type "int". *

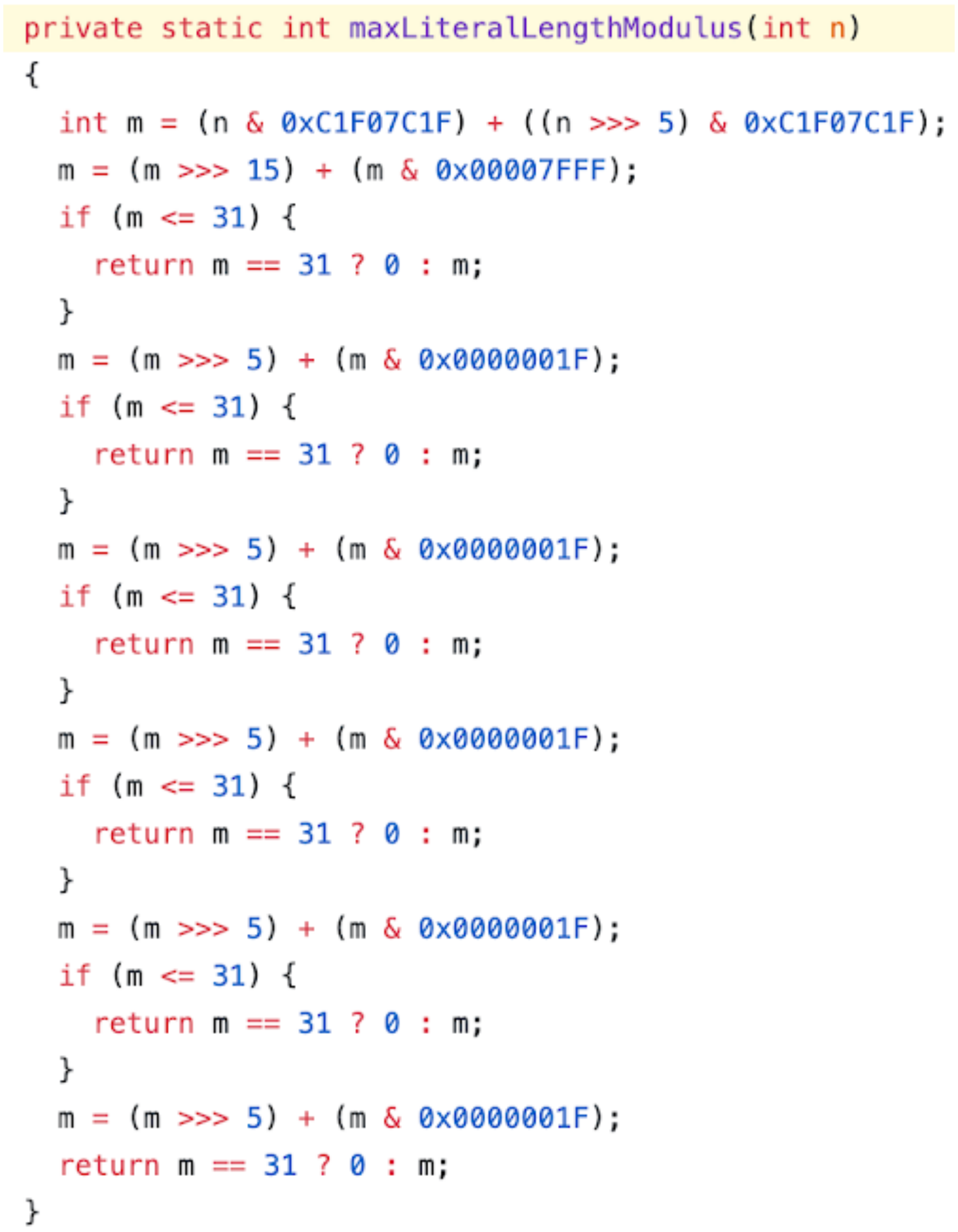


```java
private static int maxLiteralLengthDivision(int n)
{
  return n / 31;
}
```

Marcar apenas uma oval.

	0	1	2	3	
Irrelevant	◯	◯	◯	◯	Highly relevant

50. 13. How relevant is the following code to the query "max*" with arguments type: "int" and return type "int". *

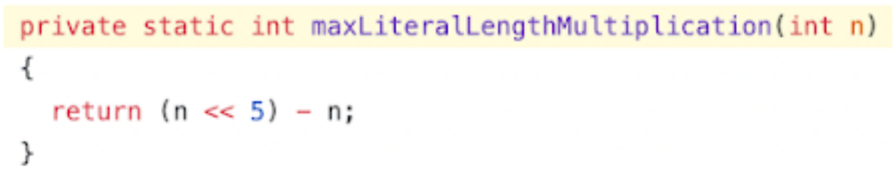


```java
private static int maxLiteralLengthModulus(int n)
{
  int m = (n & 0xC1F07C1F) + ((n >>> 5) & 0xC1F07C1F);
  m = (m >>> 15) + (m & 0x00007FFF);
  if (m <= 31) {
    return m == 31 ? 0 : m;
  }
  m = (m >>> 5) + (m & 0x0000001F);
  if (m <= 31) {
    return m == 31 ? 0 : m;
  }
  m = (m >>> 5) + (m & 0x0000001F);
  if (m <= 31) {
    return m == 31 ? 0 : m;
  }
  m = (m >>> 5) + (m & 0x0000001F);
  if (m <= 31) {
    return m == 31 ? 0 : m;
  }
  m = (m >>> 5) + (m & 0x0000001F);
  if (m <= 31) {
    return m == 31 ? 0 : m;
  }
  m = (m >>> 5) + (m & 0x0000001F);
  return m == 31 ? 0 : m;
}
```

Marcar apenas uma oval.

	0	1	2	3	
Irrelevant	○	○	○	○	Highly relevant

51. 14. How relevant is the following code to the query "max*" with arguments type: "int" and return type "int". *

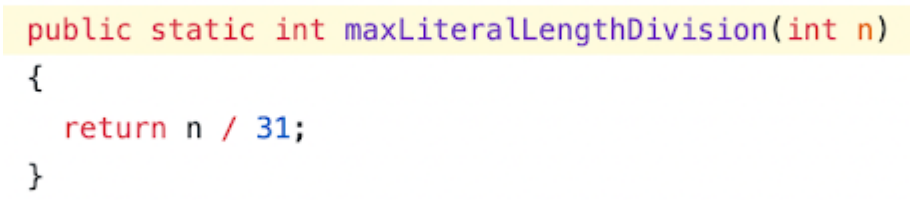


```
private static int maxLiteralLengthMultiplication(int n)
{
  return (n << 5) - n;
}
```

Marcar apenas uma oval.

	0	1	2	3	
Irrelevant	○	○	○	○	Highly relevant

52. 15. How relevant is the following code to the query "max*" with arguments type: "int" and return type "int". *

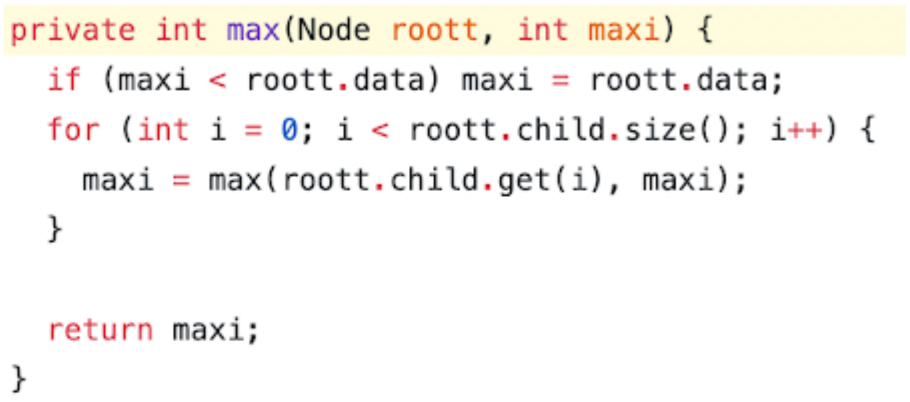


```
public static int maxLiteralLengthDivision(int n)
{
  return n / 31;
}
```

Marcar apenas uma oval.

	0	1	2	3	
Irrelevant	○	○	○	○	Highly relevant

53. 16. How relevant is the following code to the query "max*" with arguments type: "int" and return type "int". *

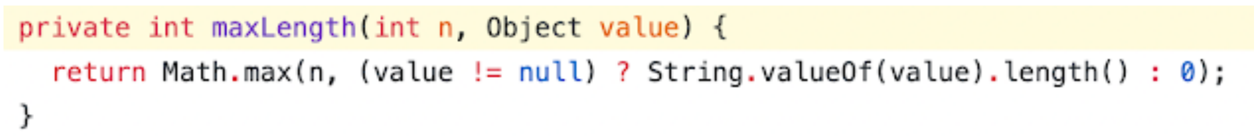


```java
private int max(Node roott, int maxi) {
  if (maxi < roott.data) maxi = roott.data;
  for (int i = 0; i < roott.child.size(); i++) {
    maxi = max(roott.child.get(i), maxi);
  }

  return maxi;
}
```

Marcar apenas uma oval.

	0	1	2	3	
Irrelevant	⬭	⬭	⬭	⬭	Highly relevant

54. 17. How relevant is the following code to the query "max*" with arguments type: "int" and return type "int". *

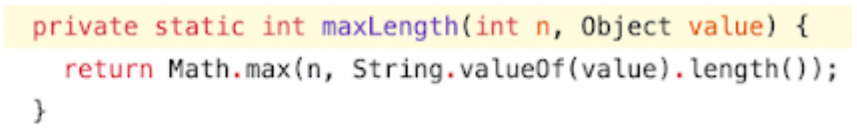


```java
private int maxLength(int n, Object value) {
  return Math.max(n, (value != null) ? String.valueOf(value).length() : 0);
}
```

Marcar apenas uma oval.

	0	1	2	3	
Irrelevant	⬭	⬭	⬭	⬭	Highly relevant

55. 18. How relevant is the following code to the query "max*" with arguments type: "int" and return type "int". *

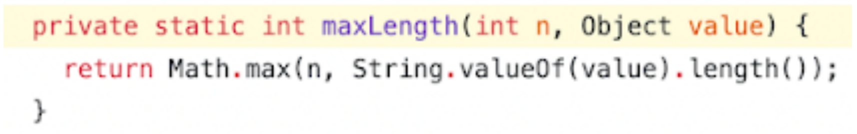


```
private static int maxLength(int n, Object value) {
  return Math.max(n, String.valueOf(value).length());
}
```

Marcar apenas uma oval.

	0	1	2	3	
Irrelevant	⬭	⬭	⬭	⬭	Highly relevant

56. 19. How relevant is the following code to the query "max*" with arguments type: "int" and return type "int". *

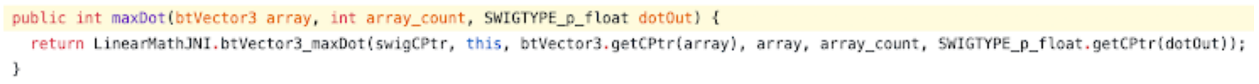


```
private static int maxLength(int n, Object value) {
  return Math.max(n, String.valueOf(value).length());
}
```

Marcar apenas uma oval.

	0	1	2	3	
Irrelevant	⬭	⬭	⬭	⬭	Highly relevant

57. 20. How relevant is the following code to the query "max*" with arguments type: "int" and return type "int". *



```
public int maxDot(btVector3 array, int array_count, SWIGTYPE_p_float dotOut) {
  return LinearMathJNI.btVector3_maxDot(swigCPtr, this, btVector3.getCPtr(array), array, array_count, SWIGTYPE_p_float.getCPtr(dotOut));
}
```

Marcar apenas uma oval.

	0	1	2	3	
Irrelevant	⬭	⬭	⬭	⬭	Highly relevant