GÖTEBORGS UNIVERSITET

Welcome

Welcome

We are two students writing our bachelor thesis, in which we have designed a survey to test the comprehension of code in software. The collected data in this survey will be anonymized, and we are unable to trace the answers to you, provided that you exclude identifiable data from your answers. By continuing in this survey, you accept to submitting data in the form of the answers that you give.

The survey can be answered on both phones and computers, we do however recommend that you answer the survey on a computer for a better experience.
If you have any questions, feel free to contact us via email.

Christofer Jidarv (gusjidach@student.gu.se)
Robin Hansen (gushanrod@student.gu.se)

Initial Questions

How many years of java experience do you have?

- ○ Under 1 year
- ○ 1-2 years
- ○ 2-3 years
- ○ 3-4 years
- ○ 4+ years

How many years of coding related studies have you completed?

- Under 1 year
- 1-2 years
- 2-3 years
- 3-4 years
- 4+ years

Do you have some form of reading disorder (e.g. dyslexia?)

- Yes
- No

GC

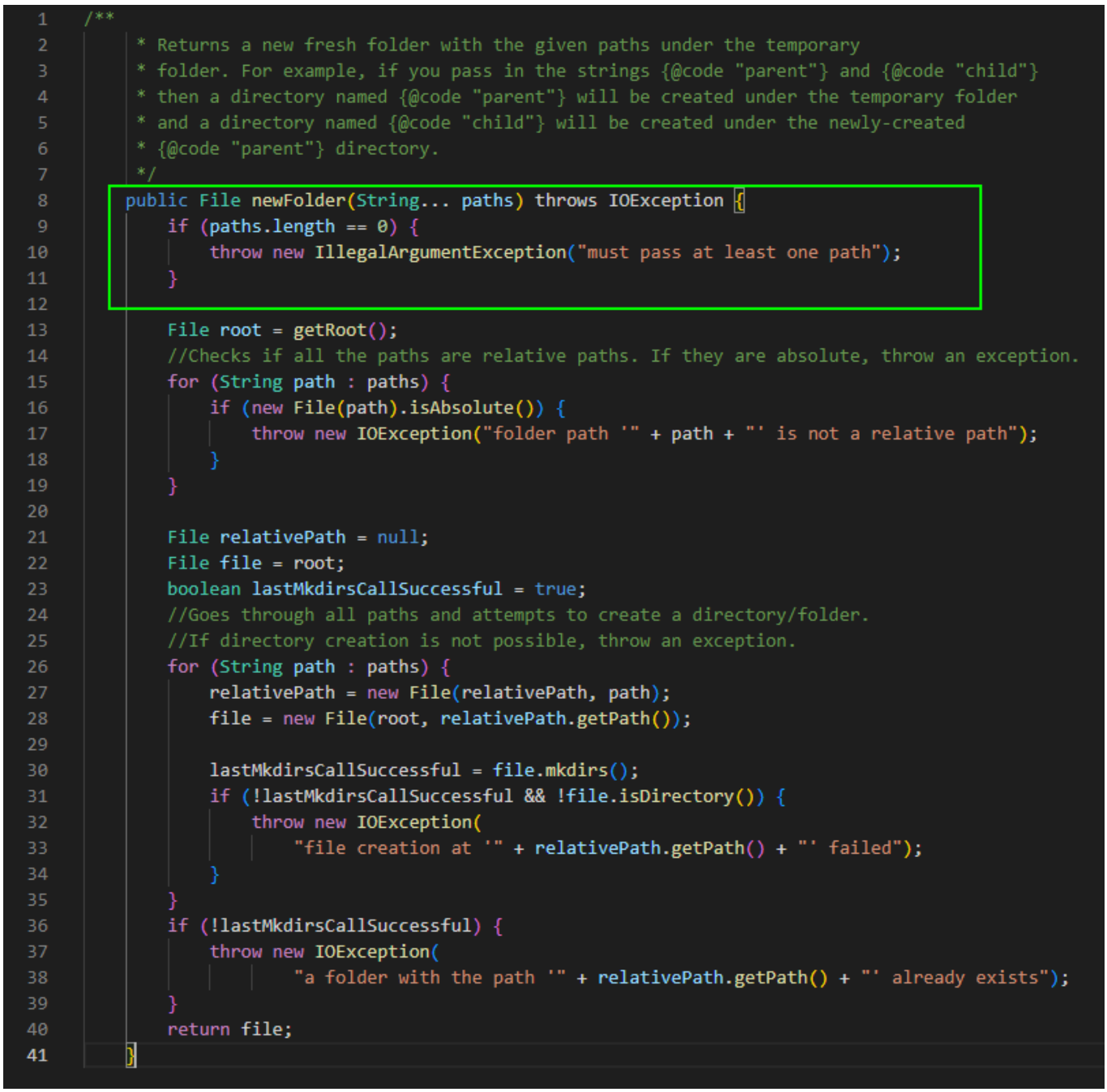
Read this code snippet carefully, there is not any time limit and when you think you understand the code fully, then please press continue.

```java
/**
 * Returns a new fresh folder with the given paths under the temporary
 * folder. For example, if you pass in the strings {@code "parent"} and {@code "child"}
 * then a directory named {@code "parent"} will be created under the temporary folder
 * and a directory named {@code "child"} will be created under the newly-created
 * {@code "parent"} directory.
 */
public File newFolder(String... paths) throws IOException {
    if (paths.length == 0) {
        throw new IllegalArgumentException("must pass at least one path");
    }

    File root = getRoot();
    //Checks if all the paths are relative paths. If they are absolute, throw an exception.
    for (String path : paths) {
        if (new File(path).isAbsolute()) {
            throw new IOException("folder path '" + path + "' is not a relative path");
        }
    }

    File relativePath = null;
    File file = root;
    boolean lastMkdirsCallSuccessful = true;
    //Goes through all paths and attempts to create a directory/folder.
    //If directory creation is not possible, throw an exception.
    for (String path : paths) {
        relativePath = new File(relativePath, path);
        file = new File(root, relativePath.getPath());

        lastMkdirsCallSuccessful = file.mkdirs();
        if (!lastMkdirsCallSuccessful && !file.isDirectory()) {
            throw new IOException(
                "file creation at '" + relativePath.getPath() + "' failed");
        }
    }
    if (!lastMkdirsCallSuccessful) {
        throw new IOException(
                "a folder with the path '" + relativePath.getPath() + "' already exists");
    }
    return file;
}
```

Based on your programming experience, how would you rate your comprehension of the previous piece of code?

- ○ Very Difficult
- ○ Difficult
- ○ Neutral
- ○ Easy
- ○ Very Easy

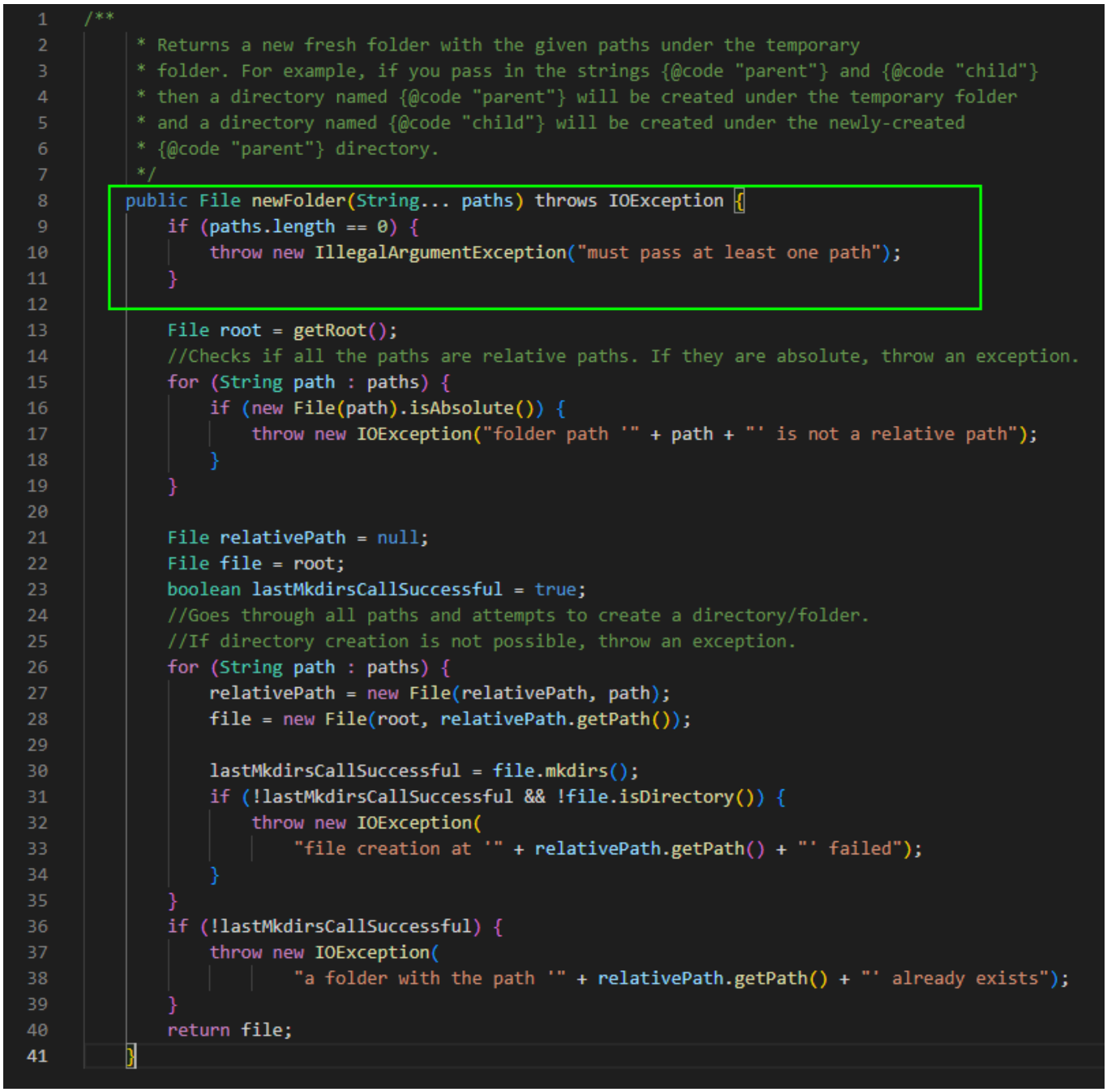
How would you rate your comprehension of the marked piece of code?



```java
/**
     * Returns a new fresh folder with the given paths under the temporary
     * folder. For example, if you pass in the strings {@code "parent"} and {@code "child"}
     * then a directory named {@code "parent"} will be created under the temporary folder
     * and a directory named {@code "child"} will be created under the newly-created
     * {@code "parent"} directory.
     */
    public File newFolder(String... paths) throws IOException {
        if (paths.length == 0) {
            throw new IllegalArgumentException("must pass at least one path");
        }

        File root = getRoot();
        //Checks if all the paths are relative paths. If they are absolute, throw an exception.
        for (String path : paths) {
            if (new File(path).isAbsolute()) {
                throw new IOException("folder path '" + path + "' is not a relative path");
            }
        }

        File relativePath = null;
        File file = root;
        boolean lastMkdirsCallSuccessful = true;
        //Goes through all paths and attempts to create a directory/folder.
        //If directory creation is not possible, throw an exception.
        for (String path : paths) {
            relativePath = new File(relativePath, path);
            file = new File(root, relativePath.getPath());

            lastMkdirsCallSuccessful = file.mkdirs();
            if (!lastMkdirsCallSuccessful && !file.isDirectory()) {
                throw new IOException(
                    "file creation at '" + relativePath.getPath() + "' failed");
            }
        }
        if (!lastMkdirsCallSuccessful) {
            throw new IOException(
                    "a folder with the path '" + relativePath.getPath() + "' already exists");
        }
        return file;
    }
```

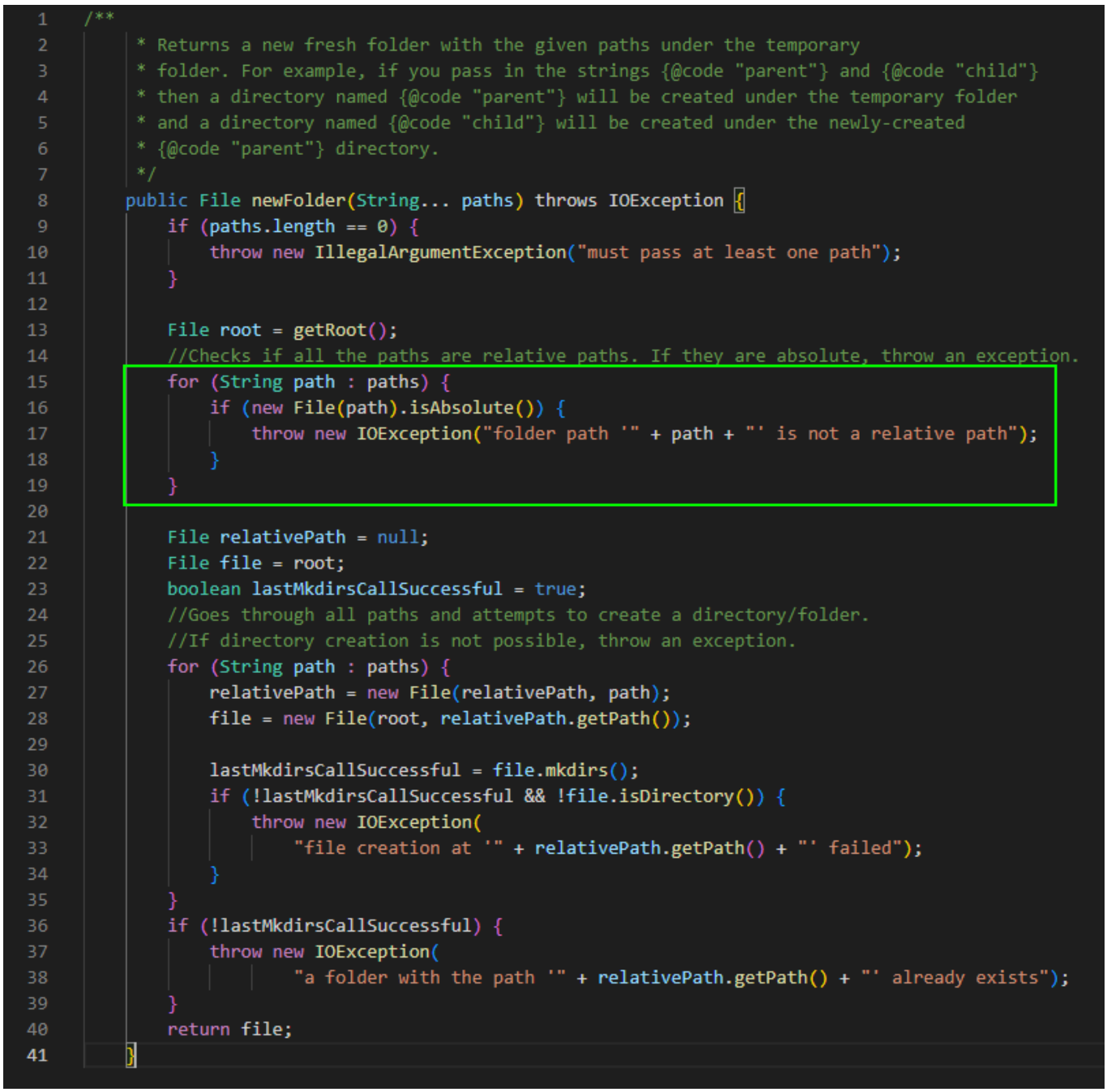
- ○ Very Difficult
- ○ Difficult
- ○ Neutral
- ○ Easy
- ○ Very Easy

What part of the marked piece of code do you find hard to understand?



```java
/**
 * Returns a new fresh folder with the given paths under the temporary
 * folder. For example, if you pass in the strings {@code "parent"} and {@code "child"}
 * then a directory named {@code "parent"} will be created under the temporary folder
 * and a directory named {@code "child"} will be created under the newly-created
 * {@code "parent"} directory.
 */
public File newFolder(String... paths) throws IOException {
    if (paths.length == 0) {
        throw new IllegalArgumentException("must pass at least one path");
    }

    File root = getRoot();
    //Checks if all the paths are relative paths. If they are absolute, throw an exception.
    for (String path : paths) {
        if (new File(path).isAbsolute()) {
            throw new IOException("folder path '" + path + "' is not a relative path");
        }
    }

    File relativePath = null;
    File file = root;
    boolean lastMkdirsCallSuccessful = true;
    //Goes through all paths and attempts to create a directory/folder.
    //If directory creation is not possible, throw an exception.
    for (String path : paths) {
        relativePath = new File(relativePath, path);
        file = new File(root, relativePath.getPath());

        lastMkdirsCallSuccessful = file.mkdirs();
        if (!lastMkdirsCallSuccessful && !file.isDirectory()) {
            throw new IOException(
                "file creation at '" + relativePath.getPath() + "' failed");
        }
    }
    if (!lastMkdirsCallSuccessful) {
        throw new IOException(
                "a folder with the path '" + relativePath.getPath() + "' already exists");
    }
    return file;
}
```

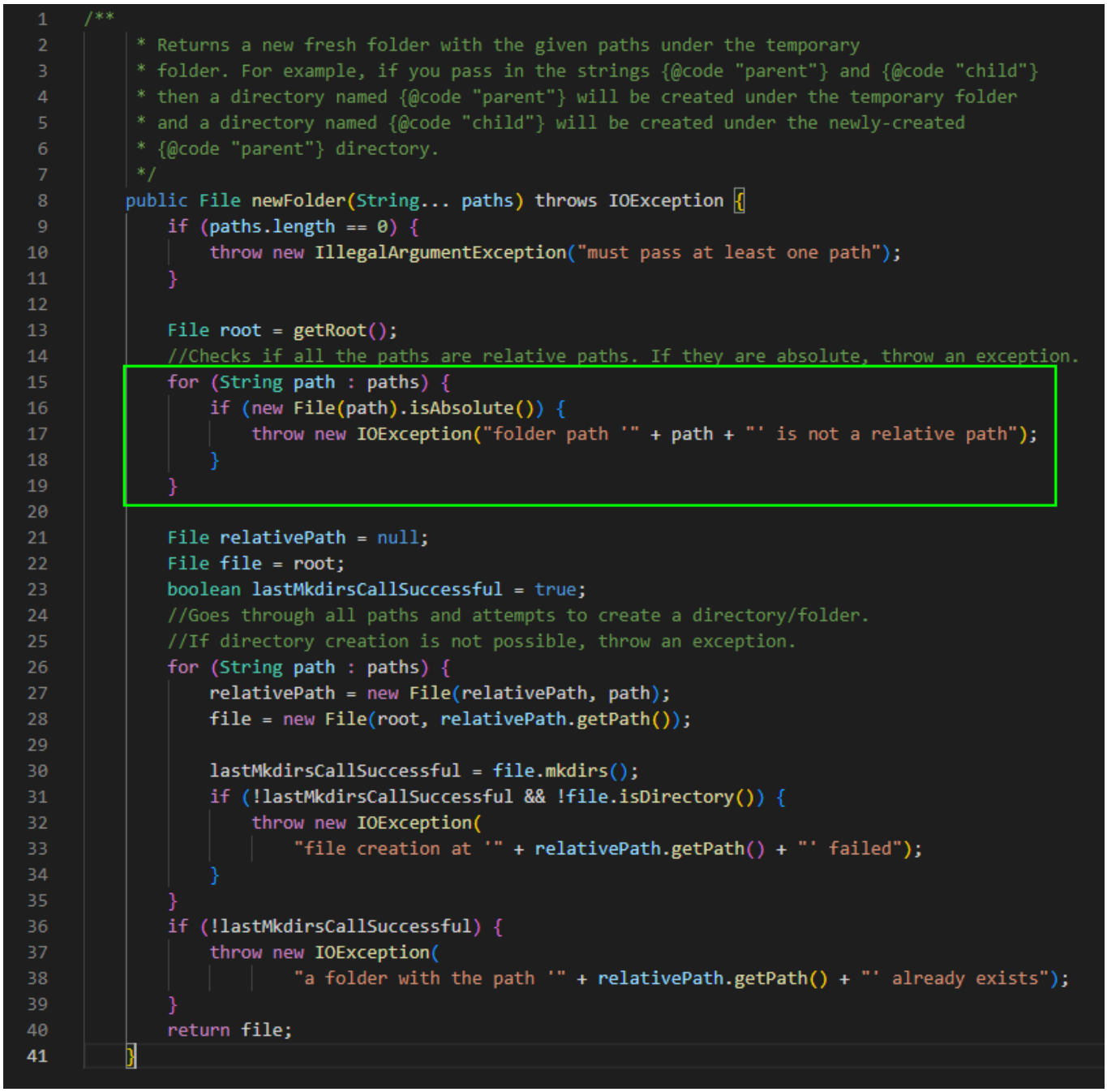
How would you rate your comprehension of the marked piece of code?



```java
/**
 * Returns a new fresh folder with the given paths under the temporary
 * folder. For example, if you pass in the strings {@code "parent"} and {@code "child"}
 * then a directory named {@code "parent"} will be created under the temporary folder
 * and a directory named {@code "child"} will be created under the newly-created
 * {@code "parent"} directory.
 */
public File newFolder(String... paths) throws IOException {
    if (paths.length == 0) {
        throw new IllegalArgumentException("must pass at least one path");
    }

    File root = getRoot();
    //Checks if all the paths are relative paths. If they are absolute, throw an exception.
    for (String path : paths) {
        if (new File(path).isAbsolute()) {
            throw new IOException("folder path '" + path + "' is not a relative path");
        }
    }

    File relativePath = null;
    File file = root;
    boolean lastMkdirsCallSuccessful = true;
    //Goes through all paths and attempts to create a directory/folder.
    //If directory creation is not possible, throw an exception.
    for (String path : paths) {
        relativePath = new File(relativePath, path);
        file = new File(root, relativePath.getPath());

        lastMkdirsCallSuccessful = file.mkdirs();
        if (!lastMkdirsCallSuccessful && !file.isDirectory()) {
            throw new IOException(
                "file creation at '" + relativePath.getPath() + "' failed");
        }
    }
    if (!lastMkdirsCallSuccessful) {
        throw new IOException(
                "a folder with the path '" + relativePath.getPath() + "' already exists");
    }
    return file;
}
```

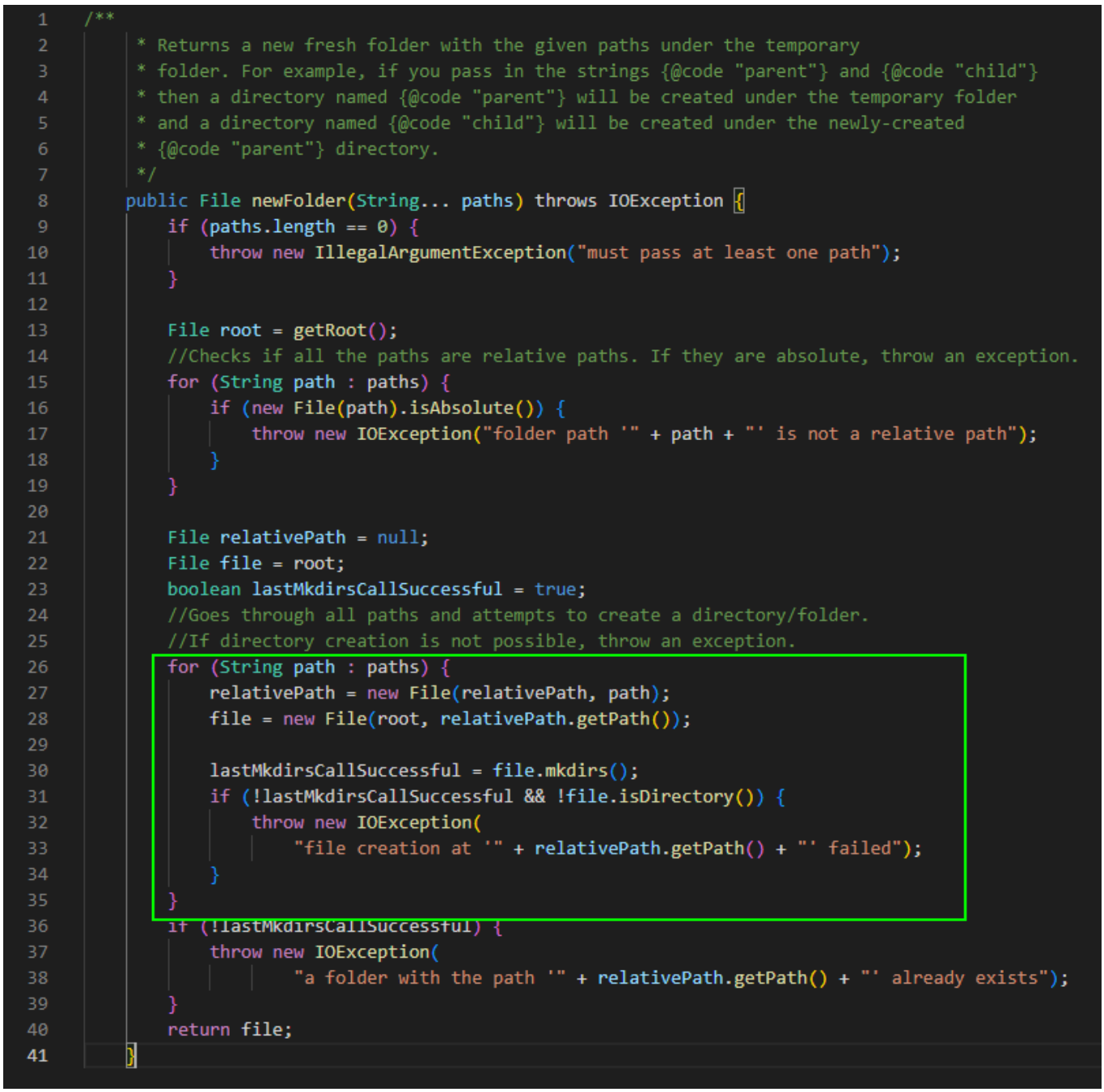
- ○ Very Difficult
- ○ Difficult
- ○ Neutral
- ○ Easy
- ○ Very Easy

What part of the marked piece of code do you find hard to understand?



```java
/**
 * Returns a new fresh folder with the given paths under the temporary
 * folder. For example, if you pass in the strings {@code "parent"} and {@code "child"}
 * then a directory named {@code "parent"} will be created under the temporary folder
 * and a directory named {@code "child"} will be created under the newly-created
 * {@code "parent"} directory.
 */
public File newFolder(String... paths) throws IOException {
    if (paths.length == 0) {
        throw new IllegalArgumentException("must pass at least one path");
    }

    File root = getRoot();
    //Checks if all the paths are relative paths. If they are absolute, throw an exception.
    for (String path : paths) {
        if (new File(path).isAbsolute()) {
            throw new IOException("folder path '" + path + "' is not a relative path");
        }
    }

    File relativePath = null;
    File file = root;
    boolean lastMkdirsCallSuccessful = true;
    //Goes through all paths and attempts to create a directory/folder.
    //If directory creation is not possible, throw an exception.
    for (String path : paths) {
        relativePath = new File(relativePath, path);
        file = new File(root, relativePath.getPath());

        lastMkdirsCallSuccessful = file.mkdirs();
        if (!lastMkdirsCallSuccessful && !file.isDirectory()) {
            throw new IOException(
                "file creation at '" + relativePath.getPath() + "' failed");
        }
    }
    if (!lastMkdirsCallSuccessful) {
        throw new IOException(
                "a folder with the path '" + relativePath.getPath() + "' already exists");
    }
    return file;
}
```

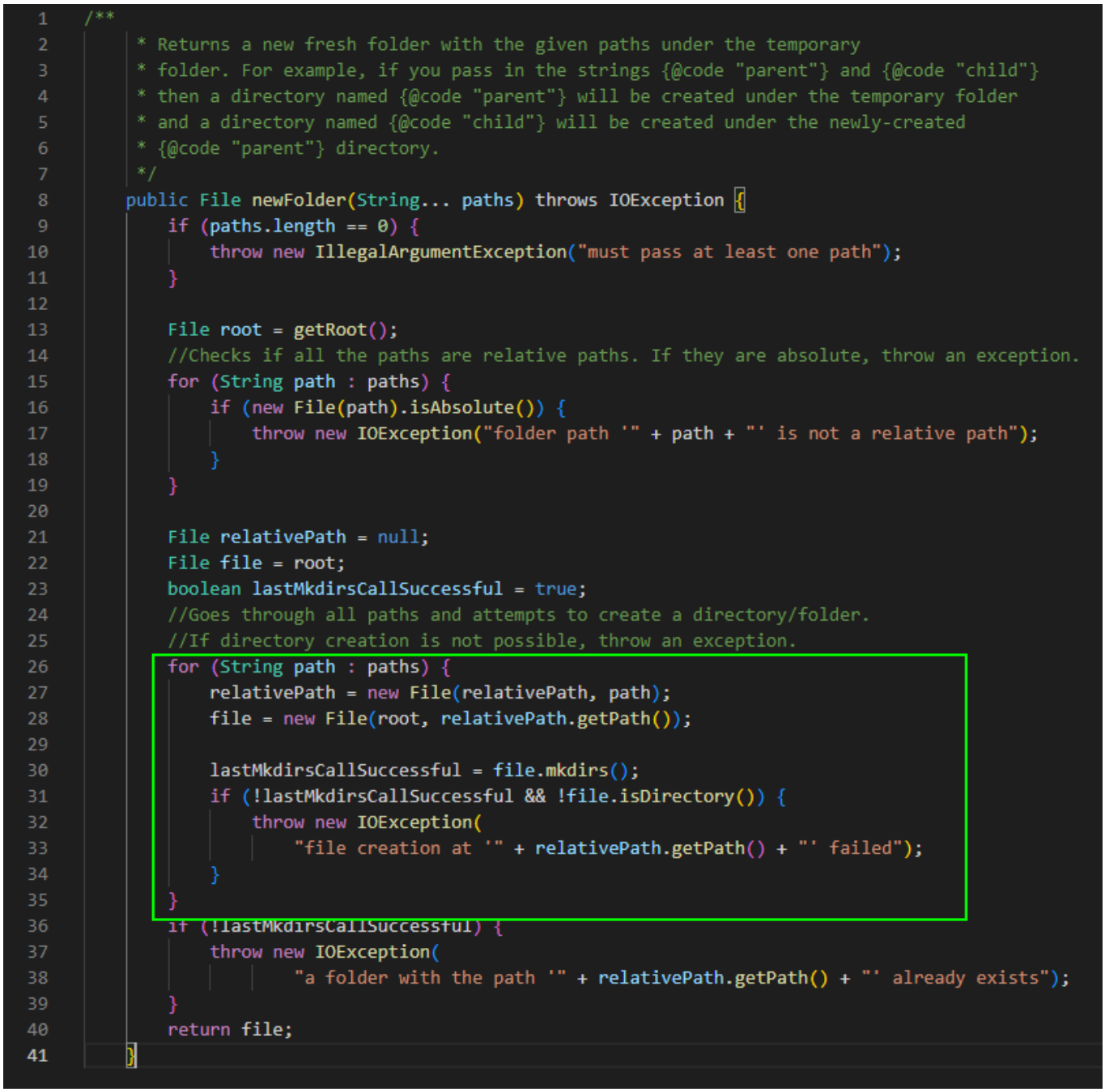
How would you rate your comprehension of the marked piece of code?



```java
/**
 * Returns a new fresh folder with the given paths under the temporary
 * folder. For example, if you pass in the strings {@code "parent"} and {@code "child"}
 * then a directory named {@code "parent"} will be created under the temporary folder
 * and a directory named {@code "child"} will be created under the newly-created
 * {@code "parent"} directory.
 */
public File newFolder(String... paths) throws IOException {
    if (paths.length == 0) {
        throw new IllegalArgumentException("must pass at least one path");
    }

    File root = getRoot();
    //Checks if all the paths are relative paths. If they are absolute, throw an exception.
    for (String path : paths) {
        if (new File(path).isAbsolute()) {
            throw new IOException("folder path '" + path + "' is not a relative path");
        }
    }

    File relativePath = null;
    File file = root;
    boolean lastMkdirsCallSuccessful = true;
    //Goes through all paths and attempts to create a directory/folder.
    //If directory creation is not possible, throw an exception.
    for (String path : paths) {
        relativePath = new File(relativePath, path);
        file = new File(root, relativePath.getPath());

        lastMkdirsCallSuccessful = file.mkdirs();
        if (!lastMkdirsCallSuccessful && !file.isDirectory()) {
            throw new IOException(
                "file creation at '" + relativePath.getPath() + "' failed");
        }
    }
    if (!lastMkdirsCallSuccessful) {
        throw new IOException(
            "a folder with the path '" + relativePath.getPath() + "' already exists");
    }
    return file;
}
```

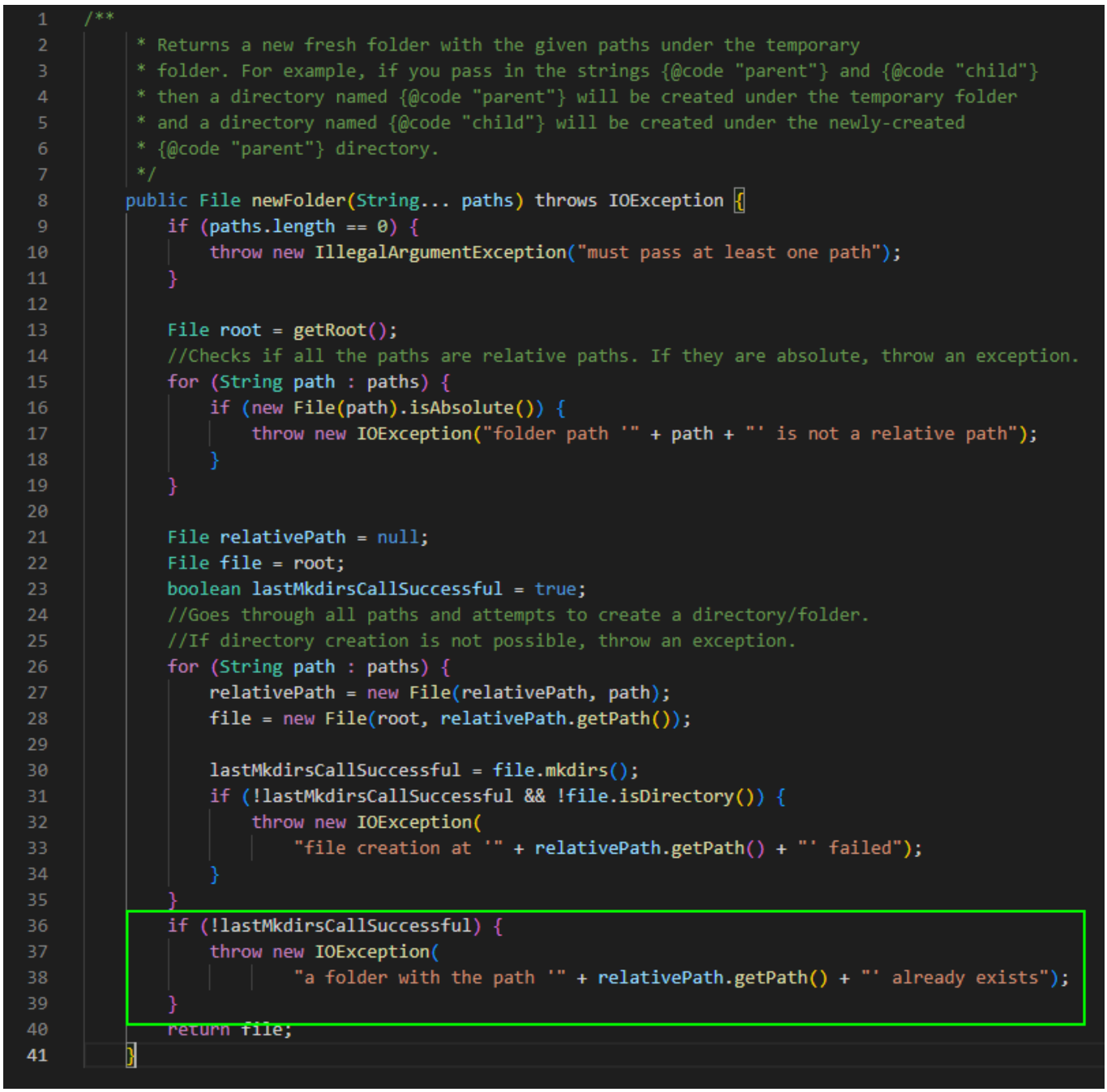
- ○ Very Difficult
- ○ Difficult
- ○ Neutral
- ○ Easy
- ○ Very Easy

What part of the marked piece of code do you find hard to understand?



```java
/**
 * Returns a new fresh folder with the given paths under the temporary
 * folder. For example, if you pass in the strings {@code "parent"} and {@code "child"}
 * then a directory named {@code "parent"} will be created under the temporary folder
 * and a directory named {@code "child"} will be created under the newly-created
 * {@code "parent"} directory.
 */
public File newFolder(String... paths) throws IOException {
    if (paths.length == 0) {
        throw new IllegalArgumentException("must pass at least one path");
    }

    File root = getRoot();
    //Checks if all the paths are relative paths. If they are absolute, throw an exception.
    for (String path : paths) {
        if (new File(path).isAbsolute()) {
            throw new IOException("folder path '" + path + "' is not a relative path");
        }
    }

    File relativePath = null;
    File file = root;
    boolean lastMkdirsCallSuccessful = true;
    //Goes through all paths and attempts to create a directory/folder.
    //If directory creation is not possible, throw an exception.
    for (String path : paths) {
        relativePath = new File(relativePath, path);
        file = new File(root, relativePath.getPath());

        lastMkdirsCallSuccessful = file.mkdirs();
        if (!lastMkdirsCallSuccessful && !file.isDirectory()) {
            throw new IOException(
                "file creation at '" + relativePath.getPath() + "' failed");
        }
    }
    if (!lastMkdirsCallSuccessful) {
        throw new IOException(
                "a folder with the path '" + relativePath.getPath() + "' already exists");
    }
    return file;
}
```

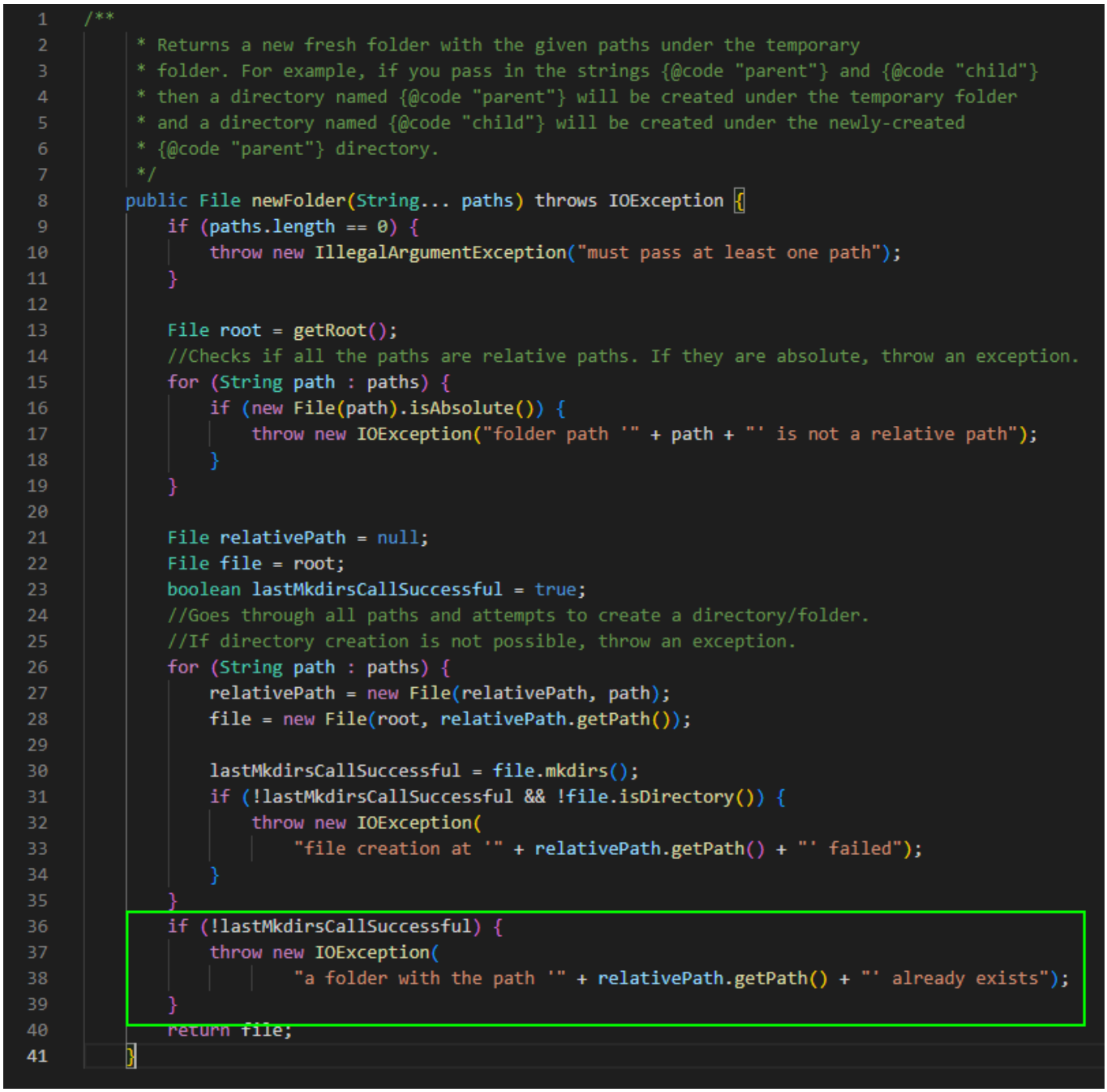
How would you rate your comprehension of the marked piece of code?



```java
/**
 * Returns a new fresh folder with the given paths under the temporary
 * folder. For example, if you pass in the strings {@code "parent"} and {@code "child"}
 * then a directory named {@code "parent"} will be created under the temporary folder
 * and a directory named {@code "child"} will be created under the newly-created
 * {@code "parent"} directory.
 */
public File newFolder(String... paths) throws IOException {
    if (paths.length == 0) {
        throw new IllegalArgumentException("must pass at least one path");
    }

    File root = getRoot();
    //Checks if all the paths are relative paths. If they are absolute, throw an exception.
    for (String path : paths) {
        if (new File(path).isAbsolute()) {
            throw new IOException("folder path '" + path + "' is not a relative path");
        }
    }

    File relativePath = null;
    File file = root;
    boolean lastMkdirsCallSuccessful = true;
    //Goes through all paths and attempts to create a directory/folder.
    //If directory creation is not possible, throw an exception.
    for (String path : paths) {
        relativePath = new File(relativePath, path);
        file = new File(root, relativePath.getPath());

        lastMkdirsCallSuccessful = file.mkdirs();
        if (!lastMkdirsCallSuccessful && !file.isDirectory()) {
            throw new IOException(
                "file creation at '" + relativePath.getPath() + "' failed");
        }
    }
    if (!lastMkdirsCallSuccessful) {
        throw new IOException(
                "a folder with the path '" + relativePath.getPath() + "' already exists");
    }
    return file;
}
```

- ○ Very Difficult
- ○ Difficult
- ○ Neutral
- ○ Easy
- ○ Very Easy

What part of the marked piece of code do you find hard to understand?



```java
/**
 * Returns a new fresh folder with the given paths under the temporary
 * folder. For example, if you pass in the strings {@code "parent"} and {@code "child"}
 * then a directory named {@code "parent"} will be created under the temporary folder
 * and a directory named {@code "child"} will be created under the newly-created
 * {@code "parent"} directory.
 */
public File newFolder(String... paths) throws IOException {
    if (paths.length == 0) {
        throw new IllegalArgumentException("must pass at least one path");
    }

    File root = getRoot();
    //Checks if all the paths are relative paths. If they are absolute, throw an exception.
    for (String path : paths) {
        if (new File(path).isAbsolute()) {
            throw new IOException("folder path '" + path + "' is not a relative path");
        }
    }

    File relativePath = null;
    File file = root;
    boolean lastMkdirsCallSuccessful = true;
    //Goes through all paths and attempts to create a directory/folder.
    //If directory creation is not possible, throw an exception.
    for (String path : paths) {
        relativePath = new File(relativePath, path);
        file = new File(root, relativePath.getPath());

        lastMkdirsCallSuccessful = file.mkdirs();
        if (!lastMkdirsCallSuccessful && !file.isDirectory()) {
            throw new IOException(
                "file creation at '" + relativePath.getPath() + "' failed");
        }
    }
    if (!lastMkdirsCallSuccessful) {
        throw new IOException(
                "a folder with the path '" + relativePath.getPath() + "' already exists");
    }
    return file;
}
```

In this piece of code we have removed part of the code at three places, we want you to explain what the missing code does, naming each part "A:", "B:" and "C:". You do not have to write code if you do not prefer, if you do than syntax is not important, it's more important that you show that you understand the code and its purpose.

```
1   public File newFolder(String... paths) throws IOException {
2       if (paths.length == 0) {
3           throw new IllegalArgumentException("must pass at least one path");
4       }
5
6       File root = getRoot();
7       for (A:            ) {
8           if (new File(path).isAbsolute()) {
9               throw new IOException("folder path '" + path + "' is not a relative path");
10          }
11      }
12
13      File relativePath = null;
14      File file = root;
15      boolean lastMkdirsCallSuccessful = true;
16      for (String path : paths) {
17          B:                  = new File(relativePath, path);
18          file = new File(root, relativePath.getPath());
19
20          lastMkdirsCallSuccessful = file.mkdirs();
21          if (C:                                        ) {
22              throw new IOException(
23                  "file creation at '" + relativePath.getPath() + "' failed");
24          }
25      }
26      if (!lastMkdirsCallSuccessful) {
27          throw new IOException(
28                  "a folder with the path '" + relativePath.getPath() + "' already exists");
29      }
30      return file;
31  }
```

A:

B:

C:

If the input to this piece of code would be ["Apple", "Orange", "Lemon", "Banana"], please explain what the output would be, and why.

```java
/**
     * Returns a new fresh folder with the given paths under the temporary
     * folder. For example, if you pass in the strings {@code "parent"} and {@code "child"}
     * then a directory named {@code "parent"} will be created under the temporary folder
     * and a directory named {@code "child"} will be created under the newly-created
     * {@code "parent"} directory.
     */
    public File newFolder(String... paths) throws IOException {
        if (paths.length == 0) {
            throw new IllegalArgumentException("must pass at least one path");
        }

        File root = getRoot();
        //Checks if all the paths are relative paths. If they are absolute, throw an exception.
        for (String path : paths) {
            if (new File(path).isAbsolute()) {
                throw new IOException("folder path '" + path + "' is not a relative path");
            }
        }

        File relativePath = null;
        File file = root;
        boolean lastMkdirsCallSuccessful = true;
        //Goes through all paths and attempts to create a directory/folder.
        //If directory creation is not possible, throw an exception.
        for (String path : paths) {
            relativePath = new File(relativePath, path);
            file = new File(root, relativePath.getPath());

            lastMkdirsCallSuccessful = file.mkdirs();
            if (!lastMkdirsCallSuccessful && !file.isDirectory()) {
                throw new IOException(
                    "file creation at '" + relativePath.getPath() + "' failed");
            }
        }
        if (!lastMkdirsCallSuccessful) {
            throw new IOException(
                    "a folder with the path '" + relativePath.getPath() + "' already exists");
        }
        return file;
    }
```

BC

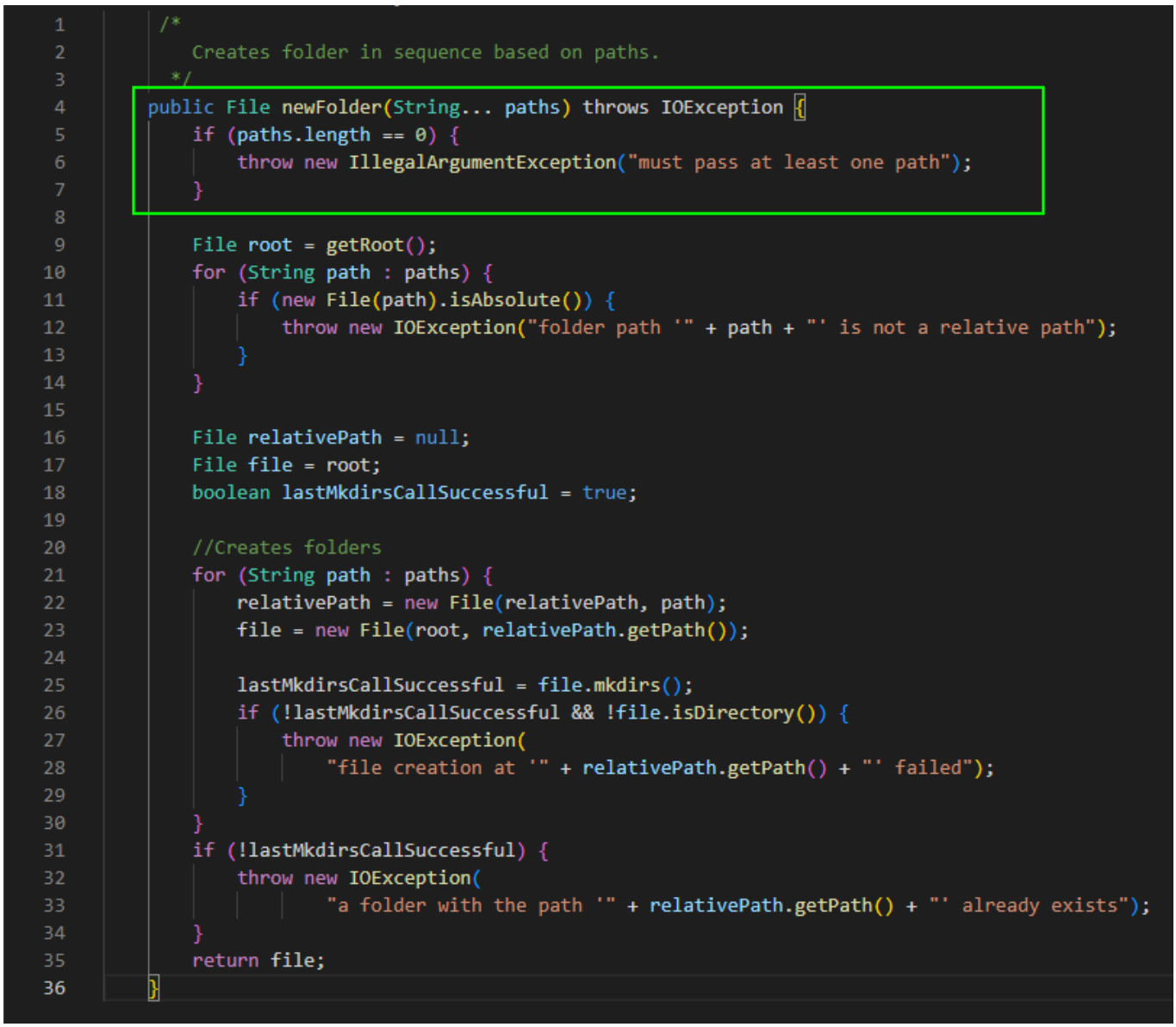
Read this code snippet carefully, there is not any time limit and when you think you understand the code fully, then please press continue.

```java
/*
   Creates folder in sequence based on paths.
 */
public File newFolder(String... paths) throws IOException {
    if (paths.length == 0) {
        throw new IllegalArgumentException("must pass at least one path");
    }

    File root = getRoot();
    for (String path : paths) {
        if (new File(path).isAbsolute()) {
            throw new IOException("folder path '" + path + "' is not a relative path");
        }
    }

    File relativePath = null;
    File file = root;
    boolean lastMkdirsCallSuccessful = true;

    //Creates folders
    for (String path : paths) {
        relativePath = new File(relativePath, path);
        file = new File(root, relativePath.getPath());

        lastMkdirsCallSuccessful = file.mkdirs();
        if (!lastMkdirsCallSuccessful && !file.isDirectory()) {
            throw new IOException(
                "file creation at '" + relativePath.getPath() + "' failed");
        }
    }
    if (!lastMkdirsCallSuccessful) {
        throw new IOException(
                "a folder with the path '" + relativePath.getPath() + "' already exists");
    }
    return file;
}
```

Based on your programming experience, how would you rate your comprehension of the previous piece of code?

- ○ Very Difficult
- ○ Difficult
- ○ Neutral
- ○ Easy
- ○ Very Easy

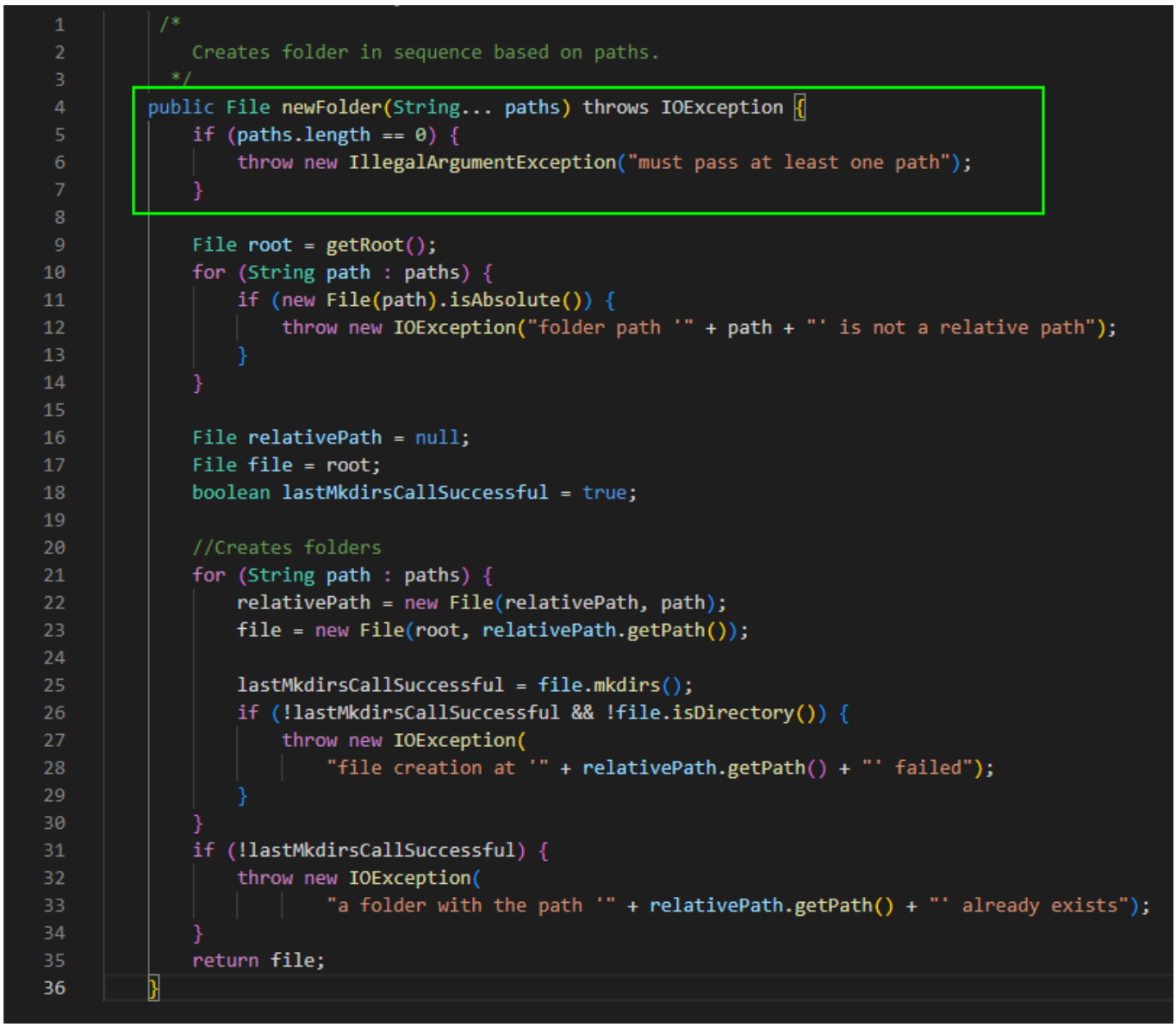
How would you rate your comprehension of the marked piece of code?



```java
/*
   Creates folder in sequence based on paths.
 */
public File newFolder(String... paths) throws IOException {
    if (paths.length == 0) {
        throw new IllegalArgumentException("must pass at least one path");
    }

    File root = getRoot();
    for (String path : paths) {
        if (new File(path).isAbsolute()) {
            throw new IOException("folder path '" + path + "' is not a relative path");
        }
    }

    File relativePath = null;
    File file = root;
    boolean lastMkdirsCallSuccessful = true;

    //Creates folders
    for (String path : paths) {
        relativePath = new File(relativePath, path);
        file = new File(root, relativePath.getPath());

        lastMkdirsCallSuccessful = file.mkdirs();
        if (!lastMkdirsCallSuccessful && !file.isDirectory()) {
            throw new IOException(
                "file creation at '" + relativePath.getPath() + "' failed");
        }
    }
    if (!lastMkdirsCallSuccessful) {
        throw new IOException(
                "a folder with the path '" + relativePath.getPath() + "' already exists");
    }
    return file;
}
```

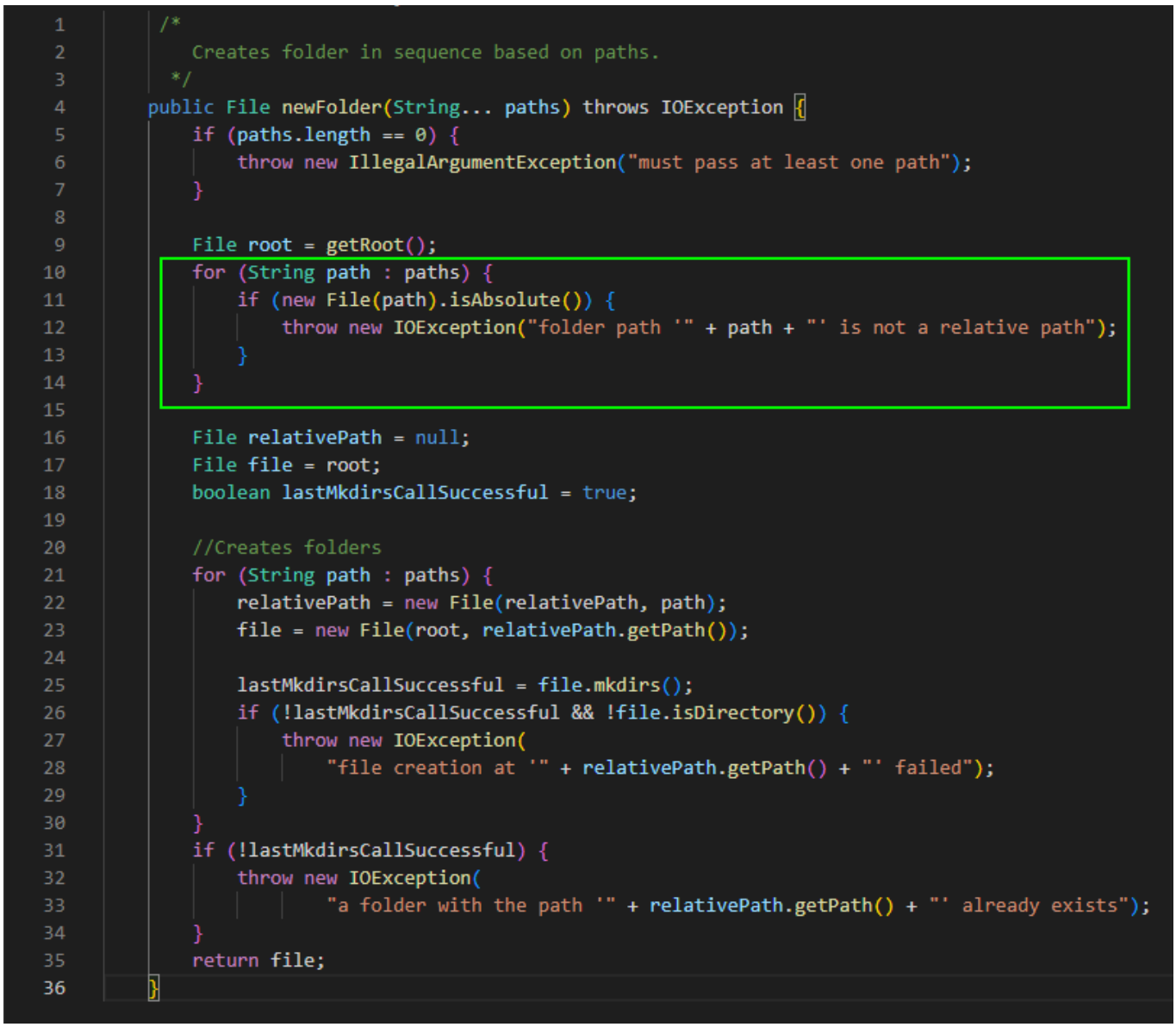
- ○ Very Difficult
- ○ Difficult
- ○ Neutral
- ○ Easy
- ○ Very Easy

What part of the marked piece of code do you find hard to understand?



```java
/*
   Creates folder in sequence based on paths.
 */
public File newFolder(String... paths) throws IOException {
    if (paths.length == 0) {
        throw new IllegalArgumentException("must pass at least one path");
    }

    File root = getRoot();
    for (String path : paths) {
        if (new File(path).isAbsolute()) {
            throw new IOException("folder path '" + path + "' is not a relative path");
        }
    }

    File relativePath = null;
    File file = root;
    boolean lastMkdirsCallSuccessful = true;

    //Creates folders
    for (String path : paths) {
        relativePath = new File(relativePath, path);
        file = new File(root, relativePath.getPath());

        lastMkdirsCallSuccessful = file.mkdirs();
        if (!lastMkdirsCallSuccessful && !file.isDirectory()) {
            throw new IOException(
                "file creation at '" + relativePath.getPath() + "' failed");
        }
    }
    if (!lastMkdirsCallSuccessful) {
        throw new IOException(
                "a folder with the path '" + relativePath.getPath() + "' already exists");
    }
    return file;
}
```

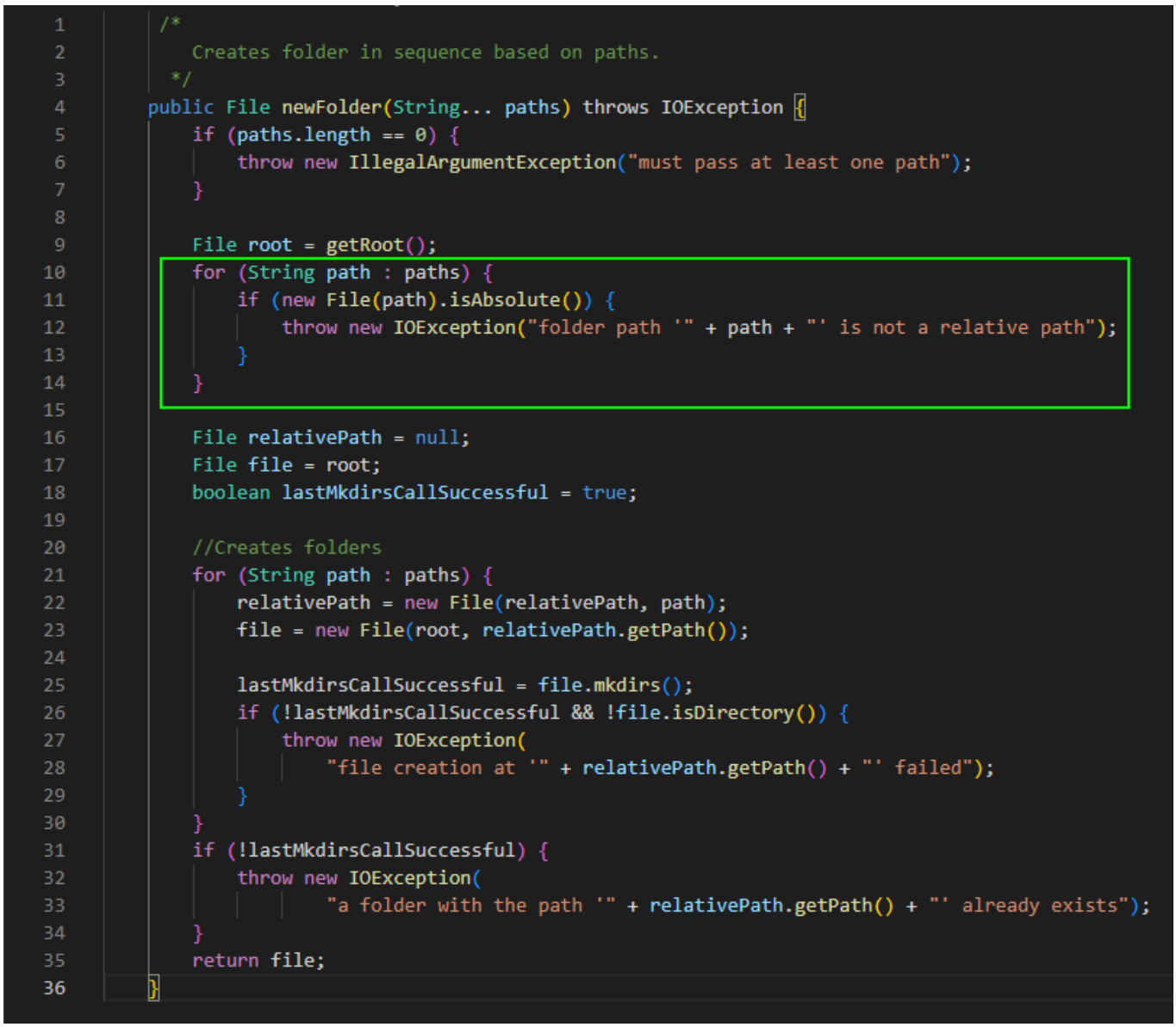
How would you rate your comprehension of the marked piece of code?



```java
/*
   Creates folder in sequence based on paths.
 */
public File newFolder(String... paths) throws IOException {
    if (paths.length == 0) {
        throw new IllegalArgumentException("must pass at least one path");
    }

    File root = getRoot();
    for (String path : paths) {
        if (new File(path).isAbsolute()) {
            throw new IOException("folder path '" + path + "' is not a relative path");
        }
    }

    File relativePath = null;
    File file = root;
    boolean lastMkdirsCallSuccessful = true;

    //Creates folders
    for (String path : paths) {
        relativePath = new File(relativePath, path);
        file = new File(root, relativePath.getPath());

        lastMkdirsCallSuccessful = file.mkdirs();
        if (!lastMkdirsCallSuccessful && !file.isDirectory()) {
            throw new IOException(
                "file creation at '" + relativePath.getPath() + "' failed");
        }
    }
    if (!lastMkdirsCallSuccessful) {
        throw new IOException(
                "a folder with the path '" + relativePath.getPath() + "' already exists");
    }
    return file;
}
```

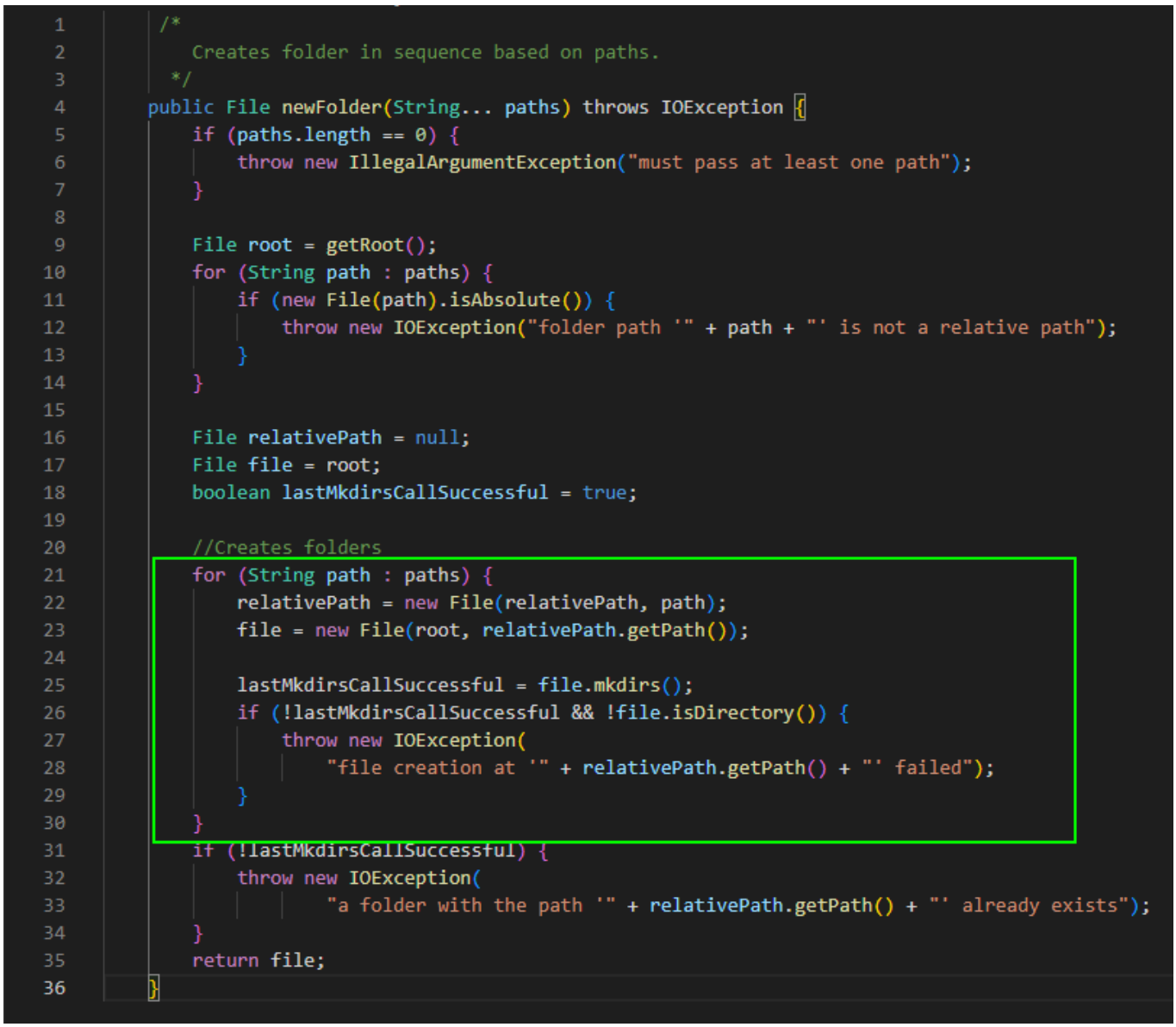
- ○ Very Difficult
- ○ Difficult
- ○ Neutral
- ○ Easy
- ○ Very Easy

What part of the marked piece of code do you find hard to understand?



```java
/*
   Creates folder in sequence based on paths.
 */
public File newFolder(String... paths) throws IOException {
    if (paths.length == 0) {
        throw new IllegalArgumentException("must pass at least one path");
    }

    File root = getRoot();
    for (String path : paths) {
        if (new File(path).isAbsolute()) {
            throw new IOException("folder path '" + path + "' is not a relative path");
        }
    }

    File relativePath = null;
    File file = root;
    boolean lastMkdirsCallSuccessful = true;

    //Creates folders
    for (String path : paths) {
        relativePath = new File(relativePath, path);
        file = new File(root, relativePath.getPath());

        lastMkdirsCallSuccessful = file.mkdirs();
        if (!lastMkdirsCallSuccessful && !file.isDirectory()) {
            throw new IOException(
                "file creation at '" + relativePath.getPath() + "' failed");
        }
    }
    if (!lastMkdirsCallSuccessful) {
        throw new IOException(
                "a folder with the path '" + relativePath.getPath() + "' already exists");
    }
    return file;
}
```

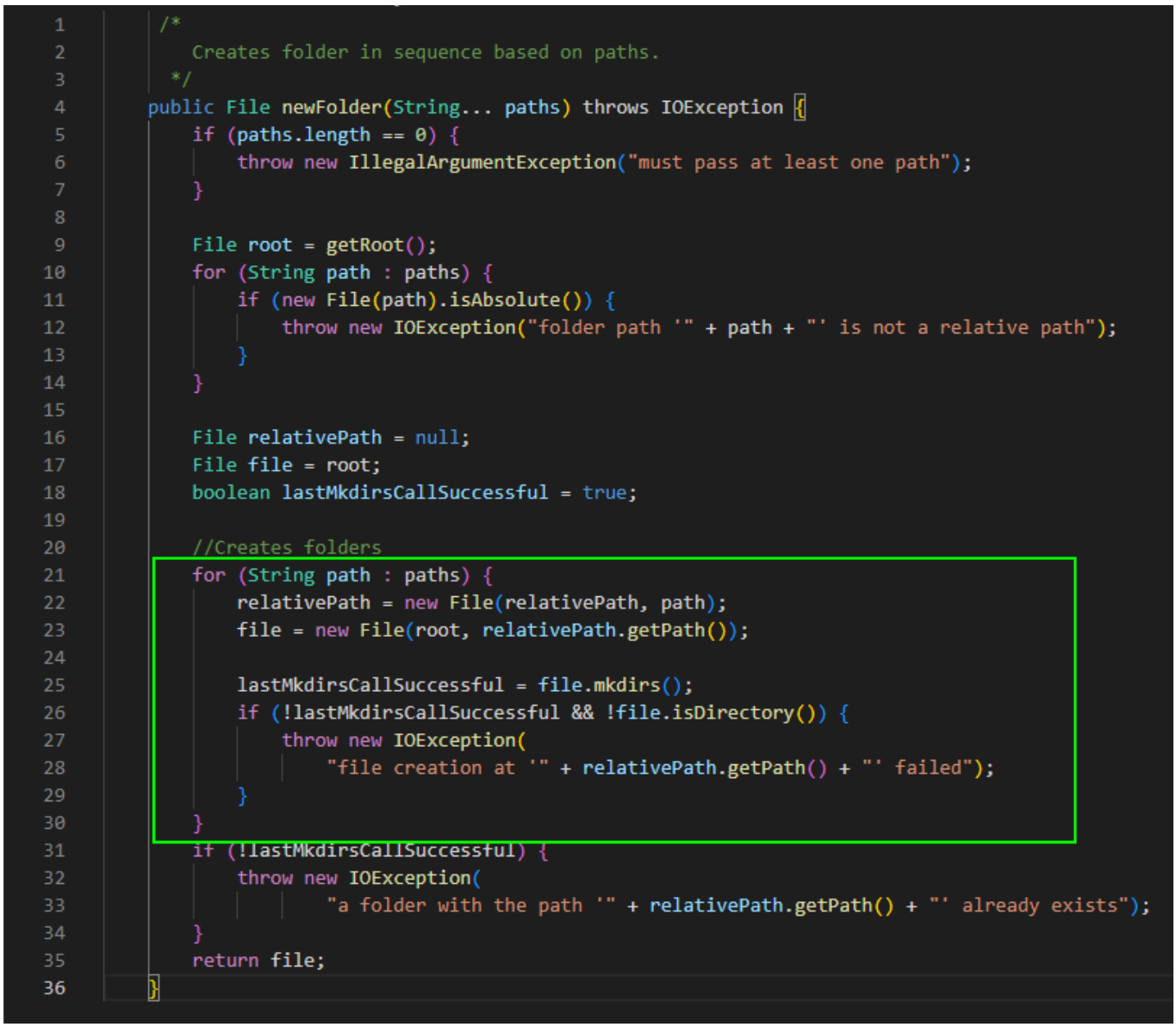
How would you rate your comprehension of the marked piece of code?



```java
/*
   Creates folder in sequence based on paths.
 */
public File newFolder(String... paths) throws IOException {
    if (paths.length == 0) {
        throw new IllegalArgumentException("must pass at least one path");
    }

    File root = getRoot();
    for (String path : paths) {
        if (new File(path).isAbsolute()) {
            throw new IOException("folder path '" + path + "' is not a relative path");
        }
    }

    File relativePath = null;
    File file = root;
    boolean lastMkdirsCallSuccessful = true;

    //Creates folders
    for (String path : paths) {
        relativePath = new File(relativePath, path);
        file = new File(root, relativePath.getPath());

        lastMkdirsCallSuccessful = file.mkdirs();
        if (!lastMkdirsCallSuccessful && !file.isDirectory()) {
            throw new IOException(
                "file creation at '" + relativePath.getPath() + "' failed");
        }
    }
    if (!lastMkdirsCallSuccessful) {
        throw new IOException(
                "a folder with the path '" + relativePath.getPath() + "' already exists");
    }
    return file;
}
```

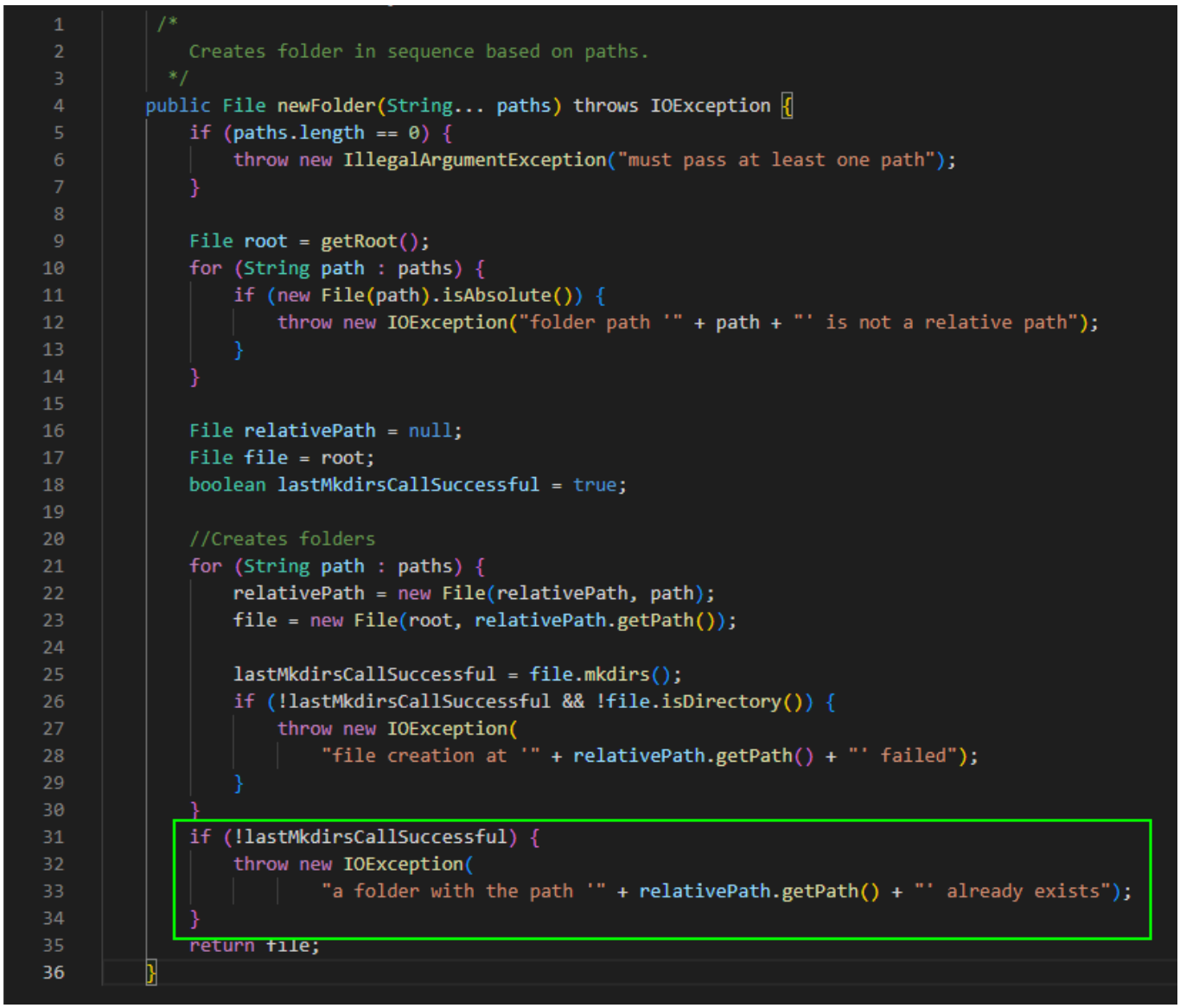
- ○ Very Difficult
- ○ Difficult
- ○ Neutral
- ○ Easy
- ○ Very Easy

What part of the marked piece of code do you find hard to understand?



```java
/*
   Creates folder in sequence based on paths.
 */
public File newFolder(String... paths) throws IOException {
    if (paths.length == 0) {
        throw new IllegalArgumentException("must pass at least one path");
    }

    File root = getRoot();
    for (String path : paths) {
        if (new File(path).isAbsolute()) {
            throw new IOException("folder path '" + path + "' is not a relative path");
        }
    }

    File relativePath = null;
    File file = root;
    boolean lastMkdirsCallSuccessful = true;

    //Creates folders
    for (String path : paths) {
        relativePath = new File(relativePath, path);
        file = new File(root, relativePath.getPath());

        lastMkdirsCallSuccessful = file.mkdirs();
        if (!lastMkdirsCallSuccessful && !file.isDirectory()) {
            throw new IOException(
                "file creation at '" + relativePath.getPath() + "' failed");
        }
    }
    if (!lastMkdirsCallSuccessful) {
        throw new IOException(
                "a folder with the path '" + relativePath.getPath() + "' already exists");
    }
    return file;
}
```

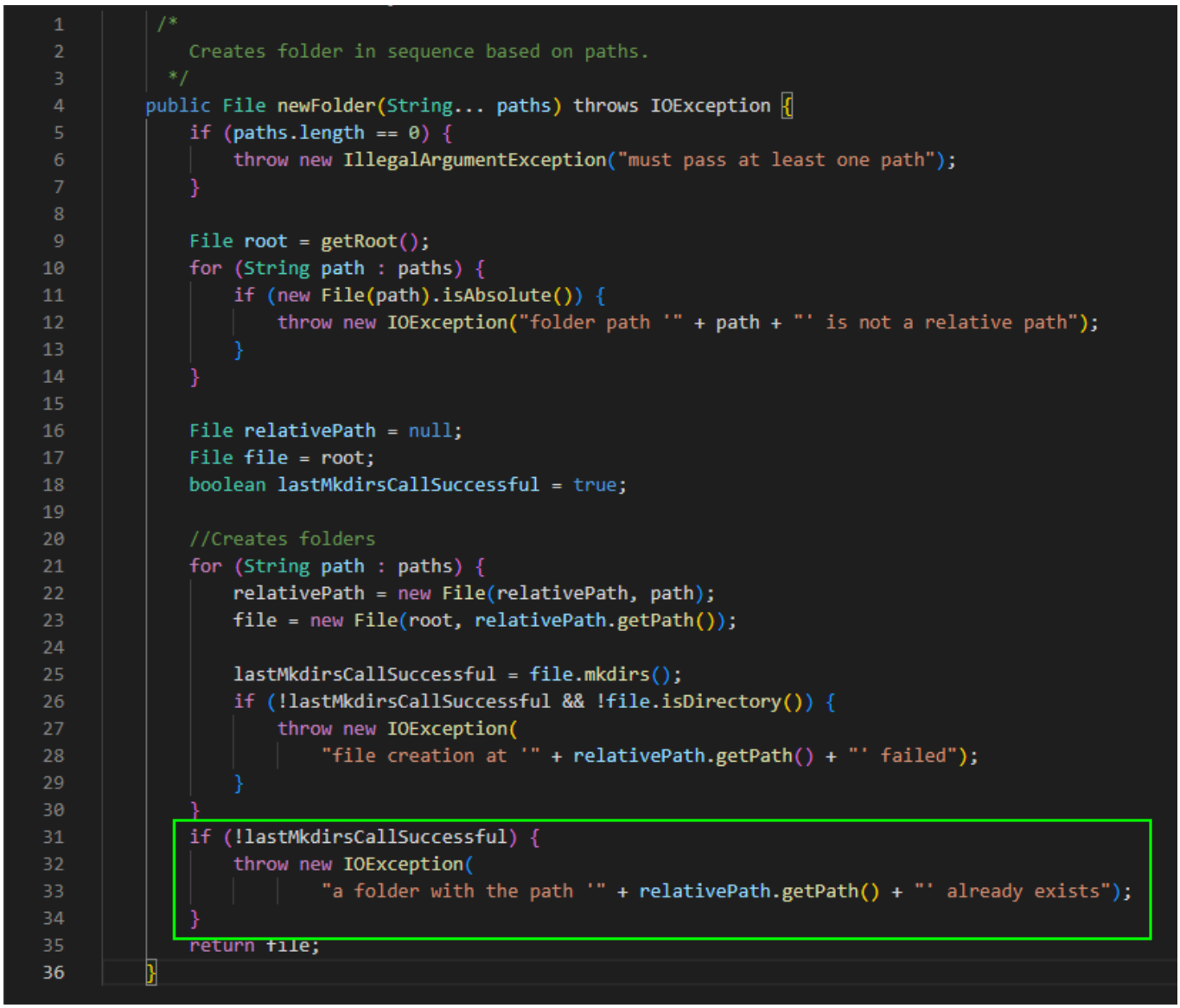
How would you rate your comprehension of the marked piece of code?



```java
/*
   Creates folder in sequence based on paths.
 */
public File newFolder(String... paths) throws IOException {
    if (paths.length == 0) {
        throw new IllegalArgumentException("must pass at least one path");
    }

    File root = getRoot();
    for (String path : paths) {
        if (new File(path).isAbsolute()) {
            throw new IOException("folder path '" + path + "' is not a relative path");
        }
    }

    File relativePath = null;
    File file = root;
    boolean lastMkdirsCallSuccessful = true;

    //Creates folders
    for (String path : paths) {
        relativePath = new File(relativePath, path);
        file = new File(root, relativePath.getPath());

        lastMkdirsCallSuccessful = file.mkdirs();
        if (!lastMkdirsCallSuccessful && !file.isDirectory()) {
            throw new IOException(
                "file creation at '" + relativePath.getPath() + "' failed");
        }
    }
    if (!lastMkdirsCallSuccessful) {
        throw new IOException(
            "a folder with the path '" + relativePath.getPath() + "' already exists");
    }
    return file;
}
```

- ○ Very Difficult
- ○ Difficult
- ○ Neutral
- ○ Easy
- ○ Very Easy

What part of the marked piece of code do you find hard to understand?



```java
/*
   Creates folder in sequence based on paths.
 */
public File newFolder(String... paths) throws IOException {
    if (paths.length == 0) {
        throw new IllegalArgumentException("must pass at least one path");
    }

    File root = getRoot();
    for (String path : paths) {
        if (new File(path).isAbsolute()) {
            throw new IOException("folder path '" + path + "' is not a relative path");
        }
    }

    File relativePath = null;
    File file = root;
    boolean lastMkdirsCallSuccessful = true;

    //Creates folders
    for (String path : paths) {
        relativePath = new File(relativePath, path);
        file = new File(root, relativePath.getPath());

        lastMkdirsCallSuccessful = file.mkdirs();
        if (!lastMkdirsCallSuccessful && !file.isDirectory()) {
            throw new IOException(
                "file creation at '" + relativePath.getPath() + "' failed");
        }
    }
    if (!lastMkdirsCallSuccessful) {
        throw new IOException(
            "a folder with the path '" + relativePath.getPath() + "' already exists");
    }
    return file;
}
```

In this piece of code we have removed part of the code at three places, we want you to explain what the missing code does, naming each part "A:", "B:" and "C:". You do not have to write code if you do not prefer, if you do than syntax is not important, it's more important that you show that you understand the code and its purpose.

```
1   public File newFolder(String... paths) throws IOException {
2       if (paths.length == 0) {
3           throw new IllegalArgumentException("must pass at least one path");
4       }
5
6       File root = getRoot();
7       for (A:            ) {
8           if (new File(path).isAbsolute()) {
9               throw new IOException("folder path '" + path + "' is not a relative path");
10          }
11      }
12
13      File relativePath = null;
14      File file = root;
15      boolean lastMkdirsCallSuccessful = true;
16      for (String path : paths) {
17          B:                  = new File(relativePath, path);
18          file = new File(root, relativePath.getPath());
19
20          lastMkdirsCallSuccessful = file.mkdirs();
21          if (C:                                          ) {
22              throw new IOException(
23                  "file creation at '" + relativePath.getPath() + "' failed");
24          }
25      }
26      if (!lastMkdirsCallSuccessful) {
27          throw new IOException(
28                  "a folder with the path '" + relativePath.getPath() + "' already exists");
29      }
30      return file;
31  }
```

A:

B:

C:

If the input to this piece of code would be [Apple, Orange, Lemon, Banana], please explain what the output would be, and why.

```java
/*
   Creates folder in sequence based on paths.
 */
public File newFolder(String... paths) throws IOException {
    if (paths.length == 0) {
        throw new IllegalArgumentException("must pass at least one path");
    }

    File root = getRoot();
    for (String path : paths) {
        if (new File(path).isAbsolute()) {
            throw new IOException("folder path '" + path + "' is not a relative path");
        }
    }

    File relativePath = null;
    File file = root;
    boolean lastMkdirsCallSuccessful = true;

    //Creates folders
    for (String path : paths) {
        relativePath = new File(relativePath, path);
        file = new File(root, relativePath.getPath());

        lastMkdirsCallSuccessful = file.mkdirs();
        if (!lastMkdirsCallSuccessful && !file.isDirectory()) {
            throw new IOException(
                "file creation at '" + relativePath.getPath() + "' failed");
        }
    }
    if (!lastMkdirsCallSuccessful) {
        throw new IOException(
                "a folder with the path '" + relativePath.getPath() + "' already exists");
    }
    return file;
}
```

NC

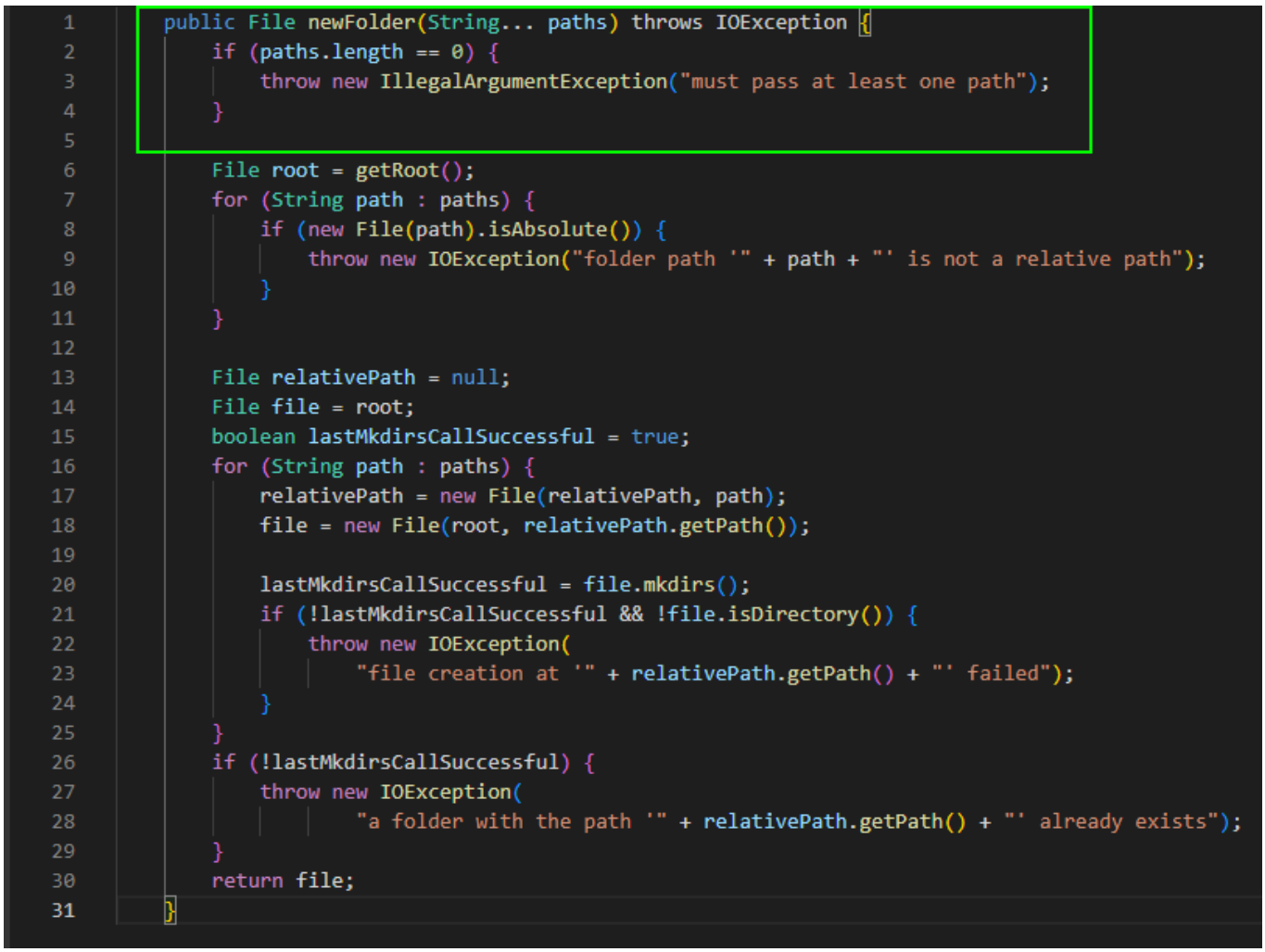
Read this code snippet carefully, there is not any time limit and when you think you understand the code fully, then please press continue.

```
public File newFolder(String... paths) throws IOException {
    if (paths.length == 0) {
        throw new IllegalArgumentException("must pass at least one path");
    }

    File root = getRoot();
    for (String path : paths) {
        if (new File(path).isAbsolute()) {
            throw new IOException("folder path '" + path + "' is not a relative path");
        }
    }

    File relativePath = null;
    File file = root;
    boolean lastMkdirsCallSuccessful = true;
    for (String path : paths) {
        relativePath = new File(relativePath, path);
        file = new File(root, relativePath.getPath());

        lastMkdirsCallSuccessful = file.mkdirs();
        if (!lastMkdirsCallSuccessful && !file.isDirectory()) {
            throw new IOException(
                "file creation at '" + relativePath.getPath() + "' failed");
        }
    }
    if (!lastMkdirsCallSuccessful) {
        throw new IOException(
            "a folder with the path '" + relativePath.getPath() + "' already exists");
    }
    return file;
}
```

Based on your programming experience, how would you rate your comprehension of the previous piece of code?

- ○ Very Difficult
- ○ Difficult
- ○ Neutral
- ○ Easy
- ○ Very Easy

How would you rate your comprehension of the marked piece of code?



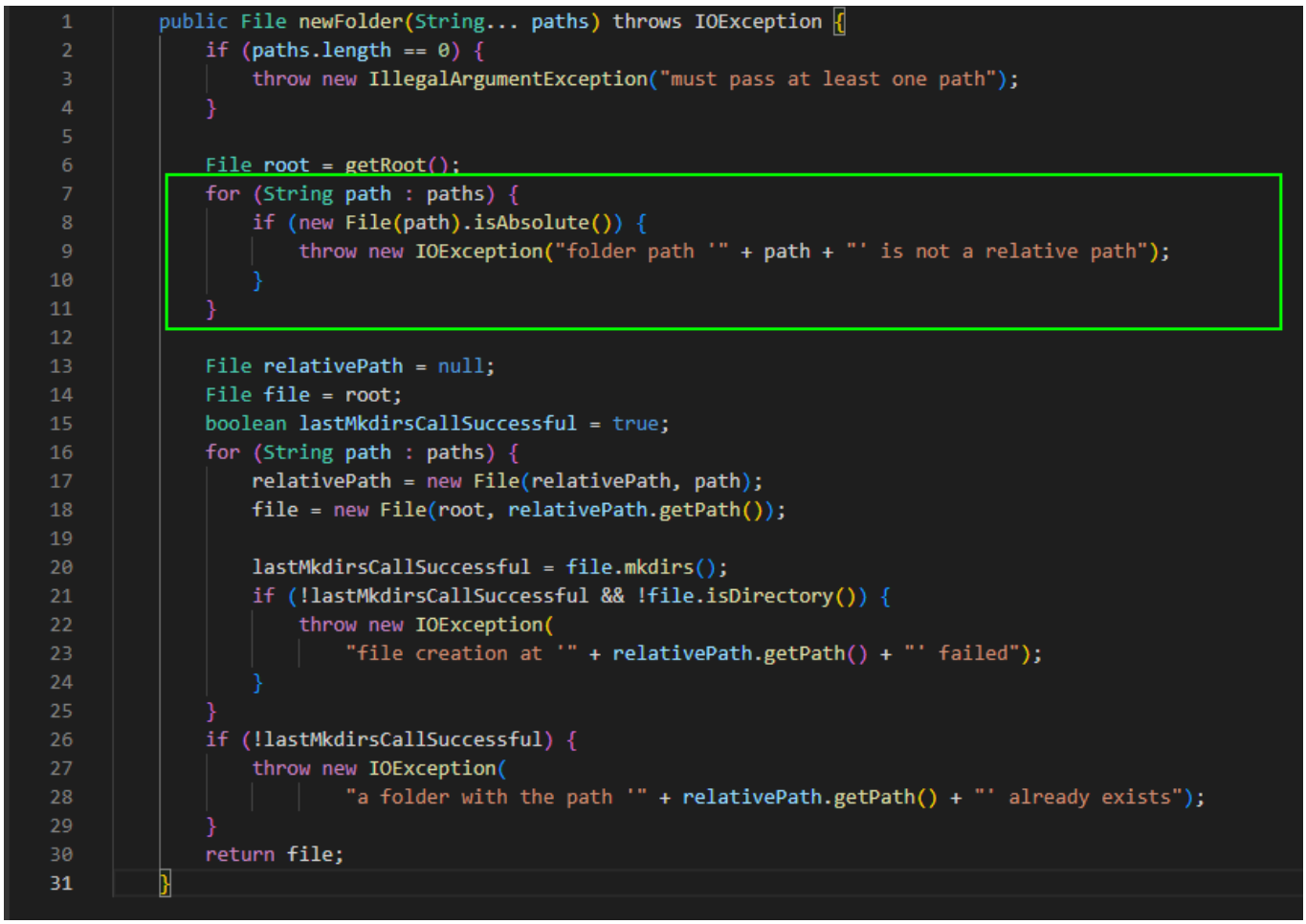
```
public File newFolder(String... paths) throws IOException {
    if (paths.length == 0) {
        throw new IllegalArgumentException("must pass at least one path");
    }

    File root = getRoot();
    for (String path : paths) {
        if (new File(path).isAbsolute()) {
            throw new IOException("folder path '" + path + "' is not a relative path");
        }
    }

    File relativePath = null;
    File file = root;
    boolean lastMkdirsCallSuccessful = true;
    for (String path : paths) {
        relativePath = new File(relativePath, path);
        file = new File(root, relativePath.getPath());

        lastMkdirsCallSuccessful = file.mkdirs();
        if (!lastMkdirsCallSuccessful && !file.isDirectory()) {
            throw new IOException(
                "file creation at '" + relativePath.getPath() + "' failed");
        }
    }
    if (!lastMkdirsCallSuccessful) {
        throw new IOException(
                "a folder with the path '" + relativePath.getPath() + "' already exists");
    }
    return file;
}
```

- ○ Very Difficult
- ○ Difficult
- ○ Neutral
- ○ Easy
- ○ Very Easy

What part of the marked piece of code do you find hard to understand?

```java
public File newFolder(String... paths) throws IOException {
    if (paths.length == 0) {
        throw new IllegalArgumentException("must pass at least one path");
    }

    File root = getRoot();
    for (String path : paths) {
        if (new File(path).isAbsolute()) {
            throw new IOException("folder path '" + path + "' is not a relative path");
        }
    }

    File relativePath = null;
    File file = root;
    boolean lastMkdirsCallSuccessful = true;
    for (String path : paths) {
        relativePath = new File(relativePath, path);
        file = new File(root, relativePath.getPath());

        lastMkdirsCallSuccessful = file.mkdirs();
        if (!lastMkdirsCallSuccessful && !file.isDirectory()) {
            throw new IOException(
                "file creation at '" + relativePath.getPath() + "' failed");
        }
    }
    if (!lastMkdirsCallSuccessful) {
        throw new IOException(
                "a folder with the path '" + relativePath.getPath() + "' already exists");
    }
    return file;
}
```

How would you rate your comprehension of the marked piece of code?



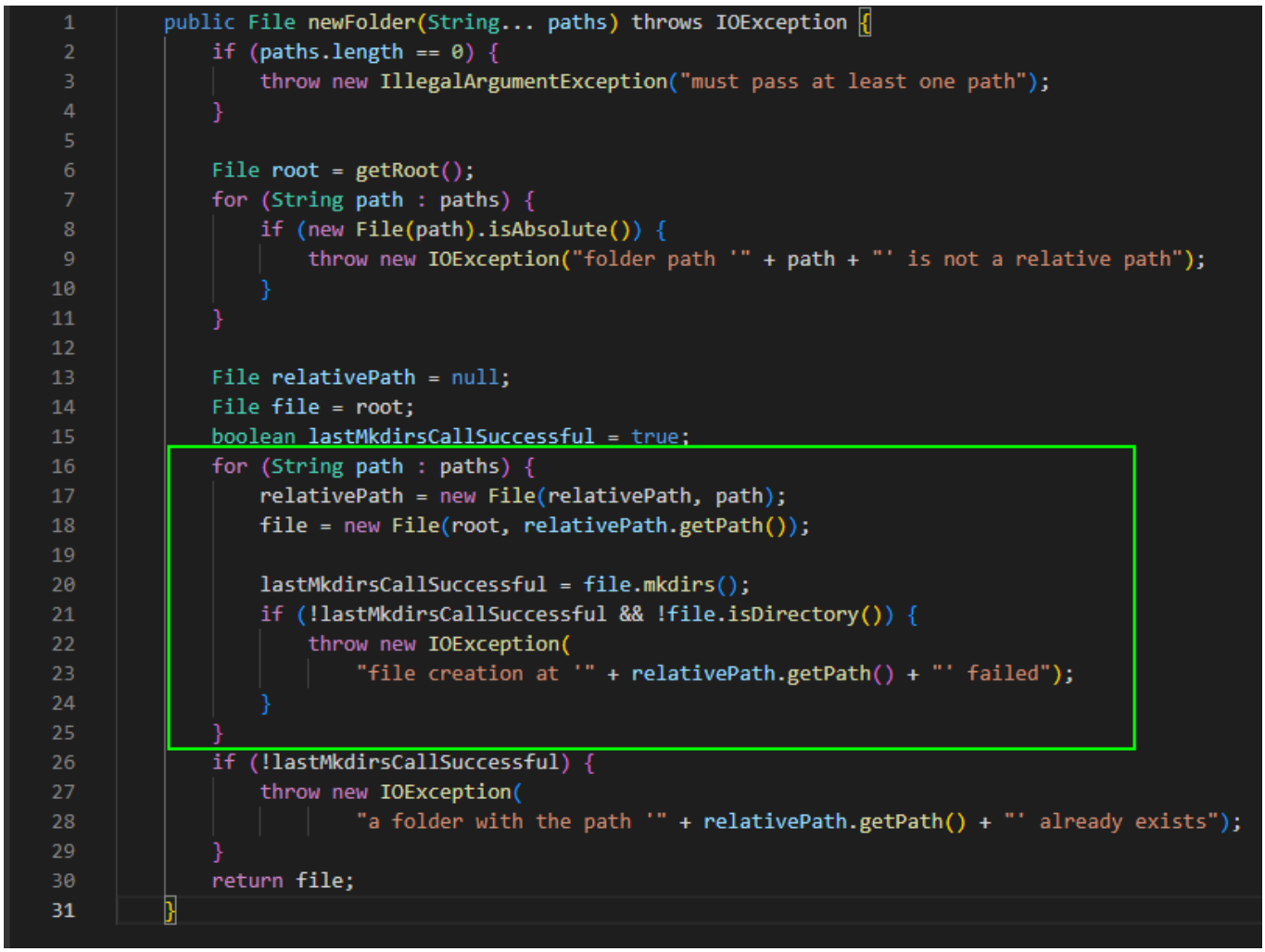
```
1   public File newFolder(String... paths) throws IOException {
2       if (paths.length == 0) {
3           throw new IllegalArgumentException("must pass at least one path");
4       }
5
6       File root = getRoot();
7       for (String path : paths) {
8           if (new File(path).isAbsolute()) {
9               throw new IOException("folder path '" + path + "' is not a relative path");
10          }
11      }
12
13      File relativePath = null;
14      File file = root;
15      boolean lastMkdirsCallSuccessful = true;
16      for (String path : paths) {
17          relativePath = new File(relativePath, path);
18          file = new File(root, relativePath.getPath());
19
20          lastMkdirsCallSuccessful = file.mkdirs();
21          if (!lastMkdirsCallSuccessful && !file.isDirectory()) {
22              throw new IOException(
23                  "file creation at '" + relativePath.getPath() + "' failed");
24          }
25      }
26      if (!lastMkdirsCallSuccessful) {
27          throw new IOException(
28                  "a folder with the path '" + relativePath.getPath() + "' already exists");
29      }
30      return file;
31  }
```

- ○ Very Difficult
- ○ Difficult
- ○ Neutral
- ○ Easy
- ○ Very Easy

What part of the marked piece of code do you find hard to understand?

```java
public File newFolder(String... paths) throws IOException {
    if (paths.length == 0) {
        throw new IllegalArgumentException("must pass at least one path");
    }

    File root = getRoot();
    for (String path : paths) {
        if (new File(path).isAbsolute()) {
            throw new IOException("folder path '" + path + "' is not a relative path");
        }
    }

    File relativePath = null;
    File file = root;
    boolean lastMkdirsCallSuccessful = true;
    for (String path : paths) {
        relativePath = new File(relativePath, path);
        file = new File(root, relativePath.getPath());

        lastMkdirsCallSuccessful = file.mkdirs();
        if (!lastMkdirsCallSuccessful && !file.isDirectory()) {
            throw new IOException(
                "file creation at '" + relativePath.getPath() + "' failed");
        }
    }
    if (!lastMkdirsCallSuccessful) {
        throw new IOException(
                "a folder with the path '" + relativePath.getPath() + "' already exists");
    }
    return file;
}
```

How would you rate your comprehension of the marked piece of code?



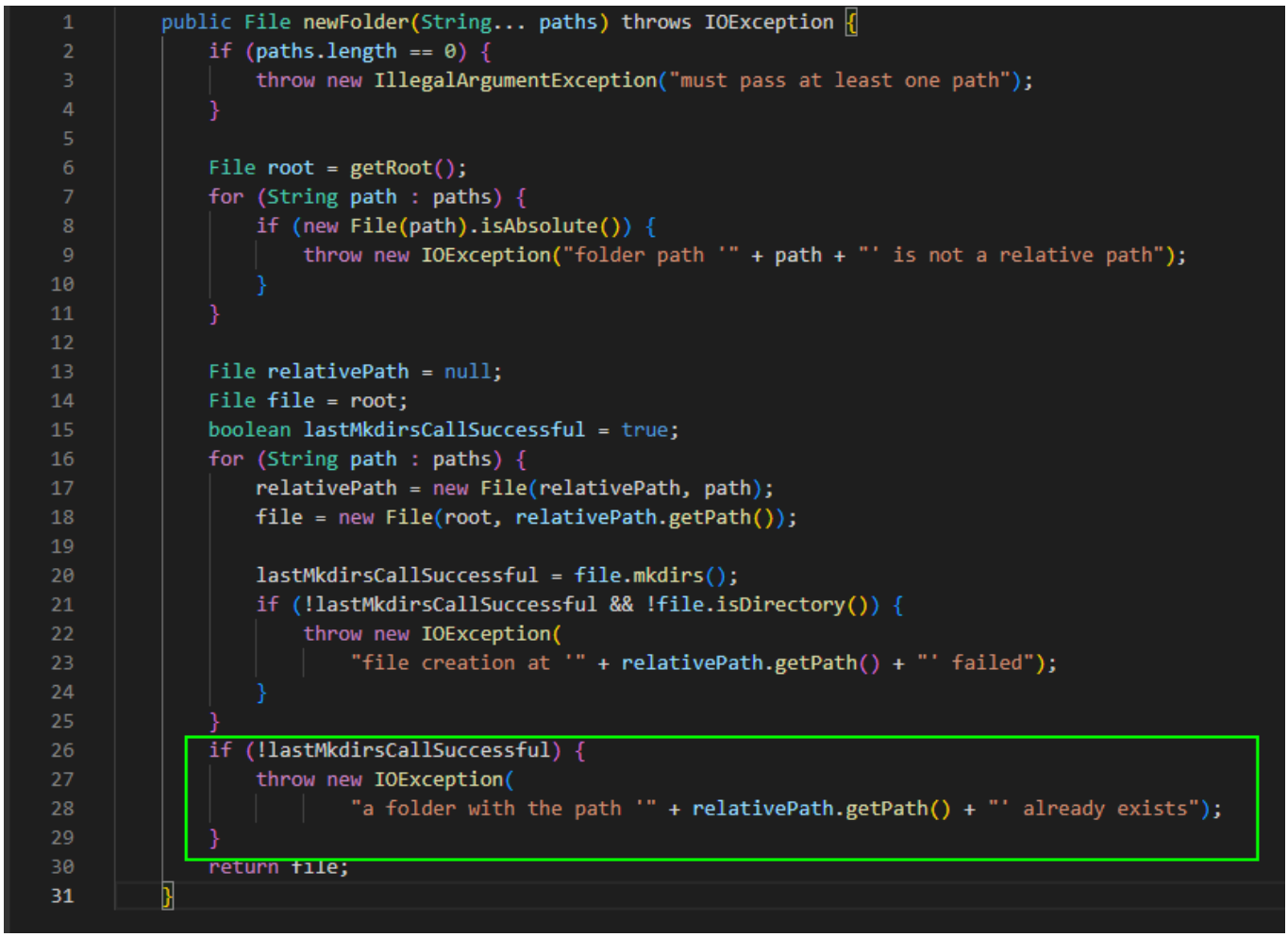
```
1   public File newFolder(String... paths) throws IOException {
2       if (paths.length == 0) {
3           throw new IllegalArgumentException("must pass at least one path");
4       }
5
6       File root = getRoot();
7       for (String path : paths) {
8           if (new File(path).isAbsolute()) {
9               throw new IOException("folder path '" + path + "' is not a relative path");
10          }
11      }
12
13      File relativePath = null;
14      File file = root;
15      boolean lastMkdirsCallSuccessful = true;
16      for (String path : paths) {
17          relativePath = new File(relativePath, path);
18          file = new File(root, relativePath.getPath());
19
20          lastMkdirsCallSuccessful = file.mkdirs();
21          if (!lastMkdirsCallSuccessful && !file.isDirectory()) {
22              throw new IOException(
23                  "file creation at '" + relativePath.getPath() + "' failed");
24          }
25      }
26      if (!lastMkdirsCallSuccessful) {
27          throw new IOException(
28                  "a folder with the path '" + relativePath.getPath() + "' already exists");
29      }
30      return file;
31  }
```

- ○ Very Difficult
- ○ Difficult
- ○ Neutral
- ○ Easy
- ○ Very Easy

What part of the marked piece of code do you find hard to understand?

```java
public File newFolder(String... paths) throws IOException {
    if (paths.length == 0) {
        throw new IllegalArgumentException("must pass at least one path");
    }

    File root = getRoot();
    for (String path : paths) {
        if (new File(path).isAbsolute()) {
            throw new IOException("folder path '" + path + "' is not a relative path");
        }
    }

    File relativePath = null;
    File file = root;
    boolean lastMkdirsCallSuccessful = true;
    for (String path : paths) {
        relativePath = new File(relativePath, path);
        file = new File(root, relativePath.getPath());

        lastMkdirsCallSuccessful = file.mkdirs();
        if (!lastMkdirsCallSuccessful && !file.isDirectory()) {
            throw new IOException(
                "file creation at '" + relativePath.getPath() + "' failed");
        }
    }
    if (!lastMkdirsCallSuccessful) {
        throw new IOException(
                "a folder with the path '" + relativePath.getPath() + "' already exists");
    }
    return file;
}
```

How would you rate your comprehension of the marked piece of code?



```
public File newFolder(String... paths) throws IOException {
    if (paths.length == 0) {
        throw new IllegalArgumentException("must pass at least one path");
    }

    File root = getRoot();
    for (String path : paths) {
        if (new File(path).isAbsolute()) {
            throw new IOException("folder path '" + path + "' is not a relative path");
        }
    }

    File relativePath = null;
    File file = root;
    boolean lastMkdirsCallSuccessful = true;
    for (String path : paths) {
        relativePath = new File(relativePath, path);
        file = new File(root, relativePath.getPath());

        lastMkdirsCallSuccessful = file.mkdirs();
        if (!lastMkdirsCallSuccessful && !file.isDirectory()) {
            throw new IOException(
                "file creation at '" + relativePath.getPath() + "' failed");
        }
    }
    if (!lastMkdirsCallSuccessful) {
        throw new IOException(
            "a folder with the path '" + relativePath.getPath() + "' already exists");
    }
    return file;
}
```

- ○ Very Difficult
- ○ Difficult
- ○ Neutral
- ○ Easy
- ○ Very Easy

What part of the marked piece of code do you find hard to understand?

```java
public File newFolder(String... paths) throws IOException {
    if (paths.length == 0) {
        throw new IllegalArgumentException("must pass at least one path");
    }

    File root = getRoot();
    for (String path : paths) {
        if (new File(path).isAbsolute()) {
            throw new IOException("folder path '" + path + "' is not a relative path");
        }
    }

    File relativePath = null;
    File file = root;
    boolean lastMkdirsCallSuccessful = true;
    for (String path : paths) {
        relativePath = new File(relativePath, path);
        file = new File(root, relativePath.getPath());

        lastMkdirsCallSuccessful = file.mkdirs();
        if (!lastMkdirsCallSuccessful && !file.isDirectory()) {
            throw new IOException(
                "file creation at '" + relativePath.getPath() + "' failed");
        }
    }
    if (!lastMkdirsCallSuccessful) {
        throw new IOException(
            "a folder with the path '" + relativePath.getPath() + "' already exists");
    }
    return file;
}
```

In this piece of code we have removed part of the code at three places, we want you to explain what the missing code does, naming each part "A:", "B:" and "C:". You do not have to write code if you do not prefer, if you do than syntax is not important, it's more important that you show that you understand the code and its purpose.

```
1   public File newFolder(String... paths) throws IOException {
2       if (paths.length == 0) {
3           throw new IllegalArgumentException("must pass at least one path");
4       }
5
6       File root = getRoot();
7       for (A:            ) {
8           if (new File(path).isAbsolute()) {
9               throw new IOException("folder path '" + path + "' is not a relative path");
10          }
11      }
12
13      File relativePath = null;
14      File file = root;
15      boolean lastMkdirsCallSuccessful = true;
16      for (String path : paths) {
17          B:                  = new File(relativePath, path);
18          file = new File(root, relativePath.getPath());
19
20          lastMkdirsCallSuccessful = file.mkdirs();
21          if (C:                                        ) {
22              throw new IOException(
23                  "file creation at '" + relativePath.getPath() + "' failed");
24          }
25      }
26      if (!lastMkdirsCallSuccessful) {
27          throw new IOException(
28                  "a folder with the path '" + relativePath.getPath() + "' already exists");
29      }
30      return file;
31  }
```

A:

B:

C:

If the input to this piece of code would be [Apple, Orange, Lemon, Banana], please explain what the output would be, and why.

```
public File newFolder(String... paths) throws IOException {
    if (paths.length == 0) {
        throw new IllegalArgumentException("must pass at least one path");
    }

    File root = getRoot();
    for (String path : paths) {
        if (new File(path).isAbsolute()) {
            throw new IOException("folder path '" + path + "' is not a relative path");
        }
    }

    File relativePath = null;
    File file = root;
    boolean lastMkdirsCallSuccessful = true;
    for (String path : paths) {
        relativePath = new File(relativePath, path);
        file = new File(root, relativePath.getPath());

        lastMkdirsCallSuccessful = file.mkdirs();
        if (!lastMkdirsCallSuccessful && !file.isDirectory()) {
            throw new IOException(
                "file creation at '" + relativePath.getPath() + "' failed");
        }
    }
    if (!lastMkdirsCallSuccessful) {
        throw new IOException(
                "a folder with the path '" + relativePath.getPath() + "' already exists");
    }
    return file;
}
```