Класс File

Класс File, определенный в пакете java.io, не работает напрямую с потоками. Его задачей является управление информацией о файлах и каталогах.

Хотя на уровне операционной системы файлы и каталоги отличаются, но в Java они описываются одним классом File.

Методы класса File

Метод	Описание
`boolean isDirectory()`	Является ли «объект файла» директорией
`boolean isFile()`	Является ли объект файлом
`long length()`	Возвращает размер/длину файла в байтах.
`boolean createNewFile()`	Создает файл. Если такой файл уже был, возвращает false.
`boolean mkdir()`	Создает директорию. Название mkdir происходит от «make directory».
`boolean mkdirs()`	Создает директорию и все поддиректории.
`boolean delete()`	Удаляет файл объекта на диске. Если объект – директория, то только, если в ней нет файлов.
`void deleteOnExit()`	Добавляет файл в специальный список файлов, которые будут автоматически удалены при закрытии программы.

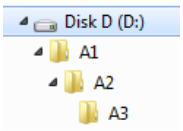
`File createTempFile(`
`String prefix,`
`String suffix,`
`File directory)`
	Создает «временный файл» — файл с случайно сгенерированным уникальным именем – что-типа «dasd4d53sd».
	Дополнительные параметры – префикс к имени, суффикс (окончание). Если директория не указана, то файл создается в специальной директории ОС для временных файлов

`boolean exists()`	Возвращает true, если файл с таким именем существует на диске компьютера.
`String getAbsolutePath()`	Возвращает полный путь файла со всеми поддиректориями.
`String getCanonicalPath()`	Возвращает канонический путь файла. Например, преобразовывает путь «c:/dir/dir2/../a.txt» к

	пути «c:/dir/a.txt»
`String[] list()`	Возвращает массив имен файлов, которые содержатся в директории, которой является текущий объект-файл.
`File[] listFiles()`	Возвращает массив файлов, которые содержатся в директории, которой является текущий объект-файл.
`long getTotalSpace()`	Возвращает размер диска (количество байт) на котором расположен файл.
`long getFreeSpace()`	Возвращает количество свободного места (количество байт) на диске, на котором расположен файл.
`boolean renameTo(File)`	Переименовывает файл – содержимое файла фактически получает новое имя. Т.е. можно переименовать файл «c:/dir/a.txt» в «d:/out/text/b.doc».
`String getName()`	Возвращает только имя файла, без пути.
`String getParent()`	Возвращает только путь (директорию) к текущему файлу, без самого имени.
`Path toPath()`	Возвращает объект Path, который соответствует текущему объекту File.

Пример 1. mkdirs

Создать вложенные папки на диске D:\A1\A2\A3




```
import java.io.*;
public class Main {

    public static void main(String... args) {

        File f = new File("d://A1//A2//A3");

        if (f.mkdirs()) {

            System.out.println("Directory is created");
        }
        else {

            System.out.println("Directory cannot be created");
        }
    }}
```

Пример 2. mkdir

Создать вложенную папку A4 на диске D:\A1\A2\A3**A4**

В конец предыдущего примера добавим две строки:

```
File ff = new File("d://A1//A2//A3//A4");
ff.mkdirs();
```

Пример 3. getAbsolutePath и isDirectory

В конец предыдущего примера добавим строку:

```
System.out.println(ff.getAbsolutePath()+" is directory - "+ff.isDirectory());
```

Пример 4. delete

Удаляет файл объекта на диске. Если объект – директория, то только, если в ней нет файлов.

В конец предыдущего примера добавим строки:

```
if(ff.delete())
{
    System.out.println("File deleted successfully");
}
else
{
    System.out.println("Failed to delete the file");
}
```

При этом будет удалена папка **A4**.

Пример 5.deleteOnExit

Добавляет файл в специальный список файлов, которые будут автоматически удалены при закрытии программы.

Создайте на диске папку **ThisFolderMustBeDeletedOnExit** с помощью Проводника.

```
import java.io.*;
public class Main {

    public static void main(String... args) {

        File f = new File("d://ThisFolderMustBeDeletedOnExit");

        f.deleteOnExit();
        System.out.print(f + " :- This file will delete on exit. ");

    }
}
```

Проверьте, что папка действительно была удалена после закрытия программы.

Пример 6. createNewFile

Создает файл. Если такой файл уже был, возвращает false.

```
import java.io.*;
public class Main {

    public static void main(String... args) {

        File f = null;
        boolean b = false;

        try {

            f = new File("d://test.txt");
            b = f.createNewFile();

            System.out.println("File created: "+b);

        } catch(Exception e) {
            e.printStackTrace();
        }
    }
}
```

Пример 7. exists

Возвращает true, если файл с таким именем существует на диске компьютера.

```
import java.io.*;
public class Main {

    public static void main(String... args) {

        File f = new File("d://test.txt");

        if (f.exists())
            System.out.println("Exists");
        else
            System.out.println("Does not Exists");
    }
}
```


Пример 8. getName, getParent, getPath, getAbsolutePath, getCanonicalPath

Путь может быть абсолютным или относительным.

Полный или абсолютный путь — это путь, который указывает на одно и то же место в файловой системе, вне зависимости от текущего рабочего каталога или других обстоятельств. Полный путь всегда начинается с корневого каталога.

```
C:\user\docs\Letter.txt
A:Picture.jpg
```

Относительный путь представляет собой путь по отношению к текущему рабочему каталогу пользователя или активных приложений.

```
\..\.\file.txt
.. – родительский каталог
. – текущий каталог
```

getName - Возвращает только имя файла, без пути.

getParent - возвращает только путь (директорию) к текущему файлу, без самого имени.

getPath - возвращает представление *String* абстрактного пути к файлу.

getAbsolutePath - возвращает полный путь файла со всеми поддиректориями.

getCanonicalPath возвращает полный путь файла со всеми поддиректориями.

Канонический путь может зависеть от платформы, на которой выполняется приложение.

getPath() получает строку пути, с которой был создан объект File, и может быть относительным текущим каталогом.

getAbsolutePath() получает строку пути после ее разрешения против текущего каталога, если он относительный, что приводит к полностью квалифицированному пути.

getCanonicalPath() получает строку пути после разрешения любого относительного пути к текущему каталогу и удаляет любые относительные пути (. и ..), а любые ссылки файловой системы возвращают путь, который файловая система считает каноническим означает ссылку на объект файловой системы, на который он указывает.


```
import java.io.File;

public class Main {

    public static void main(String[] args) {

        File f = new File("test/.././file.txt");

        System.out.println(f.getPath());

        System.out.println(f.getAbsolutePath());

        try {
            System.out.println(f.getCanonicalPath());
        }
        catch(Exception e) { e.printStackTrace();}
}
}
```

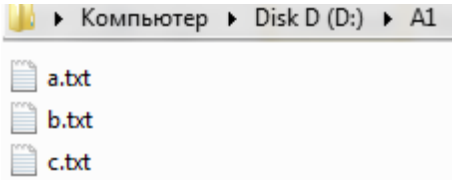
Вывод (приблизительно):

```
test\..\.\file.txt

C:\Users\Oleandr\IdeaProjects\pro_001\test\..\.\file.txt

C:\Users\Oleandr\IdeaProjects\pro_001\file.txt
```

Пример 9. list

list - возвращает массив имен файлов, которые содержатся в директории, которой является текущий объект-файл.

Пусть в папке A1, расположенной на диске D, есть файлы a.txt, b.txt, c.txt:



Нужно вывести имена файлов, которые находятся в папке D:\A1\

```
import java.io.*;
public class Main {

    public static void main(String... args) {

        File f = new File("D://A1");

        String[] List = f.list();

        for (String fName : List)

            System.out.println(fName);
    }}
```


Пример 10. listFiles, length

listFiles - возвращает массив файлов, которые содержатся в директории, которой является текущий объект-файл.

Length- возвращает размер/длину файла в байтах

```
import java.io.*;
public class Main {

    public static void main(String... args) {

        File f = new File("D://A1");

        File[] ff = f.listFiles();

        for (File elf : ff)
            System.out.println(elf.length());
    }
}
```

Пример 11. renameTo(File)

renameTo - переименовывает файл – содержимое файла фактически получает новое имя.

```
import java.io.*;
public class Main {
```

```
    public static void main(String... args) {

        File f = new File("d://A1");

        File newf = new File("d://AAA");

        if (f.renameTo(newf)) {
            System.out.println("File is renamed");
        }
        else {
            System.out.println("File cannot be renamed");
        }
    }
}
```

Пример 12. canWrite, canRead, lastModified

canRead() - проверка, можно ли читать данные из файла.

canWrite() - проверка, можно ли записывать данные в файл.

isHidden() - проверка, являются ли каталог или файл скрытыми.

lastModified() - функция определения даты последней модификации файла.

```
import java.io.*;
public class Main {

public static void main(String... args) {

        File fl = new File("d:\\test.txt");

        System.out.println ("Имя файла: " + fl .getName());

        System.out.println ("Полный путь: " + fl.getAbsolutePath());

        System.out.println ("Родительский каталог: " + fl.getParent());

        System.out.println (fl.exists() ?
                "Файл существует" :
                "Файл  не существует");

        System.out.println (fl.canWrite() ?
                "Свойство - можно записывать" :
                "Свойство - нельзя записывать");

        System.out.println (fl.canRead() ?
                "Свойство - можно читать" :
                "Свойство - нельзя читать");

        System.out.println ("Это директория ? " +
                (fl.isDirectory() ?
                        "да": " нет"));

        System.out.println ("Это обычный файл ? " +
                (fl.isFile() ?
                        "да" : "нет"));

        System.out.println ("Последняя модификация файла : "
                + fl.lastModified());

        System.out.println ("Размер файла : " + fl.length() + " bytes");
    }}
```

Задания

Создайте методы, которые выполняют следующие действия:

1. Создать n (n<100) текстовых файлов с именами f_001.txt, f_002.txt, …, f_034.txt, ….

2. Записать в файлы из предыдущего задания целое число - номер файла. Вывести суммарный размер файлов.

3. Записать в 10 файлов случайные целые числа в диапазоне от 1 до 100.

4. Посчитать сумму чисел, записанных в файлы в задании 3.

5. Переименовать файлы из задания 1 в файлы с именами file_001.txt, file_002.txt, …

6. Найти максимальный размер для файлов из задания 3.

7. Переименовать файлы из задания 1 в файлы с именами file_001.txt, file_002.txt, …

8. Вывести на экран размер диска в байтах, свободное пространство на диске.