

Java IO File (NIO.2) - 0

Given code of Test.java file:

```
import java.nio.file.Path;
import java.nio.file.Paths;

public class Test {
    public static void main(String[] args) {
        Path path = Paths.get("F:\\A\\B\\C\\Book.java");
        System.out.println(path.subpath(1,4));
    }
}
```

What will be the result of compiling and executing Test class?

- A - A\\B\\C
- B - A\\B\\C\\Book.java
- C - B\\C\\Book.java
- D - Exception is thrown at runtime

Java IO File (NIO.2) - 1

F: is accessible for reading/writing and below is the directory structure for F:

```
F:.\n├── Parent\n│   ├── a.txt\n│   └── b.txt\n└── Child\n    ├── c.txt\n    └── d.txt
```

Given code of Test.java file:

```
import java.io.IOException;\nimport java.nio.file.Files;\nimport java.nio.file.Path;\nimport java.nio.file.Paths;\nimport java.nio.file.attribute.BasicFileAttributes;\nimport java.util.function.BiPredicate;\nimport java.util.stream.Stream;\n\npublic class Test {\n    public static void main(String[] args) throws IOException {\n        Path root = Paths.get("F:");\n        BiPredicate<Path, BasicFileAttributes> predicate = (p,a) ->\n            p.toString().endsWith("txt");\n        try(Stream<Path> paths = Files.find(root, 2, predicate))\n        {\n            paths.forEach(System.out::println);\n        }\n    }\n}
```

What will be the result of compiling and executing Test class?

A - Above program executes successfully and prints nothing on to the console.

B - Above program executes successfully and prints below lines on to the console:

```
F:Parent\a.txt\nF:Parent\b.txt
```

C - Above program executes successfully and prints below lines on to the console:

```
F:Parent\Child\c.txt\nF:Parent\Child\d.txt
```

F:Parent\a.txt
F:Parent\b.txt

Java IO File (NIO.2) - 2

C: is accessible for reading/writing and below is the content of 'C:\TEMP' folder:

```
C:\TEMP
|
|  msg
|
└─ Parent
    └─ Child
        Message.txt
```

'msg' is a symbolic link file for 'C:\TEMP\Parent\Child\Message.txt'.

Message.txt contains following text: Welcome!

Given code of Test.java file:

```
import java.io.BufferedReader;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;

public class Test {
    public static void main(String[] args) throws IOException {
        Path path = Paths.get("C:", "TEMP", "msg");

        try (BufferedReader reader = Files.newBufferedReader(path))
        {
            String str = null;
            while ((str = reader.readLine()) != null) {
                System.out.println(str);
            }
        }
    }
}
```

What will be the result of compiling and executing Test class?

- A - Program executes successfully and produces no output
- B - An exception is thrown at runtime
- C - Compilation error
- D - Program executes successfully and prints 'Welcome!' on to the console

Java IO File (NIO.2) - 3

Given code of Test.java file:

```
import java.io.*;
import java.nio.file.Files;
import java.nio.file.Paths;

public class Test {
    public static void main(String[] args) throws IOException {
        /*INSERT*/
    }
}
```

F: is accessible for reading and contains 'Book.java' file.

Which of the following statements, if used to replace `/*INSERT*/`, will successfully print contents of 'Book.java' on to the console? Select 3 options.

A - Files.

`lines(Paths.get("F:\Book.java")).stream().forEach(System.out::println);`

B -

`Files.readAllLines(Paths.get("F:\Book.java")).forEach(System.out::println);`

C -

`Files.readAllLines(Paths.get("F:\Book.java")).stream().forEach(System.out::println);`

D - `Files.lines(Paths.get("F:\Book.java")).forEach(System.out::println);`

Java IO File (NIO.2) - 4

Given code of Test.java file:

```
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.stream.Stream;

public class Test {
    public static void main(String[] args) throws IOException {
        Stream<Path> files =
            Files.list(Paths.get(System.getProperty("user.home")));
        files.forEach(System.out::println);
    }
}
```

System.getProperty("user.home") returns the HOME directory of the User (Both in windows and Linux).

What will be the result of compiling and executing Test class?

- A - It will only print the paths of files (not directories) under HOME directory.
- B - It will print the paths of files (not directories) under HOME directory and its sub-directories.
- C - It will print the paths of directories, sub-directories and files under HOME directory.
- D - It will only print the paths of directories and files under HOME directory.

Java IO File (NIO.2) - 5

Given code of Test.java file:

```
import java.nio.file.Path;
import java.nio.file.Paths;

public class Test {
    public static void main(String[] args) {
        Path path1 = Paths.get("F:\\A\\B\\C");
        Path path2 = Paths.get("F:\\A");
        System.out.println(path1.relativeTo(path2));
        System.out.println(path2.relativeTo(path1));
    }
}
```

What will be the result of compiling and executing Test class?

A -

B\C
..\..\

B - Compilation error

C -

..\..\
B\C

D - An exception is thrown at runtime

Java IO File (NIO.2) - 6

Given code of Test.java file:

```
import java.nio.file.Path;
import java.nio.file.Paths;

public class Test {
    public static void main(String[] args) {
        Path path1 = Paths.get("C:\\A\\B\\C");
        Path path2 = Paths.get("D:\\A");
        System.out.println(path1.relativeTo(path2));
        System.out.println(path2.relativeTo(path1));
    }
}
```

What will be the result of compiling and executing Test class?

A -

B\C
..\..\

B - Compilation error.

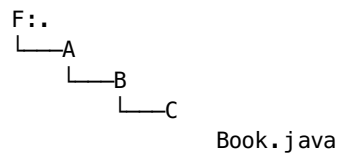
C - An exception is thrown at runtime

D -

..\..\
B\C

Java IO File (NIO.2) - 7

F: is accessible for reading and below is the directory structure for F:



'Book.java' file is available under 'C' directory.

Given code of Test.java file:

```
import java.nio.file.Path;
import java.nio.file.Paths;

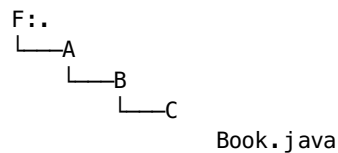
public class Test {
    public static void main(String[] args) {
        Path file = Paths.get("F:\\A\\B\\Book.java");
        System.out.println(file.toAbsolutePath());
    }
}
```

What will be the result of compiling and executing Test class?

- A - F:\A\B\Book.java
- B - Book.java
- C - FileNotFoundException is thrown at runtime.
- D - NoSuchFileException is thrown at runtime.

Java IO File (NIO.2) - 8

F: is accessible for reading and below is the directory structure for F:



Given code of Test.java file:

```
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.io.File;

public class Test {
    public static void main(String[] args) throws IOException{
        Path path = Paths.get("F:\\A\\B");
        /*INSERT*/
    }
}
```

Which of the following statements, if used to replace `/*INSERT*/`, will successfully print 'true' on to the console? Select 3 options.

- A - `System.out.println(Files.getAttribute(path, "isDirectory"));`
- B - `System.out.println(path.toFile().isDirectory());`
- C - `System.out.println(Files.isDirectory(path));`
- D - `System.out.println(new File(path).isDirectory());`
- E - `System.out.println(File.isDirectory(path));`

Java IO File (NIO.2) - 9

C:\ is accessible for reading/writing and below is the content of 'C:\TEMP' folder:

```
C:\TEMP
|
|  msg
|
└─Parent
    └─Child
        Message.txt
```

'msg' is a symbolic link file for 'C:\TEMP\Parent\Child\Message.txt'.

Given code of Test.java file:

```
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;

public class Test {
    public static void main(String[] args) throws IOException {
        Path src = Paths.get("C:", "TEMP", "msg");

        Path tgt = Paths.get("C:", "TEMP", "Parent", "copy");
        Files.copy(src, tgt);

        System.out.println(Files.isSymbolicLink(src) + ":" +
            Files.isSymbolicLink(tgt));
    }
}
```

What will be the result of compiling and executing Test class?

A - false:false

B - true:false

C - false:true

D - true:true

Java IO File (NIO.2) - 10

F: is accessible for reading/writing and below is the directory structure for F:

F:.
└─X

Directory X exists under F:.

Given code of Test.java file:

```
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;

public class Test {
    public static void main(String[] args) throws IOException{
        Path path = Paths.get("F:\\X\\Y\\Z");
        Files.createDirectory(path);
    }
}
```

What will be the result of compiling and executing Test class?

- A - An exception is thrown at runtime.
- B - Directory Y will be created under X and directory Z will be created under Y.
- C - Only directory Y will be created under X.

Java IO File (NIO.2) - 11

Given code of Test.java file:

```
import java.io.*;
import java.nio.file.Path;
import java.nio.file.Paths;

public class Test {
    public static void main(String[] args) throws IOException {
        Path path = Paths.get("F:\\A\\B\\C\\");
        System.out.printf("%d, %s, %s", path.getNameCount(),
            path.getFileName(), path.getName(2));
    }
}
```

F: is accessible for reading/writing but is currently blank.

What will be the result of compiling and executing Test class?

A - 4, C, B

B - Runtime Exception

C - 4, null, B

D - 3, C, B

E - 3, C, C

Java IO File (NIO.2) - 12

Below is the directory structure for F:

```
F:.\n├── A\n│   └── B\n│       └── C\n│           └── Book.java
```

Given code of Test.java file:

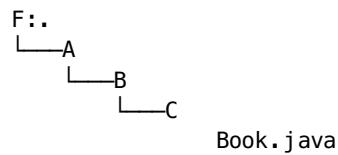
```
import java.io.IOException;\nimport java.nio.file.Files;\nimport java.nio.file.Path;\nimport java.nio.file.Paths;\n\npublic class Test {\n    public static void main(String[] args) throws IOException {\n        Path src = Paths.get("F:\\A\\B\\C\\Book.java");\n        Path tgt = Paths.get("F:\\A\\B");\n        Files.copy(src, tgt);\n    }\n}
```

What will be the result of compiling and executing Test class?

- A - java.nio.file.FileAlreadyExistsException is thrown at runtime
- B - 'Book.java' will be copied successfully to 'F:\\A\\B\\' directory
- C - Program terminates successfully without copying 'Book.java' file
- D - java.io.FileNotFoundException is thrown at runtime

Java IO File (NIO.2) - 13

F: is accessible for reading and below is the directory structure for F:



'Book.java' file is available under 'C' directory.

Given code of Test.java file:

```
import java.nio.file.Path;
import java.nio.file.Paths;

public class Test {
    public static void main(String[] args) {
        Path file = Paths.get("Book.java");
        System.out.println(file.toAbsolutePath());
    }
}
```

Actual path of generated class file is: "C:\classes\Test.class".

What will be the result of compiling and executing Test class?

- A - F:\A\B\Book.java
- B - FileNotFoundException is thrown at runtime.
- C - C:\classes\Book.java
- D - NoSuchFileException is thrown at runtime.

Java IO File (NIO.2) - 14

F: is accessible for reading and below is the directory structure for F:

```
F:.\n└─process\n    file.txt\n    file.docx\n    file.pdf
```

Above 3 files contain HELLO as the only word in the files. “file.txt” is a text file, “file.docx” is a Microsoft Word document and “file.pdf” is a PDF file.

Given code of Test.java file:

```
import java.io.IOException;\nimport java.nio.file.Files;\nimport java.nio.file.Path;\nimport java.nio.file.Paths;\nimport java.util.stream.Stream;\n\npublic class Test {\n    public static void main(String[] args) throws IOException {\n        Stream<Path> paths = Files.walk(Paths.get("F:\\pprocess"));\n        paths.filter(path -> !Files.isDirectory(path)).forEach(\n            path -> {\n                try {\n                    Files.readAllLines(path).stream()\n                        .forEach(System.out::println);\n                } catch (IOException e) {\n                    System.out.println("FAILED");\n                }\n            }\n        );\n    }\n}
```

What will be the result of compiling and executing Test class?

- A - FAILED will be printed three times
- B - HELLO will be printed twice and FAILED will be printed once
- C - HELLO will be printed once and FAILED will be printed twice
- D - HELLO will be printed three times on to the console

Java IO File (NIO.2) - 15

C:\ is accessible for reading/writing and below is the content of 'C:\TEMP' folder:

```
C:\\TEMP
|
|  msg
|
└─Parent
    └─Child
        Message.txt
```

'msg' is a symbolic link file for 'C:\TEMP\Parent\Child\Message.txt'.

Given code of Test.java file:

```
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;

public class Test {
    public static void main(String[] args) throws IOException {
        Path src = Paths.get("C:", "TEMP", "msg");
        Files.delete(src);
    }
}
```

What will be the result of compiling and executing Test class?

- A - The code executes successfully and deletes the file referred by symbolic link 'Message. txt'
- B - The code executes successfully but doesn't delete anything
- C - The code executes successfully and deletes symbolic link file 'msg'
- D - The code executes successfully and deletes all the directories and files in the path 'C:\TEMP\Parent\Child\Message.txt'

Java IO File (NIO.2) - 16

F: is accessible for reading/writing and below is the directory structure for F:

F:.
└─X

Directory X exists under F:.

Given code of Test.java file:

```
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;

public class Test {
    public static void main(String[] args) throws IOException{
        Path path = Paths.get("F:\\X\\Y\\Z");
        Files.createDirectories(path);
    }
}
```

What will be the result of compiling and executing Test class?

A - Only directory Y will be created under X

B - An exception is thrown at runtime

C - Directory Y will be created under X and directory Z will be created under Y

Java IO File (NIO.2) - 17

Given code of Test.java file:

```
import java.nio.file.Path;
import java.nio.file.Paths;

public class Test {
    public static void main(String[] args) {
        Path path = Paths.get("F:", "user", "..", "training..");
        System.out.println(path.normalize());
    }
}
```

What will be the result of compiling and executing Test class?

- A - F:\training..
- B - F:\user
- C - F:\
- D - F:\user\training

Java IO File (NIO.2) - 18

F: is accessible for reading/writing and below is the directory structure for F:

```
F:.  
├── Parent  
│   └── Child  
│       Message.txt  
├── Shortcut  
│   Child.lnk  
└── Other  
    └── Logs
```

Given code of Test.java file:

```
import java.nio.file.Path;  
import java.nio.file.Paths;  
  
public class Test {  
    public static void main(String[] args) {  
        Path path1 = Paths.get("F:", "Other", "Logs");  
        Path path2 = Paths.get("../", "../", "Shortcut", "Child.lnk",  
                                "Message.txt");  
        Path path3 = path1.resolve(path2).normalize();  
        Path path4 = path1.resolveSibling(path2).normalize();  
        System.out.println(path3.equals(path4));    }  
}
```

What will be the result of compiling and executing Test class?

A - true

B - false

C - An exception is thrown at runtime.

Java IO File (NIO.2) - 19

Given code of Test.java file:

```
import java.nio.file.Path;
import java.nio.file.Paths;

public class Test {
    public static void main(String[] args) {
        Path file1 = Paths.get("F:\\A\\B\\C");
        Path file2 = Paths.get("Book.java");
        System.out.println(file1.resolve(file2));
        System.out.println(file1.resolveSibling(file2));
    }
}
```

What will be the result of compiling and executing Test class?

A -

F:\\A\\B\\Book.java
F:\\A\\B\\C\\Book.java

B -

Book.java
F:\\A\\B\\Book.java

C -

Book.java
Book.java

D -

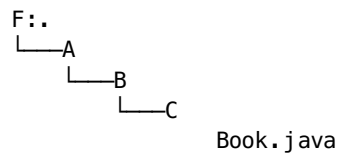
F:\\A\\B\\C\\Book.java
F:\\A\\B\\Book.java

E -

F:\\A\\B\\C\\Book.java
Book.java

Java IO File (NIO.2) - 20

F: is accessible for reading and below is the directory structure for F:



'Book.java' file is available under 'C' directory.

Given code of Test.java file:

```
import java.io.IOException;
import java.nio.file.Path;
import java.nio.file.Paths;

public class Test {
    public static void main(String[] args) throws IOException {
        Path file = Paths.get("F:\\A\\.\\B\\C\\D\\.\\Book.java");
        System.out.println(file.toRealPath());
    }
}
```

What will be the result of compiling and executing Test class?

- A - Book.java
- B - F:\A\B\C\Book.java
- C - NoSuchFileException is thrown at runtime.
- D - F:\A\.\B\C\D\...java
- E - FileNotFoundException is thrown at runtime.

Java IO File (NIO.2) - 21

Given code of Test.java file:

```
import java.nio.file.*;

public class Test {
    public static void main(String[] args) {
        Path path = Paths.get("F:\A");
        System.out.println(path.getRoot().equals(path.getParent()));
    }
}
```

What will be the result of compiling and executing Test class?

A - NullPointerException is thrown at runtime

B - false

C - true

Java IO File (NIO.2) - 22

F: is accessible for reading/writing and below is the directory structure for F:

```
F:.\n├── A\n│   └── B\n│       └── C\n│           └── Book.java
```

Given code of Test.java file:

```
import java.io.IOException;\nimport java.nio.file.Files;\nimport java.nio.file.Path;\nimport java.nio.file.Paths;\n\npublic class Test {\n    public static void main(String[] args) throws IOException{\n        Path src = Paths.get("F:\\A\\B\\C\\Book.java");\n        Path tgt = Paths.get("F:\\A\\B\\Book.java");\n        Path copy = Files.copy(src, tgt);\n        System.out.println(Files.isSameFile(src, copy));\n        System.out.println(Files.isSameFile(tgt, copy));\n    }\n}
```

What will be the result of compiling and executing Test class?

A -

false
false

B -

true
true

C -

false
true

D -

true
false

Java IO File (NIO.2) - 23

Given code of Test.java file:

```
import java.nio.file.Path;
import java.nio.file.Paths;

public class Test {
    public static void main(String[] args) {
        Path path = Paths.get("F:\\A\\B\\C\\Book.java");
        System.out.println(path.subpath(1,5));
    }
}
```

What will be the result of compiling and executing Test class?

- A - A\\B\\C
- B - Exception is thrown at runtime
- C - F:\\A\\B\\C\\Book.java
- D - A\\B\\C\\Book.java

Java IO File (NIO.2) - 24

Given code of Test.java file:

```
import java.nio.file.*;
import java.util.*;

public class Test {
    public static void main(String[] args) {
        Path path = Paths.get("F:\\A\\B\\C\\Book.java");
        /*INSERT*/
    }
}
```

Which of the following statements, if used to replace `/*INSERT*/`, will print below output on to the console?

- A
- B
- C
- Book.java

Select ALL that apply.

A -

```
for(int i = 6; i < path.getNameCount(); i++) {
    System.out.println(path.getName(i));
}
```

B -

```
Iterator<Path> iterator = path.iterator();
while(iterator.hasNext()) {
    System.out.println(iterator.next());
}
```

C -

```
for(Path p : path) {
    System.out.println(p);
}
```

D -

```
path.forEach(System.out::println);
```

Java IO File (NIO.2) - 25

F: is accessible for reading/writing and below is the directory structure for F:

```
F:.\n├── Parent\n│   ├── a.txt\n│   └── b.txt\n└── Child\n    ├── c.txt\n    └── d.txt
```

Given code of Test.java file:

```
import java.io.IOException;\nimport java.nio.file.Files;\nimport java.nio.file.Path;\nimport java.nio.file.Paths;\nimport java.nio.file.attribute.BasicFileAttributes;\nimport java.util.function.BiPredicate;\nimport java.util.stream.Stream;\n\npublic class Test {\n    public static void main(String[] args) throws IOException {\n        Path root = Paths.get("F:");\n        BiPredicate<Path, BasicFileAttributes> predicate = (p,a) ->\n            p.endsWith(null);\n        try(Stream<Path> paths = Files.find(root, 2, predicate))\n        {\n            paths.forEach(System.out::println);\n        }\n    }\n}
```

What will be the result of compiling and executing Test class?

A - Above program executes successfully and prints below lines on to the console:

```
F:Parent\a.txt\nF:Parent\b.txt
```

B - Above program executes successfully and prints nothing on to the console

C - Above program executes successfully and prints below lines on to the console:

```
F:Parent\Child\c.txt\nF:Parent\Child\d.txt
```

F:Parent\a.txt
F:Parent\b.txt

D - Above program executes successfully and prints nothing on to the console

E - Compilation error

Java IO File (NIO.2) - 26

F: is accessible for reading/writing and below is the directory structure for F:

```
F:.\n├── Parent\n│   ├── a.txt\n│   └── b.txt\n└── Child\n    ├── c.txt\n    └── d.txt
```

Given code of Test.java file:

```
import java.io.IOException;\nimport java.nio.file.Files;\nimport java.nio.file.Path;\nimport java.nio.file.Paths;\nimport java.nio.file.attribute.BasicFileAttributes;\nimport java.util.function.BiPredicate;\nimport java.util.stream.Stream;\n\npublic class Test {\n    public static void main(String[] args) throws IOException {\n        Path root = Paths.get("F:");\n        BiPredicate<Path, BasicFileAttributes> predicate = (p,a) ->\n            p.endsWith(".txt");\n        try(Stream<Path> paths = Files.find(root, 2, predicate))\n        {\n            paths.forEach(System.out::println);\n        }\n    }\n}
```

What will be the result of compiling and executing Test class?

A - Above program executes successfully and prints below lines on to the console:

```
F:Parent\\Child\\c.txt\nF:Parent\\Child\\d.txt\nF:Parent\\a.txt\nF:Parent\\b.txt
```

B - Above program executes successfully and prints nothing on to the console

C - Above program executes successfully and prints below lines on to the console:

F:Parent\a.txt
F:Parent\b.txt

Java IO File (NIO.2) - 27

F: is accessible for reading/writing and below is the directory structure for F:

```
F:.  
├── A  
│   └── B  
│       └── Book.java
```

Given code of Test.java file:

```
import java.io.File;  
import java.io.IOException;  
import java.nio.file.*;  
  
public class Test {  
    public static void main(String[] args) throws IOException{  
        Path path = Paths.get("F:\\A\\B\\Book.java");  
        long size1 = Files.size(path);  
  
        File file = new File("F:\\A\\B\\Book.java");  
        long size2 = file.length();  
  
        System.out.println(size1 == size2);  
    }  
}
```

What will be the result of compiling and executing Test class?

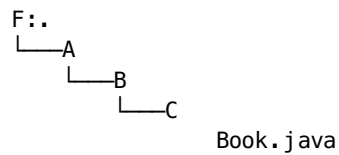
A - Compilation error

B - false

C - true

Java IO File (NIO.2) - 28

F: is accessible for reading and below is the directory structure for F:



'Book.java' file is available under 'C' directory.

Given code of Test.java file:

```
import java.nio.file.Path;
import java.nio.file.Paths;

public class Test {
    public static void main(String[] args) {
        Path file = Paths.get("F:\\A\\.\\B\\C\\D\\..\\Book.java");
        System.out.println(file.toRealPath());
    }
}
```

What will be the result of compiling and executing Test class?

- A - Compilation Error
- B - F:\A\B\C\D\...\java
- C - Book.java
- D - F:\A\B\C\Book.java
- E - NoSuchFileException is thrown at runtime.

Java IO File (NIO.2) - 29

F: is accessible for reading/writing and below is the directory structure for F:

```
F:.  
├── A  
│   └── B  
│       Book.java
```

Book.java is a text file.

Given code of Test.java file:

```
import java.io.BufferedReader;  
import java.io.IOException;  
import java.nio.file.*;  
  
public class Test {  
    public static void main(String[] args) throws IOException{  
        Path src = Paths.get("F:\\A\\B\\Book.java");  
        try(BufferedReader reader = Files.newBufferedReader(src))  
        {  
            String str = null;  
            while((str = reader.readLine()) != null) {  
                System.out.println(str);  
            }  
        }  
    }  
}
```

What will be the result of compiling and executing Test class?

- A - Compilation error
- B - Contents of Book.java are printed on to the console.
- C - An exception is thrown at runtime.

Java IO File (NIO.2) - 30

Given code of Test.java file:

```
import java.nio.file.Path;
import java.nio.file.Paths;

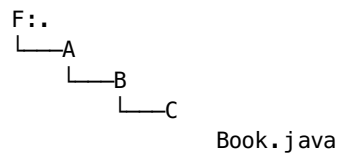
public class Test {
    public static void main(String[] args) {
        Path path = Paths.get("F:\\A\\.\\B\\C\\D\\..\\Book.java");
        path.normalize();
        System.out.println(path);
    }
}
```

What will be the result of compiling and executing Test class?

- A - F:\\A\\B\\C\\Book.java
- B - None of the other options
- C - F:\\A\\B\\C\\D\\Book.java
- D - F:\\A\\.\\B\\C\\D\\..\\Book.java

Java IO File (NIO.2) - 31

F: is accessible for reading and below is the directory structure for F:



Given code of Test.java file:

```
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.nio.file.attribute.BasicFileAttributes;

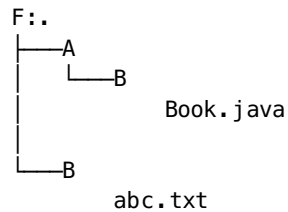
public class Test {
    public static void main(String[] args) throws IOException{
        Path path = Paths.get("F:\\A\\B\\C\\Book.java");
        /*INSERT*/
    }
}
```

Which of the following statements, if used to replace `/*INSERT*/`, will successfully print creation time of 'Book.java' on to the console? Select 3 options.

- A - `System.out.println(Files.readAttributes(path, "*"),creationTime());`
- B - `System.out.println(Files.readAttributes(path, BasicFileAttributes.class).creationTime());`
- C - `System.out.println(Files.getAttribute(path, "creationTime"));`
- D - `System.out.println(Files.readAttributes(path, "*").get("creationTime"));`
- E - `System.out.println(Files.readAttributes(path, BasicFileAttributes.class).get("creationTime"));`

Java IO File (NIO.2) - 32

F: is accessible for reading/writing and below is the directory structure for F:



Given code of Test.java file:

```
import java.io.IOException;
import java.nio.file.*;

public class Test {
    public static void main(String[] args) throws IOException{
        Path src = Paths.get("F:\\A\\B");
        Path tgt = Paths.get("F:\\B");
        Files.move(src, tgt, StandardCopyOption.REPLACE_EXISTING);
    }
}
```

What will be the result of compiling and executing Test class?

- A - An exception is thrown at runtime
- B - Compilation error
- C - Directory B with its contents will move successfully from 'F:\A\B' to 'F:\B'

Java IO File (NIO.2) - 33

Given code of Test.java file:

```
import java.nio.file.Path;
import java.nio.file.Paths;

public class Test {
    public static void main(String[] args) {
        Path file1 = Paths.get("F:\\A\\B");
        Path file2 = Paths.get("F:\\A\\B\\C\\Book.java");

        System.out.println(file1.resolve(file2).equals(file1.resolveSibling(file2)));
    }
}
```

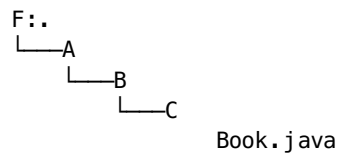
What will be the result of compiling and executing Test class?

A - true

B - false

Java IO File (NIO.2) - 34

F: is accessible for reading and below is the directory structure for F:



'Book.java' file is available under 'C' directory.

Given code of Test.java file:

```
import java.nio.file.Path;
import java.nio.file.Paths;

public class Test {
    public static void main(String[] args) {
        Path file = Paths.get("F:\\A\\.\\B\\C\\D\\.\\.\\Book.java");
        System.out.println(file.toAbsolutePath());
    }
}
```

What will be the result of compiling and executing Test class?

- A - Compilation Error
- B - Book.java
- C - F:\A\B\C\D\..\Book.java
- D - F:\A\B\C\Book.java
- E - NoSuchFileException is thrown at runtime

Java IO File (NIO.2) - 35

F: is currently blank and accessible for Reading/Writing.

Given code of Test.java file:

```
package com.training.ocp;

import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;

public class Test {
    public static void main(String[] args) throws IOException {
        Path path = Paths.get("F:", "A", "B", "File.txt");

        /*INSERT*/
    }
}
```

Test.class file is under 'C:'.

Which of the following statements, if used to replace `/*INSERT*/`, will successfully create the file "File.txt" under F:directory?

A -

```
Files.createDirectories(path.getParent());
Files.createFile(path.getFileName());
```

B -

```
Files.createDirectories(path);
Files.createFile(path);
```

C -

```
Files.createDirectories(path.getParent());
Files.createFile(path);
```

D - Files.createFile(path);