Ввод-вывод, доступ к файловой системе

Алексей Владыкин

java.io.File

```
// on Windows:
File javaExecutable = new File(
    "C:\\jdk1.8.0_60\\bin\\java.exe");

File networkFolder = new File(
    "\\\\server\\share");


// on Unix:
File lsExecutable = new File("/usr/bin/ls");
```

Сборка пути

```
String sourceDirName = "src";
String mainFileName = "Main.java";

String mainFilePath = sourceDirName
        + File.separator
        + mainFileName;
```

```
File mainFile =
        new File(sourceDirName, mainFileName);
```

Абсолютные и относительные пути

```
File absoluteFile = new File("/usr/bin/java");
absoluteFile.isAbsolute(); // true
absoluteFile.getAbsolutePath();
    // /usr/bin/java
```

```
File relativeFile = new File("readme.txt");
relativeFile.isAbsolute(); // false
relativeFile.getAbsolutePath();
    // /home/stepic/readme.txt
```

Разбор пути

```
File file = new File("/usr/bin/java");

String path = file.getPath();  // /usr/bin/java

String name = file.getName();  // java

String parent = file.getParent(); // /usr/bin
```

Канонические пути

```
File file = new File("./prj/../symlink.txt");

String canonicalPath = file.getCanonicalPath();
    // "/home/stepic/readme.txt"
```

Работа с файлами

```
File java = new File("/usr/bin/java");
java.exists();       // true
java.isFile();       // true
java.isDirectory(); // false

java.length();       // 1536
java.lastModified();// 1231914805000
```

Работа с директориями

```
File usrbin = new File("/usr/bin");
usrbin.exists();       // true
usrbin.isFile();       // false
usrbin.isDirectory(); // true

usrbin.list();         // String[]
usrbin.listFiles();    // File[]
```

Фильтрация файлов

```
File[] javaSourceFiles = dir.listFiles(
        f -> f.getName().endsWith(".java"));
```

```
// java.io.FileFilter:
//    boolean accept(File pathname)

// java.io.FilenameFilter:
//    boolean accept(File dir, String name)
```

Создание файла

```
try {
    boolean success = file.createNewFile();
} catch (IOException e) {
    // handle error
}
```

Создание директории

```
File dir = new File("a/b/c/d");

boolean success = dir.mkdir();

boolean success2 = dir.mkdirs();
```

Удаление файла или директории

```
boolean success = file.delete();
```

Переименование/перемещение

```
boolean success = file.renameTo(targetFile);
```

java.nio.file.Path

```
Path path = Paths.get("prj/stepic");

File fromPath = path.toFile();

Path fromFile = fromPath.toPath();
```

Разбор пути

```
Path java = Paths.get("/usr/bin/java");
java.isAbsolute();    // true
java.getFileName();   // java
java.getParent();     // /usr/bin

java.getNameCount();              // 3
java.getName(1);                  // bin
java.resolveSibling("javap"); // /usr/bin/javap
java.startsWith("/usr");          // true
Paths.get("/usr").relativize(java); // bin/java
```

Работа с файлами

```
Path java = Paths.get("/usr/bin/java");
Files.exists(java);           // true
Files.isRegularFile(java);    // true
Files.size(java);             // 1536
Files.getLastModifiedTime(java)
        .toMillis();          // 1231914805000


Files.copy(java,
        Paths.get("/usr/bin/java_copy"),
        StandardCopyOption.REPLACE_EXISTING);
```

Работа с директориями

```
Path usrbin = Paths.get("/usr/bin");
Files.exists(usrbin);          // true
Files.isDirectory(usrbin);     // true

try (DirectoryStream<Path> dirStream =
        Files.newDirectoryStream(usrbin)) {
    for (Path child : dirStream) {
        System.out.println(child);
    }
}
```

Создание директории

```
Path dir = Paths.get("a/b/c/d");

Files.createDirectory(dir);

Files.createDirectories(dir);
```

Рекурсивное удаление

```
Path directory = Paths.get("/tmp");

Files.walkFileTree(directory, new SimpleFileVisitor<Path>() {

    @Override
    public FileVisitResult visitFile(
            Path file, BasicFileAttributes attrs)
            throws IOException {
        Files.delete(file);
        return FileVisitResult.CONTINUE;
    }

    @Override
    public FileVisitResult postVisitDirectory(
            Path dir, IOException exc) throws IOException {
        if (exc == null) {
            Files.delete(dir);
            return FileVisitResult.CONTINUE;
        } else {
            throw exc;
        }
    }

});
```

Виртуальные файловые системы

```
Path zipPath = Paths.get("jdk1.8.0_60/src.zip");

try (FileSystem zipfs = FileSystems.newFileSystem(zipPath, null))
    for (Path path : zipfs.getRootDirectories()) {
        Files.walkFileTree(path, new SimpleFileVisitor<Path>() {

            @Override
            public FileVisitResult visitFile(
                    Path file, BasicFileAttributes attrs)
                    throws IOException {
                System.out.println(file);
                return FileVisitResult.CONTINUE;
            }
        });
    }
}
```

Потоки байт

- Ввод данных
  `java.io.InputStream`
- Вывод данных
  `java.io.OutputStream`

```java
package java.io;

public abstract class InputStream implements Closeable {

    public abstract int read() throws IOException;

    public int read(byte b[]) throws IOException {
        return read(b, 0, b.length);
    }

    public int read(byte b[], int off, int len)
            throws IOException {
        // ...
    }

    public long skip(long n) throws IOException {
        // ...
    }

    public void close() throws IOException {}

    // ...
}
```

```
package java.io;

public abstract class OutputStream
        implements Closeable, Flushable {

    public abstract void write(int b) throws IOException;

    public void write(byte b[]) throws IOException {
        write(b, 0, b.length);
    }

    public void write(byte b[], int off, int len)
            throws IOException {
        // ...
    }

    public void flush() throws IOException {
        // ...
    }

    public void close() throws IOException {
        // ...
    }
}
```

Копирование InputStream -> OutputStream

```
int totalBytesWritten = 0;
byte[] buf = new byte[1024];
int blockSize;
while ((blockSize = inputStream.read(buf)) > 0) {
    outputStream.write(buf, 0, blockSize);
    totalBytesWritten += blockSize;
}
```

```
InputStream inputStream =
    new FileInputStream(new File("in.txt"));


OutputStream outputStream =
    new FileOutputStream(new File("out.txt"));
```

```
InputStream inputStream =
    Files.newInputStream(Paths.get("in.txt"));
```

```
OutputStream outputStream =
    Files.newOutputStream(Paths.get("out.txt"));
```

```
try (InputStream inputStream =
        Main.class.getResourceAsStream("Main.class")) {

    int read = inputStream.read();
    while (read >= 0) {
        System.out.printf("%02x", read);
        read = inputStream.read();
    }

}
```

```
try (Socket socket = new Socket("ya.ru", 80)) {

    OutputStream outputStream = socket.getOutputStream();
    outputStream.write("GET / HTTP/1.0\r\n\r\n".getBytes());
    outputStream.flush();

    InputStream inputStream = socket.getInputStream();
    int read = inputStream.read();
    while (read >= 0) {
        System.out.print((char) read);
        read = inputStream.read();
    }

}
```

```
byte[] data = {1, 2, 3, 4, 5};
InputStream inputStream =
    new ByteArrayInputStream(data);



ByteArrayOutputStream outputStream =
    new ByteArrayOutputStream();
// ...
byte[] result = outputStream.toByteArray();
```

```
package java.io;

public class DataOutputStream
    extends FilterOutputStream implements DataOutput {

    public DataOutputStream(OutputStream out) {
        // ...
    }

    public final void writeInt(int v) throws IOException {
        out.write((v >>> 24) & 0xFF);
        out.write((v >>> 16) & 0xFF);
        out.write((v >>>  8) & 0xFF);
        out.write((v >>>  0) & 0xFF);
        incCount(4);
    }

    // ...
}
```

```
package java.io;

public class DataInputStream
    extends FilterInputStream implements DataInput {

    public DataInputStream(InputStream in) {
        // ...
    }

    public final int readInt() throws IOException {
        int ch1 = in.read();
        int ch2 = in.read();
        int ch3 = in.read();
        int ch4 = in.read();
        if ((ch1 | ch2 | ch3 | ch4) < 0)
            throw new EOFException();
        return ((ch1 << 24) + (ch2 << 16)
              + (ch3 << 8) + (ch4 << 0));
    }

    // ...
}
```

```
byte[] originalData = {1, 2, 3, 4, 5};

ByteArrayOutputStream os = new ByteArrayOutputStream();
try (OutputStream dos = new DeflaterOutputStream(os)) {
    dos.write(originalData);
}

byte[] deflatedData = os.toByteArray();

try (InflaterInputStream iis = new InflaterInputStream(
        new ByteArrayInputStream(deflatedData))) {
    int read = iis.read();
    while (read >= 0) {
        System.out.printf("%02x", read);
        read = iis.read();
    }
}
```

Потоки символов

- Ввод данных
  `java.io.Reader`
- Вывод данных
  `java.io.Writer`

```java
package java.io;

public abstract class Reader implements Readable, Closeable {

    public int read() throws IOException {
        // ...
    }

    public int read(char cbuf[]) throws IOException {
        return read(cbuf, 0, cbuf.length);
    }

    public abstract int read(char cbuf[], int off, int len)
            throws IOException;

    public long skip(long n) throws IOException {
        // ...
    }

    public abstract void close() throws IOException;

    // ...
}
```

```
package java.io;

public abstract class Writer
        implements Appendable, Closeable, Flushable {

    public void write(int c) throws IOException {
        // ...
    }

    public void write(char cbuf[]) throws IOException {
        write(cbuf, 0, cbuf.length);
    }

    public abstract void write(char cbuf[], int off, int len)
            throws IOException;

    public abstract void flush() throws IOException;

    public abstract void close() throws IOException;

    // ...
}
```

```
Reader reader =
  new InputStreamReader(inputStream, "UTF-8");
```

```
Charset charset = StandardCharsets.UTF_8;
Writer writer =
  new OutputStreamWriter(outputStream, charset);
```

```java
Reader reader = new FileReader("in.txt");

Writer writer = new FileWriter("out.txt");



Reader reader2 = new InputStreamReader(
    new FileInputStream("in.txt"), StandardCharsets.UTF_8);

Writer writer2 = new OutputStreamWriter(
    new FileOutputStream("out.txt"), StandardCharsets.UTF_8);
```

```
Reader reader = new CharArrayReader(
    new char[] {'a', 'b', 'c'});

Reader reader2 = new StringReader("Hello World!");



CharArrayWriter writer = new CharArrayWriter();
writer.write("Test");
char[] resultArray = writer.toCharArray();

StringWriter writer2 = new StringWriter();
writer2.write("Test");
String resultString = writer2.toString();
```

```
package java.io;

public class BufferedReader extends Reader {

    public BufferedReader(Reader in) {
        // ...
    }

    public String readLine() throws IOException {
        // ...
    }

    // ...
}
```

```
try (BufferedReader reader =
        new BufferedReader(
             new InputStreamReader(
                    new FileInputStream("in.txt"),
                    StandardCharsets.UTF_8))) {

    String line;
    while ((line = reader.readLine()) != null) {
        // process line
    }

}
```

```java
try (BufferedReader reader = Files.newBufferedReader(
        Paths.get("in.txt"), StandardCharsets.UTF_8)) {

    String line;
    while ((line = reader.readLine()) != null) {
        // process line
    }
}
```

```java
List<String> lines = Files.readAllLines(
        Paths.get("in.txt"), StandardCharsets.UTF_8);

for (String line : lines) {
    // process line
}
```

```
try (BufferedWriter writer = Files.newBufferedWriter(
        Paths.get("out.txt"), StandardCharsets.UTF_8)) {

    writer.write("Hello");
    writer.newLine();
}
```

```
List<String> lines = Arrays.asList("Hello", "world");
Files.write(Paths.get("out.txt"), lines,
        StandardCharsets.UTF_8);
```

```
package java.io;

public class PrintWriter extends Writer {

    public PrintWriter (Writer out) {
        // ...
    }

    public void print(int i) {
        // ...
    }

    public void println(Object obj) {
        // ...
    }

    public PrintWriter printf(String format, Object ... args) {
        // ...
    }

    public boolean checkError() {
        // ...
    }

    // ...
}
```

```
package java.io;

public class PrintStream extends FilterOutputStream
    implements Appendable, Closeable {

    public PrintStream(OutputStream out) {
        // ...
    }

    public void print(int i) {
        // ...
    }

    public void println(Object obj) {
        // ...
    }

    public PrintWriter printf(String format, Object ... args) {
        // ...
    }

    public boolean checkError() {
        // ...
    }

    // ...
}
```

```
// java.io.StreamTokenizer
StreamTokenizer streamTokenizer =
    new StreamTokenizer(
        new StringReader("Hello world"));


// java.util.StringTokenizer
StringTokenizer stringTokenizer =
    new StringTokenizer("Hello world");
```

```
Reader reader = new StringReader(
        "abc|true|1,1e3|-42");

Scanner scanner = new Scanner(reader)
        .useDelimiter("\\|")
        .useLocale(Locale.forLanguageTag("ru"));

String token = scanner.next();
boolean bool = scanner.nextBoolean();
double dbl = scanner.nextDouble();
int integer = scanner.nextInt();
```

```java
package java.lang;

public final class System {

    public static final InputStream in = null;

    public static final PrintStream out = null;

    public static final PrintStream err = null;

    // ...
}
```