

1.

[BOS] Block until the latch is counted down and return the error received or when the wait is interrupted or times out, null otherwise. [EOS]

```

0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24
public Throwable blockingGetError(Long timeout, TimeUnit unit) {
    if (getCount() != 0) { Predicted: 2 5 6 7
        try { Predicted: 8
            BlockingHelper.verifyNonBlocking(); Predicted: 1
            if (!await(timeout, unit)) { Predicted: 3 4 15 16 18 20 21
                dispose(); Predicted: 19
                throw ExceptionHelper.wrapOrThrow(new TimeoutException(timeoutMessage(timeout, unit))); Predicted: 11
            }
        } catch (InterruptedException ex) { Predicted: 13 14 17 23
            dispose(); Predicted: 12
            throw ExceptionHelper.wrapOrThrow(ex); Predicted: 22
        }
    }
    return error; Predicted: 9 10 11
}

```

2.

[BOS] Tries to add the given subscriber to the subscribers array automatically or returns false if the subject has terminated. [EOS]

```

0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
boolean add(PublishDisposable<T> ps) { Predicted: 3 16
    for (;;) {
        PublishDisposable<T>[] a = subscribers.get(); Predicted: 5 6
        if (a == TERMINATED) { Predicted: 12 15 17 18 19
            return false; Predicted: 13 14
        }
        int n = a.length;
        @SuppressWarnings("unchecked")
        PublishDisposable<T>[] b = new PublishDisposable[n + 1];
        System.arraycopy(a, 0, b, 0, n);
        b[n] = ps; Predicted: 2 4 5 6 7 8 9 10 11
        if (subscribers.compareAndSet(a, b)) { Predicted: 1
            return true;
        }
    }
}

```

3.

Copies max or less number of bytes to output stream.

```

1   2   3   4   5   6   7   8   9  10
public int copyMax(final OutputStream out, final int maxBytes) throws IOException {
    int count = 0;
    while (true) {
        byte b = readByte(); 5 6 7
        if (isBoundary(b)) {
            break;
        }
        out.write(b); 1 8 9 10
        count++;
        if (count == maxBytes) { 2
            return count;
        }
    }
    return count; 3 4
}

```

4.

Checks if some parameter is in GET parameters.

1 2 3 4 5 6 7 8

```
public boolean isGetParameter(final HttpServletRequest request, String name) {  
    name = URLEncoder.encodeQueryParam(name) + '='; 3 4  
    String query = request.getQueryString(); 7 8  
    String[] nameValuePairs = StringUtil.split(query, '&');  
    for (String nameValuePair : nameValuePairs) {  
        if (nameValuePair.startsWith(name)) { 1 2 6  
            return true; 5  
        }  
    }  
    return false;  
}
```

5.

Reads HTTP request body using the request stream.

1 2 3 4 5 6 7 8

```
public static String readRequestBodyFromStream(final HttpServletRequest request) throws IOException {  
    String charEncoding = request.getCharacterEncoding();  
    if (charEncoding == null) {  
        charEncoding = JoddCore.encoding;  
    }  
    CharArrayWriter charArrayWriter = new CharArrayWriter();  
    BufferedReader bufferedReader = null;  
    try {  
        InputStream inputStream = request.getInputStream(); 6 7 8  
        if (inputStream != null) {  
            bufferedReader = new BufferedReader(new InputStreamReader(inputStream, charEncoding)); 5  
            StreamUtil.copy(bufferedReader, charArrayWriter); 1  
        } else {  
            return StringPool.EMPTY;  
        }  
    } finally {  
        StreamUtil.close(bufferedReader);  
    }  
    return charArrayWriter.toString(); 2 3 4  
}
```

6.

Prepares response for file download with provided mime type.

1 2 3 4 5 6 7 8 9

```
public static void prepareDownload(final HttpServletResponse response, final File file, final String mimeType) { 7 8 9  
    if (!file.exists()) { 6  
        throw new IllegalArgumentException("File not found: " + file);  
    }  
    if (file.length() > Integer.MAX_VALUE) {  
        throw new IllegalArgumentException("File too big: " + file); 3  
    }  
    prepareResponse(response, file.getAbsolutePath(), mimeType, (int) file.length()); 1 2 4 5  
}
```

7.

Extracts encoding from a given content type.

1 2 3 4 5 6 7

```
public static String extractEncoding(final String contentType, String defaultEncoding) { 5
    String encoding = extractEncoding(contentType); 1 3 6 7

    if (encoding == null) { 4
        if (defaultEncoding == null) {
            defaultEncoding = JoddCore.encoding;
        }
        encoding = defaultEncoding;
    }
    return encoding; 2
}
```

8.

Generates new CSRF token and puts it in the session. Returns generated token value.

1 2 3 4 5 6 7 8 9 10 11 12 13 14

```
public static String prepareCsrfToken(final HttpSession session, final int timeToLive) {
    Set<Token> tokenSet = (Set<Token>) session.getAttribute(CSRF_TOKEN_SET); 9 10
    if (tokenSet == null) {
        tokenSet = new HashSet<>(); 1
        session.setAttribute(CSRF_TOKEN_SET, tokenSet); 2 3 4
    }
    String value;
    boolean unique;
    do {
        value = RandomString.get().randomAlphaNumeric(32); 12 13
        assureSize(tokenSet);
        unique = tokenSet.add(new Token(value, timeToLive)); 5 6 7 8
    } while (!unique);
    return value; 11 14
}
```

9.

Performs search for the scope class and returns it's instance.

1 2 3 4 5 6 7 8 9 10

```
protected MadvocScope getOrInitScope(final Class<? extends MadvocScope> madvocScopeType) { 4 5 6
    for (final MadvocScope s : allScopes) { 3
        if (s.getClass().equals(madvocScopeType)) { 1 2
            return s;
        }
    }

    // new scope detected
    final MadvocScope newScope;
    try {
        newScope = madpc.createBean(madvocScopeType); 9 10
    } catch (Exception ex) {
        throw new MadvocException("Unable to create scope: " + madvocScopeType, ex);
    }

    allScopes.add(newScope); 7

    return newScope; 8
}
```

10.

Converts all separators to the system separator.

1 2 3 4 5 6 7

```
public static String separatorsToSystem(final String path) { 3
    if (path == null) { 2
        return null;
    }
    if (SYSTEM_SEPARATOR == WINDOWS_SEPARATOR) {
        return separatorsToWindows(path); 4 5 6
    } else {
        return separatorsToUnix(path); 1 7
    }
}
```

11.

Converts property name to column name.

1 2 3 4 5 6

```
public String convertPropertyNameToColumnName(final String propertyName) { 2 3
    StringBuilder tableName = new StringBuilder(propertyName.length() * 2);

    if (splitCamelCase) {
        String convertedTableName = Format.fromCamelCase(propertyName, separatorChar); 1
        tableName.append(convertedTableName);
    } else {
        tableName.append(propertyName); 5 6
    }

    if (!changeCase) {
        return tableName.toString();
    }
    return uppercase ?
        toUpperCase(tableName).toString() : 4
        toLowerCase(tableName).toString();
}
```

12.

Creates a new clock log object using the specified message

1 2 3 4 5 6 7 8 9 10

```
public clock logtime ( string message ) {
    super . split ( ) ;
    string time = durationformatutils . formatdurationhms ( super . getsplittime ( ) ) ; 7
    string msg = message + separator + time ; 8 9 10
    if ( log != null ) {
        log . info ( msg ) ; 1 2 3
    }
    else {
        defaultlog . info ( msg ) ; 4 5 6
    }
    return this ;
}
```

13.

Add a new shard to the list at the specified index

1 2 3 4 5 6 7 8 9 10 11

```
public builder addshard ( indexshardroutingtable refdata , shardrouting shard ) { 6 7
    indexshardroutingtable indexshard = shards . get ( shard . id ( ) ) ; 2 3 4 5
    if ( indexshard == null ) {
        indexshard = new indexshardroutingtable . builder ( refdata . shardid ( ) ) .
        addshard ( new shardrouting ( shard ) ) . build ( ) ;
    }
    else {
        indexshard = new indexshardroutingtable . builder ( indexshard ) . addshard ( 1
        new shardrouting ( shard ) ) . build ( ) ;
    }
    shards . put ( indexshard . shardid ( ) . id ( ) , indexshard ) ; 8 9 10 11
    return this ;
}
```

14.

writes the xml attribute value to the xml file

1 2 3 4 5 6 7 8 9

```
private void writeattribute ( java . lang . string prefix , java . lang . string namespace , java
. lang . string attname , java . lang . string attvalue , javax . xml . stream . xmlstreamwriter 2 3 4 5
xmlwriter ) throws javax . xml . stream . xmlstreamexception { 7 8 9
    if ( xmlwriter . getprefix ( namespace ) == null ) {
        xmlwriter . writenamespace ( prefix , namespace ) ;
        xmlwriter . setprefix ( prefix , namespace ) ;
    }
    xmlwriter . writeattribute ( namespace , attname , attvalue ) ; 1 6
}
```

15.

Process the given value

1 2 3 4

```
private string processfunction ( string value , variables variables ) { 3 4
    string returnvalue = value ;
    if ( functionhandler . validfunction ( returnvalue ) ) {
        returnvalue = functionhandler . executefunction ( returnvalue , variables ) ; 1 2
    }
    return returnvalue != null ? returnvalue : str_ ;
}
```

16.

Saves the movie in the database

1 2 3 4 5 6

```
public void savemovie ( int type ) throws moviesaveexception {
    currentframe = num_ ;
    string desc = null ;
    if ( type == movieinfoprovider . type_quicktime_jpeg ) {
        createmovformat ( ) ;
        desc = filetypedescriptor . quicktime ; 2 3
    }
    else if ( type == movieinfoprovider . type_avi_mjpeg ) {
        createjpegformat ( ) ;
        desc = filetypedescriptor . msvideo ;
    }
    else if ( type == movieinfoprovider . type_avi_raw ) {
        creatergbformat ( ) ;
        desc = filetypedescriptor . msvideo ; 4 5 6
    }
    else throw new unsupportedoperationexception ( str_ ) ;
    try {
        itm . savemovie ( mip . getmedialocator ( ) , desc , videoformat ) ; 1
    }
    catch ( exception e ) {
        throw new moviesaveexception ( e ) ;
    }
}
```

17.

sort the array in ascending order of collection sizes

1 2 3 4 5 6 7 8 9

```
public int compare ( object obj1 , object obj2 ) {
    if ( ! ( obj1 instanceof selectresults ) || ! ( obj2 instanceof selectresults ) ) {
        support . assertionfailed ( str_ ) ;
    }
    int answer = - num_ ;
    selectresults sr1 = ( selectresults ) obj1 ; 2 3
    selectresults sr2 = ( selectresults ) obj2 ;
    int sizedifference = sr1 . size ( ) - sr2 . size ( ) ; 8 9
    if ( obj1 == obj2 ) { answer = num_ ; } 7
    else if ( sizedifference > num_ ) { answer = num_ ; } 1 4 5 6
}
```

18.

tests if string value is hex value

1 2 3 4 5 6 7

```
public static boolean ishex ( string str ) {
    if ( str == null || str . length ( ) == num_ ) return bool_ ;
    for ( int i = str . length ( ) - num_ ; i >= num_ ; i -- ) {
        char c = str . charat ( i ) ;
        if ( ! ( c >= str_ && c <= str_ ) ) { 1 2
            c = character . tolowercase ( c ) ; 3 4
            if ( ! ( c == str_ || c == str_ || c == str_ || c == str_ || c == str_ ) ) 5 6 7
                return bool_ ;
        }
    }
    return bool_ ;
}
```

19.

generates a custom name containing numbers and an underscore ex .

1 2 3 4 5 6 7 8 9 10

```
public static string generatecustomname ( ) {
    random random = new random ( ) ; 1 2
    stringbuilder sb = new stringbuilder ( ) ; 3 4
    sb . append ( propertyconverter . getcurrenttimesecs ( ) ) ; 7
    sb . append ( str_ ) ; 8 9 10
    int i = random . nextint ( num_ ) ;
    if ( i < num_ ) { sb . append ( str_ ) ; }
    else if ( i < num_ ) { sb . append ( str_ ) ; }
    else if ( i < num_ ) { sb . append ( str_ ) ; }
    else if ( i < num_ ) { sb . append ( str_ ) ; }
    sb . append ( i ) ; 5 6
    return sb . toString ( ) ;
}
```

20.

Add a message destination reference for this web application

1 2 3 4 5 6 7 8 9

```
public void addmessagedestinationref ( messagedestinationref mdr ) {
    if ( entries.contains(mdr.getname()) ) {return ;}
    else {
        if ( !checkresourcetype(mdr) ) {
            throw new illegalargumentexception(sm.getstring(str_ , mdr.getname( ) , mdr.gettype())) ;
        }
        entries.add(mdr.getname()) ; 1 7 8 9
    }
    synchronized ( mdrs ) {
        mdr.setnamingresources(this); 6
        mdrs.put(mdr.getname( ) , mdr) ; 2 3 4
    }
    support.firepropertychange ( str_ , null , mdr ) ; 5
}
```