

DefNeo  
36 уровень

12.09.2016   13112   29

## Уровень 31. Ответы на вопросы к собеседованию по теме уровня

Статья из группы Архив info.javarush.ru  
15167 участников

Присоединиться



### 1. Может ли объект `File` соответствовать файлу, которого еще нет?

Да, если передать в конструктор значение директории.

```
1 String dirPath = "/";
2 File f = new File(dirPath);
3 File[] files = f.listFiles();
```

Так делают, например, для того, чтобы получить массив файлов.

```
1 public class MyClass {
2     public static void main(String[] args) {
3         boolean isObject = false;
4
5
6         File file = new File("/");
7         if (file instanceof Object){
8             isObject = true;
9         }
10    }
```

НАЧАТЬ ОБУЧЕНИЕ

```
11      Это из документации
12      System.out.println(isObject + " " + isFile);
13
14  }
15 }
```

Вывод:

true false

File наследуется от object. Ответ: да! Жду комментариев.

2. Как преобразовать объект File к типу Path ?

Метод toPath();

```
1      toPath(); //Returns a java.nio.file.Path object constructed from the this abstract path.
```

3. Зачем нужен класс Files?

Взяли за основу класс File, добавили в него немного нового, переименовывали методы, а в конце еще и разделили на два. Так что теперь есть два новых класса – Path и Files.

Path – это, фактически новый аналог класса File, а Files – это утилитный класс (по аналогии с классами Arrays & Collections), в него вынесли все статические методы класса File. Так «правильнее» с точки зрения ООП.М

Немного из документов:

```
public final class Files
extends Object
```

This class consists exclusively of static methods that operate on files, directories, or other types of files. In most cases, the methods defined here will delegate to the associated file system provider to perform the file operations.

4. Какие классы для архивации вы знаете?

Неплохая статья на эту тему и выдержка из нее: [Архивация в Java](#)

Для работы с архивами в спецификации Java существуют два пакета – java.util.zip и java.util.jar соответственно для архивов zip и jar. Различие форматов jar и zip заключается только в расширении архива zip. Пакет java.util.jar аналогичен пакету java.util.zip, за исключением реализации конструкторов и метода voidputNextEntry(ZipEntry e) класса JarOutputStream. Ниже будет рассмотрен только пакет java.util.jar. Чтобы переделать все примеры на использование zip-архива, достаточно всюду в коде заменить Jar на Zip.

5. Как добавить директорию в архив?

Для себя я понял этот вопрос, как добавление пустой директории в готовый архив. Никаких рабочих примеров я не нашел. Вот код: (Он наглядно показывает, что можно в архив положить любой файл, а вот с пустой директорией... я не знаю как ответить, постить на StackOverFlow не стал, за такой вопрос заминусят точно) Если у кого есть предложения, то напишите.

```
1      public class Main {
2          public static void main(String[] args) {
3              String[] myFiles = {"D:\\forJava\\MyArtifactName\\packForTest\\res2.txt",
4                                  "D:\\forJava\\MyArtifactName\\packForTest\\res.txt",
5                                  "D:\\forJava\\MyArtifactName\\packForTest\\res4.txt",
6                                  "D:\\forJava\\MyArtifactName\\packForTest\\testDir\\"
7                                  };
8              String zipFile = "D:\\forJava\\MyArtifactName\\packForTest\\res.zip";
```

```
11         zipUtil.zip(myFiles, zipFile);
12
13     } catch (Exception ex) {
14         // some errors occurred
15         ex.printStackTrace();
16     }
17 }
18 }
```

Вопрос о последней testDir, ее то как раз в получившийся архив JVM не кладет, со всеми остальными txt – файлами норм получается.

ZipUtility.java:

```
1  import java.io.BufferedInputStream;
2  import java.io.File;
3  import java.io.FileInputStream;
4
5  import java.io.FileOutputStream;
6  import java.io.IOException;
7  import java.util.ArrayList;
8  import java.util.List;
9  import java.util.zip.ZipEntry;
10 import java.util.zip.ZipOutputStream;
11
12 public class ZipUtility {
13
14     private static final int BUFFER_SIZE = 4096;
15
16     public void zip(List<File> listFiles, String destZipFile) throws IOException {
17         ZipOutputStream zos = new ZipOutputStream(new FileOutputStream(destZipFile));
18         for (File file : listFiles) {
19             if (file.isDirectory()) {
20                 zipDirectory(file, file.getName(), zos);
21             } else {
22                 zipFile(file, zos);
23             }
24         }
25         zos.flush();
26         zos.close();
27     }
28
29     public void zip(String[] files, String destZipFile) throws IOException {
30         List<File> listFiles = new ArrayList<File>();
31         for (int i = 0; i < files.length; i++) {
32             listFiles.add(new File(files[i]));
33         }
34         zip(listFiles, destZipFile);
35     }
36
37     private void zipDirectory(File folder, String parentFolder, ZipOutputStream zos) throws IOEx
38         for (File file : folder.listFiles()) {
39             if (file.isDirectory()) {
40                 zipDirectory(file, parentFolder + "/" + file.getName(), zos);
```

НАЧАТЬ ОБУЧЕНИЕ

```
43         zos.putNextEntry(new ZipEntry(parentFolder + "/" + file.getName()));
44         BufferedInputStream bis = new BufferedInputStream(
45             new FileInputStream(file));
46         long bytesRead = 0;
47         byte[] bytesIn = new byte[BUFFER_SIZE];
48         int read = 0;
49         while ((read = bis.read(bytesIn)) != -1) {
50             zos.write(bytesIn, 0, read);
51             bytesRead += read;
52         }
53         zos.closeEntry();
54     }
55 }
56
57 private void zipFile(File file, ZipOutputStream zos)
58     throws IOException {
59     zos.putNextEntry(new ZipEntry(file.getName()));
60     BufferedInputStream bis = new BufferedInputStream(new FileInputStream(
61         file));
62     long bytesRead = 0;
63     byte[] bytesIn = new byte[BUFFER_SIZE];
64     int read = 0;
65     while ((read = bis.read(bytesIn)) != -1) {
66         zos.write(bytesIn, 0, read);
67         bytesRead += read;
68     }
69     zos.closeEntry();
70 }
71 }
```

Код [отсюда](#)

6. Зачем нужны **Properties** ?

**Properties** – это файл свойств. Структура его: ключ – значение. Для работы с такими файлами в Java есть класс **Properties**, он унаследован от **HashTable<Object, Object>**

Есть статья про манипуляции с ним — [Java Properties file examples](#)

7. В каком виде хранятся данные в файле **.properties** ?

Ключ – значение.

8. Можно ли изменять данные в объекте **Properties** после загрузки их из файла?

Если он унаследован от **HashMap**, тогда можно, только потом нужно будет изменения в этот файл отписать. Для этого есть метод **setProperty**.

Вот код:

```
1  import java.io.FileInputStream;
2  import java.io.IOException;
3  import java.io.InputStream;
4  import java.util.Properties;
5
6  /**
7   * Created by Роман on 12.09.2016
```

```
9      public class LoadAndSetProperties {
10
11          public static void main(String[] args) {
12
13              Properties prop = new Properties();
14              InputStream input = null;
15              try {
16
17                  input = new FileInputStream("D:\\forJava\\MyArtifactName\\packForTest\\config.properties");
18
19                  // load a properties file
20                  prop.load(input);
21
22                  // get the property value and print it out
23
24
25                  prop.setProperty("database", "ddfdfdfdfdf");
26                  System.out.print(prop.getProperty("database"));
27              } catch (IOException ex) {
28                  ex.printStackTrace();
29              } finally {
30                  if (input != null) {
31                      try {
32                          input.close();
33                      } catch (IOException e) {
34                          e.printStackTrace();
35                      }
36                  }
37              }
38          }
39
40      }
```

Вывод:

**ddfdfdfdfdf**

9. Зачем нужен класс `FileReader` ?

Java Docs:

```
public class FileReader
extends InputStreamReader
```

Convenience class for reading character files. The constructors of this class assume that the default character encoding and the default byte-buffer size are appropriate. To specify these values yourself, construct an `InputStreamReader` on a `FileInputStream`.

`FileReader` is meant for reading streams of characters.

Класс для чтения символов файлов. Конструкторы этого класса предполагают, что кодировка символов дефолтная и дефолтный размер буфера являются подходящими. Чтобы задать эти значения самостоятельно, следует построить `InputStreamReader` над `FileInputStream`. `FileReader` предназначен для считывания потоков символов.

10. Зачем нужен класс `FileWriter` ?

```
public class FileWriter
extends OutputStreamWriter
```

НАЧАТЬ ОБУЧЕНИЕ



Convenience class for writing character files. The constructors of this class assume that the default character encoding and the default byte-buffer size are acceptable. To specify these values yourself, construct an `OutputStreamWriter` on a `FileOutputStream`.

Whether or not a file is available or may be created depends upon the underlying platform. Some platforms, in particular, allow a file to be opened for writing by only one `FileWriter` (or other file-writing object) at a time. In such situations the constructors in this class will fail if the file involved is already open. `FileWriter` is meant for writing streams of characters.

Класс для записи символов файлов. Конструкторы этого класса предполагают, что кодировка символов дефолтная и дефолтный размер буфера являются приемлемым. Чтобы задать эти значения самостоятельно, следует построить `OutputStreamWriter` над `FileOutputStream`. Является ли файл доступен для записи, зависит от используемой платформы. Некоторые платформы разрешают держать файл для записи только одним `FileWriter` (или другого объекта записи файла), в одно время. `FileWriter` предназначен для записи потоков символов. Для написания потоков необработанных байтов, используйте `FileOutputStream`.

Эти классы (`FileReader` и `FileWriter`) специально ориентированы для работы с текстом и строками.

−

+90

+

Комментарии (29)

популярные

новые

старые

JavaCoder

Введите текст комментария

BogarD

Уровень 35, Kherson, Украина

30 ноября 2021, 14:10

⋮

**5. Как добавить директорию в архив?**

Про `zip.putNextEntry(new ZipEntry("folder/"))` не слышали. Это же элементарщина.

**6. Зачем нужны Properties?**

**Properties – это файл свойств.**

Нет. `*.properties` – это файл свойств, а `Properties` – это класс, для работы с файлами `*.properties`

**7. В каком виде хранятся данные в файле .properties?**

**Ключ – значение.**

И снова ошибка, они хранятся в виде `ключ=значение`. Если считаете это незначительной ошибкой, то скажите это при собеседовании.

Ответить

−

+6

+

Анастасия Ольшанская

Уровень 31

9 ноября 2021, 17:52

⋮

4. Какие классы для архивации вы знаете?

Тут про классы спрашивают, а не про пакеты..

Я бы ответила так:

`ZipInputStream` и `ZipOutputStream`, `JarInputStream` и `JarOutputStream`, `JarEntry` и `ZipEntry`.

Плюс наверное нужны классы для работы с самими файлами `File`, `Files`, `Paths`, `Path` (интерфейс), иначе что нам архивировать.

Но, это только моё мнение.

Ответить

−

+2

+

Pavel Soros

Уровень 34, Санкт-Петербург, Россия

23 февраля 2021, 19:47

⋮

Ответ на первый вопрос : Да. Но логика при помощи которой пришли к этому ответу не понятна. А если вместо разделительного символа в конструктор передать любой другой символ или путь не к существующему файлу, разве не возникнет ситуация в которой объект есть, а файла нет? Манипуляции с объектом в коде возможны, хотя по факту, файла ещё нет. То есть, объект соответствует файлу которого нет.

Ответить

−

+3

+

barracuda

Уровень 41, Санкт-Петербург, Россия

EXPERT

18 января 2021, 11:56

⋮

По первому вопросу:

```
File file = new File("fdfdsafdsasa");
```

Файла еще нет, а объект есть.

НАЧАТЬ ОБУЧЕНИЕ



Marina86

Уровень 36, Москва, Россия

3 ноября 2017, 23:23

...

Дополнение к пункту 8 — «Можно ли изменять данные в объекте Properties после загрузки их из файла?»

Чтобы сохранить добавленное/измененное свойство в файле(а в этом как раз и весь смысл ибо сейчас свойство есть лишь в runtime) добавьте

prop.store(new FileOutputStream(«D:\\forJava\\MyArtifactName\\packForTest\\config.properties»), «comments»);

Ответить

+

+13

+

↺ Показать еще комментарии

ОБУЧЕНИЕ

- Курсы программирования
- Курс Java
- Помощь по задачам
- Подписки
- Задачи-игры

СООБЩЕСТВО

- Пользователи
- Статьи
- Форум
- Чат
- Истории успеха
- Активности

КОМПАНИЯ

- О нас
- Контакты
- Отзывы
- FAQ
- Поддержка



JavaRush — это интерактивный онлайн-курс по изучению Java-программирования с нуля. Он содержит 1200 практических задач с проверкой решения в один клик, необходимый минимум теории по основам Java и мотивирующие фишки, которые помогут пройти курс до конца: игры, опросы, интересные проекты и статьи об эффективном обучении и карьере Java-девелопера.

ПОДПИСЫВАЙТЕСЬ

ЯЗЫК ИНТЕРФЕЙСА

Русский

▼

