

Специализация: тонкости работы





Специализация: тонкости работы



На прошлом уроке

- 📌 Перечисления;
- 📌 Внутренние классы;
- 📌 Вложенные классы;
- 📌 Внутренние статические классы;
- 📌 Механизм исключительных ситуаций;
- 📌 Введение в многопоточность;





Что будет на уроке сегодня

- 📌 Загрузчики
- 📌 Файловые системы
 - Windows
 - Linux
- 📌 Файловая система для программы
- 📌 Файлы для Java 1.7+
- 📌 java.io, java.nio
- 📌 String
- 📌 String pool





Файловая система



Файловая система

Файловая система - один из ключевых компонентов всех операционных систем. В ее обязанности входит структуризация, чтение, хранение, запись файлов. Выделяют Windows-подход и Linux-подход.

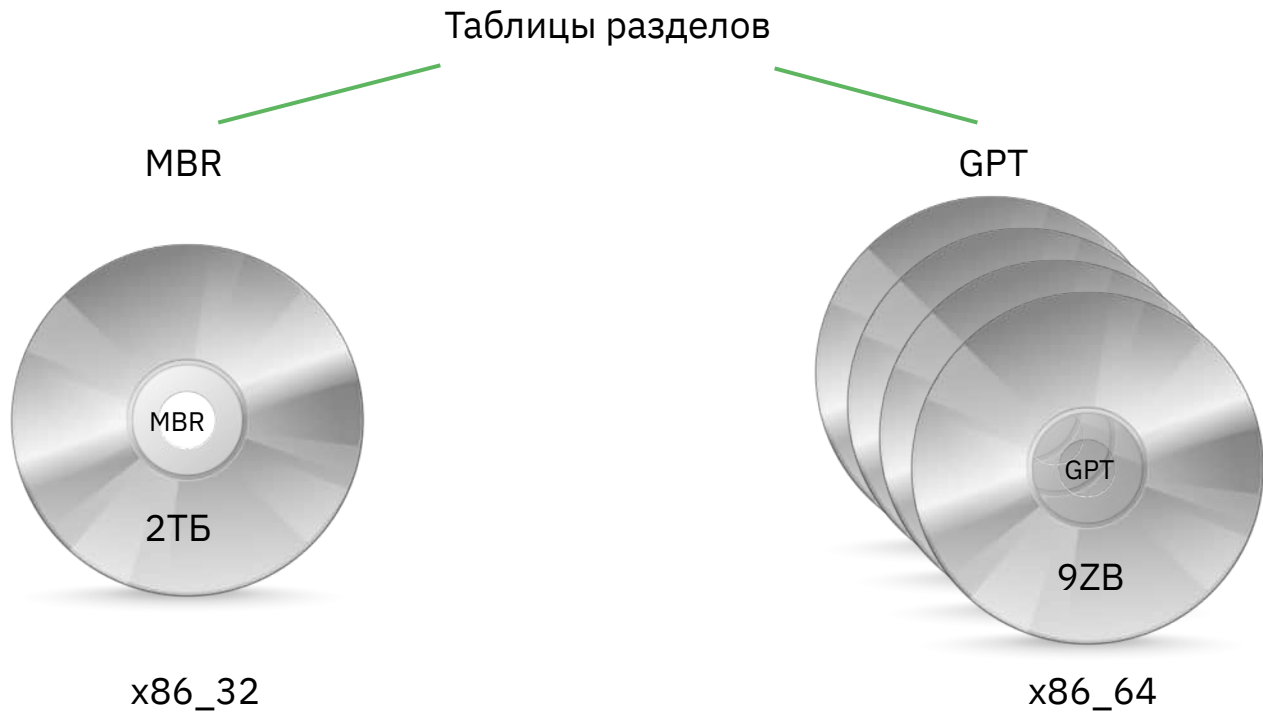


Выбор файловой системы

Linux на этапе установки предоставляет выбор из огромного числа файловых систем. Также возможно самостоятельно выбрать разделы жёсткого диска и подобрать файловую систему строго под свои нужды.



Разделы GPT и MBR






Main Boot Record

Смещение	Длина, байт	Описание	
0000h	446	Код загрузчика	
01BEh	16	Раздел 1	Таблица разделов
01CEh	16	Раздел 2	
01DEh	16	Раздел 3	
01EEh	16	Раздел 4	
01FEh	2	Сигнатура (55h AAh)	





Состав MBR

-  Основной загрузчик;
-  Таблица разделов диска;
-  Конечная подпись.

Hexspeak: 0xDEADBEEF, 0xA11F0DAD, 0xDEADFA11



Pros and Cons



Плюсы:

- Совместимость со всеми версиями Windows, включая Windows 7 и более ранние версии;
- Требуется для обеспечения совместимости со старым 32-разрядным оборудованием;
- Использует 32-битные значения, поэтому имеет меньшие накладные расходы, чем GPT.



Минусы

- Максимальная емкость раздела составляет 2 ТБ;
- Ограничено 4 основными разделами или 3 основными разделами и 1 расширенным разделом;
- Не устойчив к повреждению MBR;
- Не имеет встроенного исправления ошибок для защиты данных, поскольку использует BIOS.

Ответьте на вопрос сообщением в чат

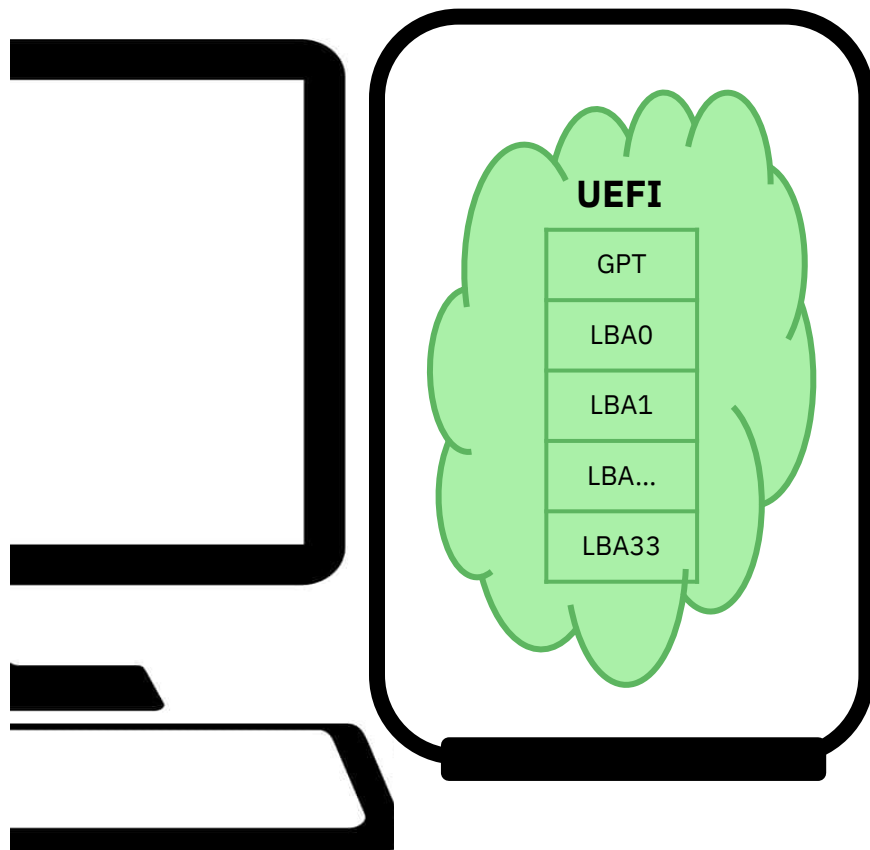
Вопрос:

1. MBR - это
 - a. main boot record;
 - b. master BIOS recovery;
 - c. minimizing binary risks.





Новый “железный” интерфейс



GUID Partition Table (GPT) United
Extensible Firmware Interface (UEFI)





Структура GPT



Защитная MBR;






Структура GPT

-  Защитная MBR;
-  Основной тег GPT;





Структура GPT

-  Защитная MBR;
-  Основной тег GPT;
-  Записи разделов;





Структура GPT

- 📌 Защитная MBR;
- 📌 Основной тег GPT;
- 📌 Записи разделов;
- 📌 Дополнительный GPT.





Структура GPT

- 📌 Защитная MBR;
 - 📌 Основной тег GPT;
 - 📌 Записи разделов;
 - 📌 Дополнительный GPT.
-
- 📌 Максимальная емкость раздела 9.4ZB;
 - 📌 128 первичных разделов;
 - 📌 Устойчив к повреждению первичного раздела, так как имеет вторичный GPT;
 - 📌 Возможность использовать функции UEFI.





MBR vs. GPT




Требования к прошивке: BIOS vs UEFI





MBR vs. GPT

 Требования к прошивке: BIOS vs UEFI

 Поддержка Windows: x32 vs x64





MBR vs. GPT

- 📌 Требования к прошивке: BIOS vs UEFI
- 📌 Поддержка Windows: x32 vs x64
- 📌 Максимальная ёмкость раздела: $(2^{32} - 1) \times 512$ байт = 2,19 ТБ vs $(2^{64} - 1) \times 512$ байт = 9,44 ZB





MBR vs. GPT

- 📌 Требования к прошивке: BIOS vs UEFI
- 📌 Поддержка Windows: x32 vs x64
- 📌 Максимальная ёмкость раздела: $(2^{32} - 1) \times 512$ байт = 2,19 ТБ vs $(2^{64} - 1) \times 512$ байт = 9,44 ZB
- 📌 Количество разделов: 4 (или 3 + 1) vs 128





MBR vs. GPT

- 📌 Требования к прошивке: BIOS vs UEFI
- 📌 Поддержка Windows: x32 vs x64
- 📌 Максимальная ёмкость раздела: $(2^{32} - 1) \times 512$ байт = 2,19 ТБ vs $(2^{64} - 1) \times 512$ байт = 9,44 ZB
- 📌 Количество разделов: 4 (или 3 + 1) vs 128
- 📌 Скорость загрузки: BIOS vs UEFI





MBR vs. GPT

- 📌 Требования к прошивке: BIOS vs UEFI
- 📌 Поддержка Windows: x32 vs x64
- 📌 Максимальная ёмкость раздела: $(2^{32} - 1) \times 512$ байт = 2,19 ТБ vs $(2^{64} - 1) \times 512$ байт = 9,44 ZB
- 📌 Количество разделов: 4 (или 3 + 1) vs 128
- 📌 Скорость загрузки: BIOS vs UEFI
- 📌 Безопасность данных





Ответьте на вопрос сообщением в чат

Вопрос:

1. Что такое GPT?
 - a. General partition trace;
 - b. GUID partition table;
 - c. Greater pass timing





Файловые системы в ОС



Файловые системы










Файловые системы - это архитектура хранения информации, размещенной на жестком диске и в оперативной памяти



Как обстоят дела в Linux

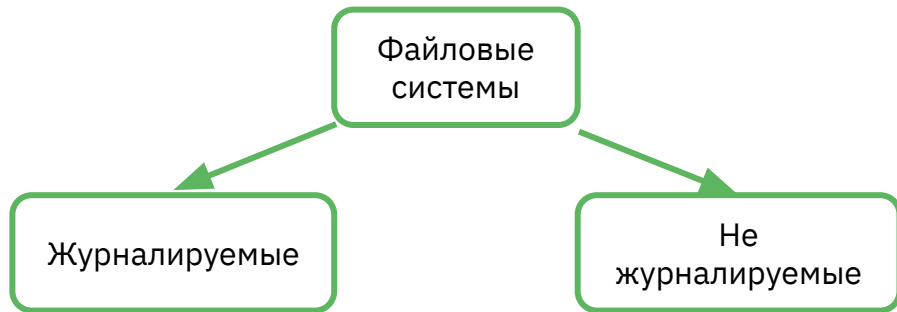


На что влияет файловая система?

-  Скорость обработки данных;
-  Сбережение файлов;
-  Оперативность записи;
-  Допустимый размер блока;
-  Возможность хранения информации в оперативной памяти;
-  Способы корректировки пользователями ядра
-  и пр.



Категории файловых систем



- 📌 Ext (extended) FS;
- 📌 Ext2, 3, 4;
- 📌 JFS, ReiserFS, XFS, Btrfs, F2FS;
- 📌 EncFS, Aufs, NFS;
- 📌 Tmpfs, Procfs, Sysfs;



Файлы в Linux



Regular File





Файлы в Linux



Regular File



Device File





Файлы в Linux



Regular File



Device File







Soft Link










Файлы в Linux

-  Regular File
-  Device File
-  Soft Link
-  Directories





Файлы в Linux

-  Regular File
-  Device File
-  Soft Link
-  Directories
-  Block devices and sockets











Как обстоят дела в Windows



Файловая система FAT(16)

FAT(16) File Allocation Table (с англ. таблица размещения файлов)

-  MS-DOS 3.0,
-  Windows 3.x,
-  Windows 95,
-  Windows 98,
-  Windows NT/2000
-  ...



Файловая система FAT32

- 📌 Возможность перемещения корневого каталога
- 📌 Возможность хранения резервных копий





Файловая система NTFS

NTFS — от англ. New Technology File System, файловая система новой технологии



Права доступа



Шифрование данных



Дисковые квоты



Хранение разреженных файлов



Журналирование



FAT



Файловая система	Параметры	
	Размеры тома	Максимальный размер файла
FAT	От 1.44 МБ до 4 ГБ	2 ГБ
FAT32	Теоретически возможен размер тома от 512МБ до 2 Тбайт. Сжатие не поддерживается	4 ГБ
NTFS	Минимальный рекомендуемый размер составляет 1.44 МБ, а максимальный 2Тбайт. Поддержка сжатия на уровне файловой системы для файлов, каталогов и томов.	Максимальный размер ограничен лишь размером тома (Теоретически 2^{64} байт минус 1 килобайт. практически - 2^{44} байт минус 64 килобайта).

Ответьте на вопросы сообщением в чат

Вопросы:

1. Что такое файловая система?
2. Возможно ли использовать разные файловые системы в рамках одной ОС?





Файловая система и представление данных



Класс файл

```
1 File file = new File("file.txt");
```

✓ 0.9s

Java



Директории

```
1 File folder = new File(".");  
2 for (File file : folder.listFiles()) {  
3     System.out.println(file.getName());  
4 }
```

✓ 0.2s

Java

.DS_Store

Icon

bin

out

Dockerfile



Получение информации

```
1 System.out.println("Is it a folder - " + folder.isDirectory());
2 System.out.println("Is it a file - " + folder.isFile());
3 File file = new File("./Dockerfile");
4 System.out.println("Length file - " + file.length());
5 System.out.println("Absolute path - " + file.getAbsolutePath());
6 System.out.println("Total space on disk - " + folder.getTotalSpace());
7 System.out.println("File deleted - " + file.delete());
8 System.out.println("File exists - " + file.exists());
9 System.out.println("Free space on disk - " + folder.getFreeSpace());
```

✓ 0.4s

Java

```
Is it a folder - true
Is it a file - false
Length file - 270
Absolute path - /Users/ivan-igorevich/GDrive/work-GB/developer-java-
drafts/sources-draft/./Dockerfile
Total space on disk - 184999997440
File deleted - true
File exists - false
Free space on disk - 119489359872
```





Классы Path, Paths, Files, FileSystem в Java 7+



Файловая система

```
import java.nio.file.FileSystem;  
import java.nio.file.FileSystems;  
import java.nio.file.Path;
```

✓ 0.4s

```
try {  
    FileSystem filesystem = FileSystems.getDefault();  
    for (Path rootdir : filesystem.getRootDirectories()) {  
        System.out.println(rootdir.toString());  
    }  
} catch (Exception e) {  
    e.printStackTrace();  
}
```

✓ 0.6s

C:\
D:\
Z:\



Путь

```
File file = new File("Main.java");  
Path filePath = Paths.get(file.getName());  
System.out.println(filePath);
```

✓ 0.9s

Main.java

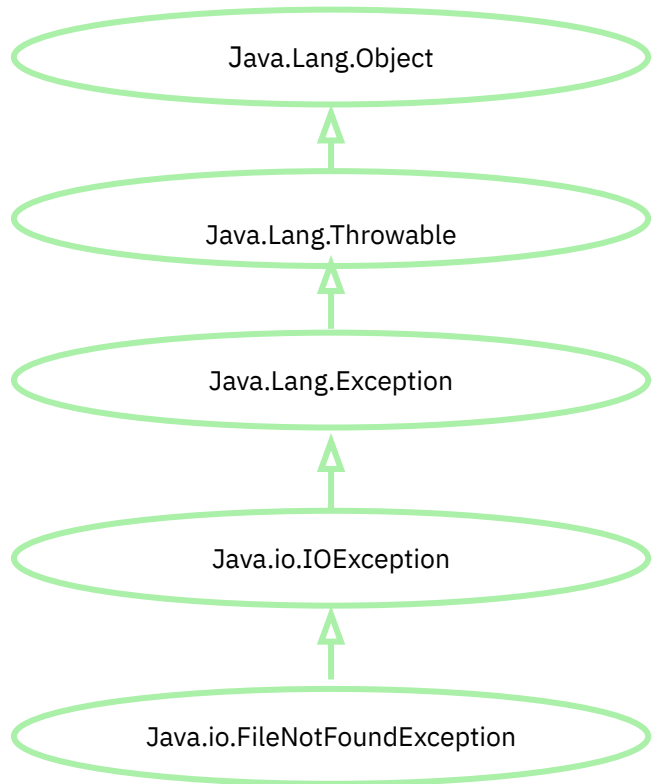
URI - Uniform Resource Identifier (унифицированный идентификатор ресурса)

URL - Uniform Resource Locator (унифицированный определитель местонахождения ресурса)

URN - Uniform Resource Name (унифицированное имя ресурса)



Когда программа не может найти файл



- Файл с указанным именем не существует;
- Файл с указанным именем существует , но недоступен по какой-либо причине.



Действия с путями

```
1 Path filePath = Paths.get("pics/logo.png");
2
3 Path fileName = filePath.getFileName();
4 System.out.println("Filename: " + fileName);
5 Path parent = filePath.getParent();
6 System.out.println("Parent directory: " + parent);
7
8 boolean endWithTxt = filePath.endsWith("logo.png");
9 System.out.println("Ends with filepath: " + endWithTxt);
10 endWithTxt = filePath.endsWith("png");
11 System.out.println("Ends with string: " + endWithTxt);
12
13 boolean startsWithPics = filePath.startsWith("pics");
14 System.out.println("Starts with filepath: " + startsWithPics);
```

✓ 0.1s

Filename: logo.png

Parent directory: pics

Ends with filepath: true

Ends with string: false

Starts with filepath: true



Очистка путей

```
System.out.println(filePath.isAbsolute());  
Path pathFirst = Paths.get("./pics/logo.png");  
System.out.println(pathFirst.normalize());  
  
Path pathSecond = Paths.get("../sources-draft/../../pics/logo.png");  
System.out.println(pathSecond.normalize());
```

✓ 0.1s

```
false  
pics\logo.png  
pics\logo.png
```





Манипулирование файлом

```
import static java.nio.file.StandardCopyOption.REPLACE_EXISTING;

Path file = Files.createFile(Paths.get("../pics/file.txt"));
System.out.print("Was the file captured successfully in pics directory? ");
System.out.println(Files.exists(Paths.get("../pics/file.txt")));

Path testDirectory = Files.createDirectory(Paths.get("../testing"));
System.out.print("Was the test directory created successfully? ");
System.out.println(Files.exists(Paths.get("../testing")));

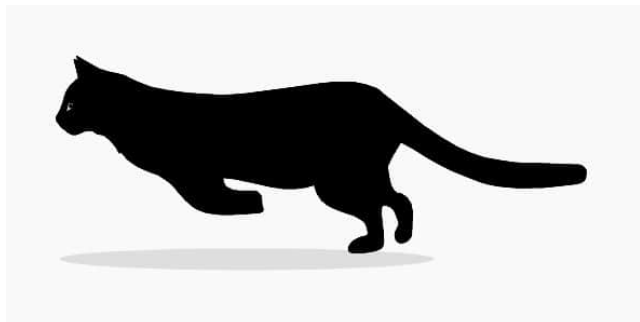
file = Files.move(file, Paths.get("../testing/file.txt"), REPLACE_EXISTING);
System.out.print("Is our file still in the pics directory? ");
System.out.println(Files.exists(Paths.get("../pics/file.txt")));
System.out.print("Has our file been moved to testDirectory? ");
System.out.println(Files.exists(Paths.get("../testing/file.txt")));

Path copyFile = Files.copy(file, Paths.get("../pics/file.txt"), REPLACE_EXISTING);
System.out.print("Has our file been copied to pics directory? ");
System.out.println(Files.exists(Paths.get("../pics/file.txt")));

Files.delete(file);
System.out.print("Does the file exist in test directory? ");
System.out.println(Files.exists(Paths.get("../testing/file.txt")));
System.out.print("Does the test directory exist? ");
System.out.println(Files.exists(Paths.get("../testing")));
```

✓ 0.2s

```
Was the file captured successfully in pics directory? true
Was the test directory created successfully? true
Is our file still in the pics directory? false
Has our file been moved to testDirectory? true
Has our file been copied to pics directory? true
Does the file exist in test directory? false
Does the test directory exist? true
```





Работа с содержимым

```
1 List<String> lines = Arrays.asList(  
2     "The cat wants to play with you",  
3     "But you don't want to play with it");  
4  
5 Path file = Files.createFile(Paths.get("cat.txt"));  
6  
7 if(Files.exists(file)) {  
8     Files.write(file, lines, StandardCharsets.UTF_8);  
9     lines = Files.readAllLines(  
10         Paths.get("cat.txt"), StandardCharsets.UTF_8);  
11  
12     for (String s : lines) {  
13         System.out.println(s);  
14     }  
15 }  
16
```

✓ 0.3s

Java

```
The cat wants to play with you  
But you don't want to play with it
```

```
1 Files.delete(file);  
✓ 0.2s
```



Ответьте на вопросы сообщением в чат

Вопросы:

1. Какое исключение выбрасывает программа, если не может открыть файл?
2. Ссылка на местонахождение - это:
 - a. URI;
 - b. URL;
 - c. URN.





java.io



Абстрактный ввод-вывод

internet



Устройства ввода

Программа

internet



Устройства вывода



Абстрактный ввод-вывод

internet



Устройства ввода

ПОТОК
ВВОДА

П
р
о
г
р
а
м
м
а

ПОТОК
ВЫВОДА

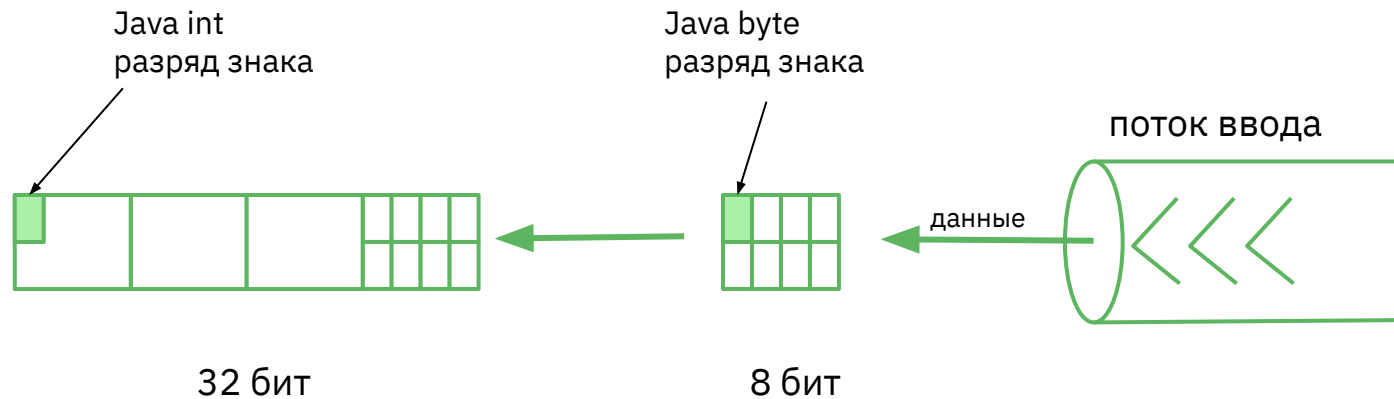
internet



Устройства вывода



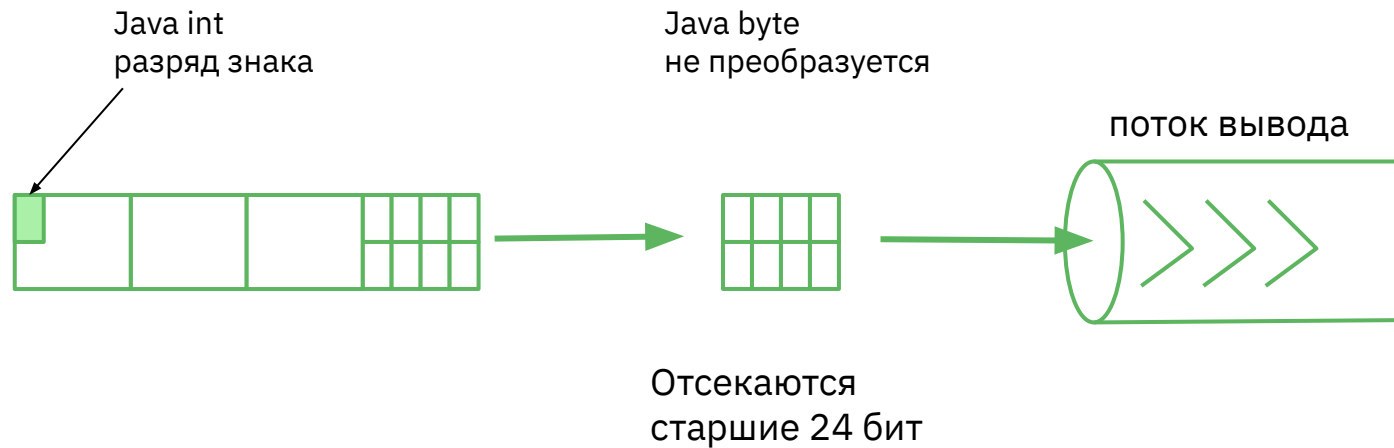
Классы InputStream и OutputStream



InputStream – это базовый абстрактный класс для потоков ввода, т.е. чтения.



Классы InputStream и OutputStream



OutputStream – это базовый абстрактный класс для потоков вывода, т.е. записи.



Классы `ByteArrayInputStream` и `ByteArrayOutputStream`

```
1  ByteArrayOutputStream out = new ByteArrayOutputStream();
2
3  out.write(1);
4  out.write(-1);
5  out.write(0);
6
7  ByteArrayInputStream in = new ByteArrayInputStream(out.toByteArray());
8
9  int value = in.read();
10 System.out.println("First element is - " + value);
11
12 value = in.read();
13 System.out.println("Second element is - " + value +
14 |           |           | ". If (byte)value - " + (byte)value);
15
16 value = in.read();
17 System.out.println("Third element is - " + value);
```



Файловый потоковый ввод-вывод

```
1 byte[] bytesToWrite = {0, 10, 12, 14, 55, 13, 23};
2 byte[] bytesToRead = new byte[10];
3 File file = new File("bytes.txt");
4
5 try {
6     System.out.println("Begin");
7     FileOutputStream outFile = new FileOutputStream(file);
8     outFile.write(bytesToWrite); outFile.close();
9     System.out.println("Bytes written");
10
11     FileInputStream inFile = new FileInputStream(file);
12     int bytesAvailable = inFile.available();
13     System.out.println("Ready to read " + bytesAvailable + " bytes");
14
15     int count = inFile.read(bytesToRead, 0, bytesAvailable);
16     for (int i = 0; i < count; i++)
17         System.out.print(" " + bytesToRead[i]);
18
19     System.out.println(); inFile.close();
20     System.out.println("Input stream closed");
21
22 } catch (FileNotFoundException e) {
23     System.out.println("Unable to write data to file - " + file.getName());
24 } catch (IOException e) {
25     System.out.println("Error input/output: " + e.toString());
26 }
27
```

✓ 0.5s

Java

26 }

27

✓ 0.5s

Begin

Bytes written

Ready to read 7 bytes

0 10 12 14 55 13 23

Input stream closed



Другие потоковые классы



PipedInputStream и Piped OutputStream





Другие потоковые классы

- 📌 PipedInputStream и Piped OutputStream
- 📌 StringBuilderInputStream





Другие потоковые классы

- 📌 PipedInputStream и Piped OutputStream
- 📌 StringBuilderInputStream
- 📌 SequenceInputStream





Другие потоковые классы

- 📌 PipedInputStream и Piped OutputStream
- 📌 StringBuilderInputStream
- 📌 SequenceInputStream
- 📌 FilterInputStream, FilterOutputStream и их наследники





Другие потоковые классы

- 📌 PipedInputStream и Piped OutputStream
- 📌 StringBuilderInputStream
- 📌 SequenceInputStream
- 📌 FilterInputStream, FilterOutputStream и их наследники
- 📌 LineNumberInputStream, LineNumberReader





Другие потоковые классы

- 📌 PipedInputStream и Piped OutputStream
- 📌 StringBuilderInputStream
- 📌 SequenceInputStream
- 📌 FilterInputStream, FilterOutputStream и их наследники
- 📌 LineNumberInputStream, LineNumberReader
- 📌 PushBackInputStream



Другие потоковые классы

- 📌 PipedInputStream и Piped OutputStream
- 📌 StringBuilderInputStream
- 📌 SequenceInputStream
- 📌 FilterInputStream, FilterOutputStream и их наследники
- 📌 LineNumberInputStream, LineNumberReader
- 📌 PushBackInputStream
- 📌 PrintStream, PrintWriter





BufferedInputStream. Сравнение

```
1  try {
2      long timeStart = System.currentTimeMillis();
3      outputStream = new BufferedOutputStream(new FileOutputStream(fileName));
4      for (int i = 1000000; --i >= 0;) { outputStream.write(i); }
5
6      long time = System.currentTimeMillis() - timeStart;
7      System.out.println("Writing time: " + time + " millisec");
8      outputStream.close();
9  } catch (IOException e) {
10     System.out.println("IOException: " + e.toString());
11     e.printStackTrace();
12 }
13
```

✓ 0.3s

Java

Writing time: 24 millisec



BufferedInputStream. Сравнение

```
1  try {
2      long timeStart = System.currentTimeMillis();
3      InputStream inStream = new FileInputStream(fileName);
4      while (inStream.read() != -1) { }
5
6      long time = System.currentTimeMillis() - timeStart;
7      inStream.close();
8      System.out.println("Direct read time: " + (time) + " millisec");
9  } catch (IOException e) {
10     System.out.println("IOException: " + e.toString());
11     e.printStackTrace();
12 }
13
```

✓ 1.4s

Java

Direct read time: 1220 millisec



BufferedInputStream. Сравнение

```
1  try {
2      long timeStart = System.currentTimeMillis();
3      inStream = new BufferedInputStream(new FileInputStream(fileName));
4      while (inStream.read() != -1) { }
5
6      long time = System.currentTimeMillis() - timeStart;
7      inStream.close();
8      System.out.println("Buffered read time: " + (time) + " millisec");
9  } catch (IOException e) {
10     System.out.println("IOException: " + e.toString());
11     e.printStackTrace();
12 }
13
```

✓ 0.2s

Java

Buffered read time: 44 millisec



Усложнение данных. DataOutputStream



```
6  
7  ByteArrayOutputStream out = new ByteArrayOutputStream();
```

✓ 0.5s

Java

```
1  try {  
2      DataOutputStream outData = new DataOutputStream(out);  
3  
4      outData.writeByte(128);  
5      outData.writeInt(128);  
6      outData.writeLong(128);  
7      outData.writeDouble(128);  
8      outData.close();  
9  } catch (Exception e) {  
10     System.out.println("Impossible IOException occurs: " + e.toString());  
11     e.printStackTrace();  
12 }  
13
```

✓ 0.1s

Java



Усложнение данных. DataInputStream

```
1  try {
2      byte[] bytes = out.toByteArray();
3      InputStream in = new ByteArrayInputStream(bytes);
4      DataInputStream inData = new DataInputStream(in);
5
6      System.out.println("Reading in the correct sequence: ");
7      System.out.println("readByte: " + inData.readByte());
8      System.out.println("readInt: " + inData.readInt());
9      System.out.println("readLong: " + inData.readLong());
10     System.out.println("readDouble: " + inData.readDouble());
11     inData.close();
12 } catch (Exception e) {
13     System.out.println("Impossible IOException occurs: " + e.toString());
14     e.printStackTrace();
15 }
16
```

✓ 0.2s

Java

Reading in the correct sequence:

readByte: -128

readInt: 128

readLong: 128

readDouble: 128.0



Усложнение данных. Ломаем DataInputStream

```
1  try {
2      byte[] bytes = out.toByteArray();
3      InputStream in = new ByteArrayInputStream(bytes);
4      DataInputStream inData = new DataInputStream(in);
5
6      System.out.println("Reading in a modified sequence:");
7      System.out.println("readInt: " + inData.readInt());
8      System.out.println("readDouble: " + inData.readDouble());
9      System.out.println("readLong: " + inData.readLong());
10
11     inData.close();
12 } catch (Exception e) {
13     System.out.println("Impossible IOException occurs: " + e.toString());
14     e.printStackTrace();
15 }
16
```

✓ 0.2s

Java

Reading in a modified sequence:

readInt: -2147483648

readDouble: -0.0

readLong: -9205252085229027328



Ещё сложнее: `ObjectInputStream`, `ObjectOutputStream`





Байтовые и символьные потоки

Байтовый поток	Символьный поток
InputStream	Reader
OutputStream	Writer
ByteArrayInputStream	CharArrayReader
ByteArrayOutputStream	CharArrayWriter
Нет аналога	InputStreamReader
Нет аналога	OutputStreamWriter
FileInputStream	FileReader

Байтовый поток	Символьный поток
FileOutputStream	FileWriter
BufferedInputStream	BufferedReader
BufferedOutputStream	BufferedWriter
PrintStream	PrintWriter
DataInputStream	Нет аналога
DataOutputStream	Нет аналога

Классы-мосты InputStreamReader и OutputStreamWriter

Ответьте на вопросы сообщением в чат

Вопросы:

1. Возможно ли чтение совершенно случайного байта данных из объекта `BufferedInputStream`?
2. Возможно ли чтение совершенно случайного байта данных из потока, к которому подключен объект `BufferedInputStream`?

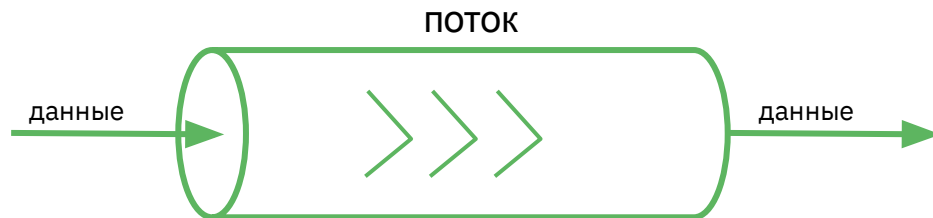




java.nio, java.nio2

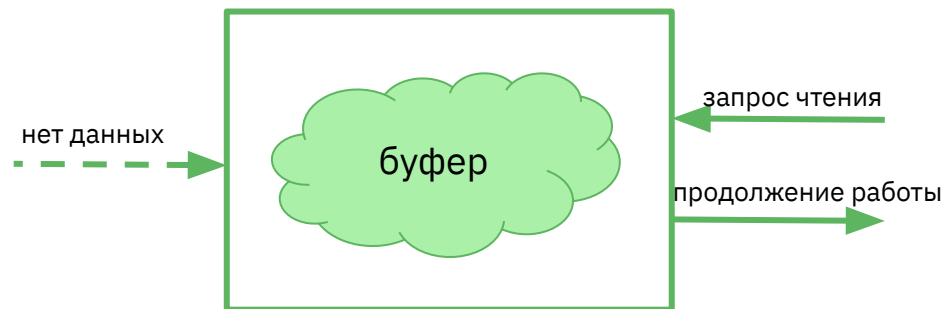
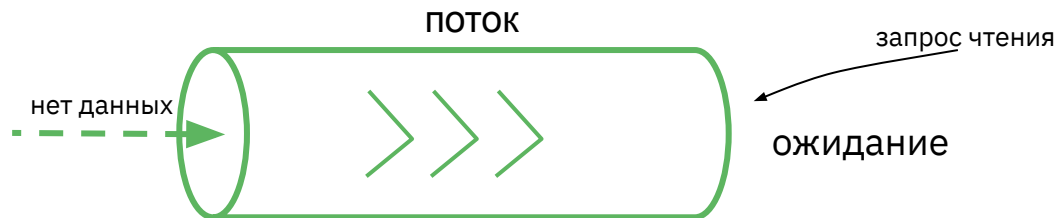


io vs. nio: потоки и буферы



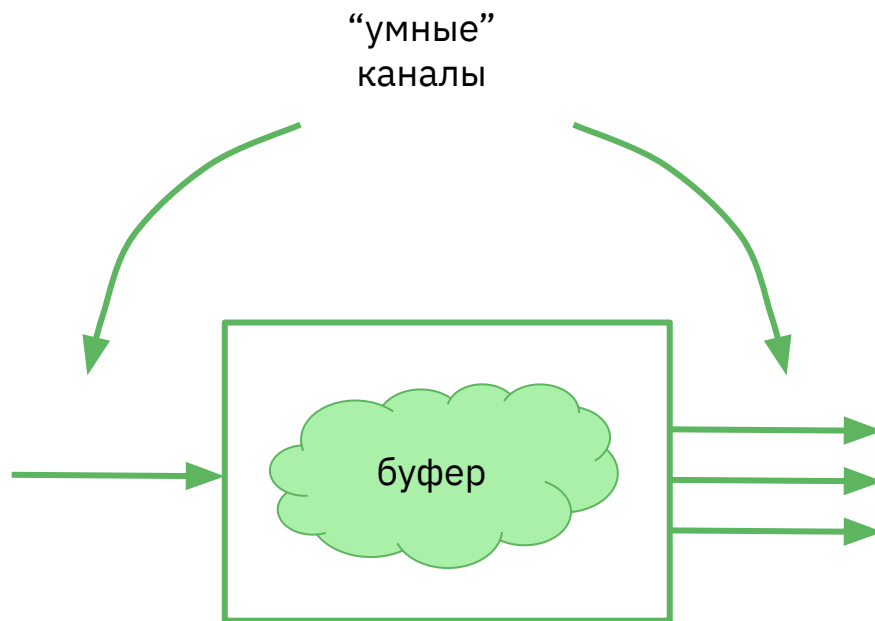


Ожидание чтения





Каналы





Канал и буфер

```
1 try (RandomAccessFile catFile = new RandomAccessFile("cat.txt", "rw")) {
2     FileChannel inChannel = catFile.getChannel();
3     ByteBuffer buf = ByteBuffer.allocate(100);
4     int bytesRead = inChannel.read(buf);
5
6     while (bytesRead != -1) {
7         System.out.println("Read " + bytesRead + " bytes");
8         // Set read mode
9         buf.flip();
10        while (buf.hasRemaining()) {
11            System.out.print((char) buf.get());
12        }
13
14        buf.clear();
15        bytesRead = inChannel.read(buf);
16    }
17 } catch (IOException e) { e.printStackTrace(); }
```

✓ 0.2s

Java


```
Read 60 bytes
Hello from file-cat
Hello from file-cat
Hello from file-cat
```




String



Строка - это не только символы

 immutable

 final

```
private final char value[];  
private final byte[] value;
```





Экземпляры класса String

```
1  import java.nio.charset.StandardCharsets;
2
3  String s1 = "Java";
4  String s2 = new String("Home");
5  String s3 = new String(new char[] { 'A', 'B', 'C' });
6  String s4 = new String(s3);
7  String s5 = new String(new byte[] { 65, 66, 67 });
8  String s6 = new String(new byte[] { 0, 65, 0, 66 }, StandardCharsets.UTF_16);
9
```

✓ 2.7s

Java

ASCII - American standard code for information interchange



Конкатенация

```
1 String a1 = "Hello ";
2 String a2 = "World";
3 String a3 = a1 + a2;
4
5 System.out.println(a3);
6
7 String b1 = "Number 10: ";
8 int b2 = 10;
9 String b3 = b1 + b2;
10
11 System.out.println(b3);
12
13 String c1 = "Home";
14 String c2 = "[" + c1 + "] = " + 1;
15
16 System.out.println(c2);
17
```

✓ 1.5s

Java

Hello World

Number 10: 10

[Home] = 1



Проблема конкатенации

```
1 String str0 = "Fifty five is " + 50 + 5;  
2 System.out.println(str0);  
3 String str1 = 50 + 5 + " = Fifty five";  
4 System.out.println(str1);  
5
```

✓ 0.8s








Java

Fifty five is 505

55 = Fifty five



Часто используемые методы строки

-  `int length();`
-  `char charAt(int pos);`
-  `char[] toCharArray();`
-  `boolean equals(Object obj);`
-  `boolean equalsIgnoreCase(Object obj);`
-  `String concat(String obj);`
-  `String toLowerCase(); String toUpperCase();`



StringBuilder

```
1 String s = "Example";
2 long timeStart = System.nanoTime();
3 for (int i = 0; i < 100_000; ++i) { s = s + i; }
4 double deltaTime = (System.nanoTime() - timeStart) * 0.000000001;
5 System.out.println("Delta time (sec): " + deltaTime);
```

✓ 7.9s

Java

Delta time (sec): 5.96577258



```
1 StringBuilder sb = new StringBuilder("Example");
2 long timeStart = System.nanoTime();
3 for (int i = 0; i < 100_000; ++i) { sb = sb.append(i); }
4 double deltaTime = (System.nanoTime() - timeStart) * 0.000000001;
5 System.out.println("Delta time (sec): " + deltaTime);
```

✓ 0.4s

Java

Delta time (sec): 0.17202656200000002

Ответьте на вопросы сообщением в чат

Вопросы:

1. String - это:
 - a. объект;
 - b. примитив;
 - c. класс.
2. Строки в языке Java это:
 - a. классы;
 - b. массивы;
 - c. объекты.

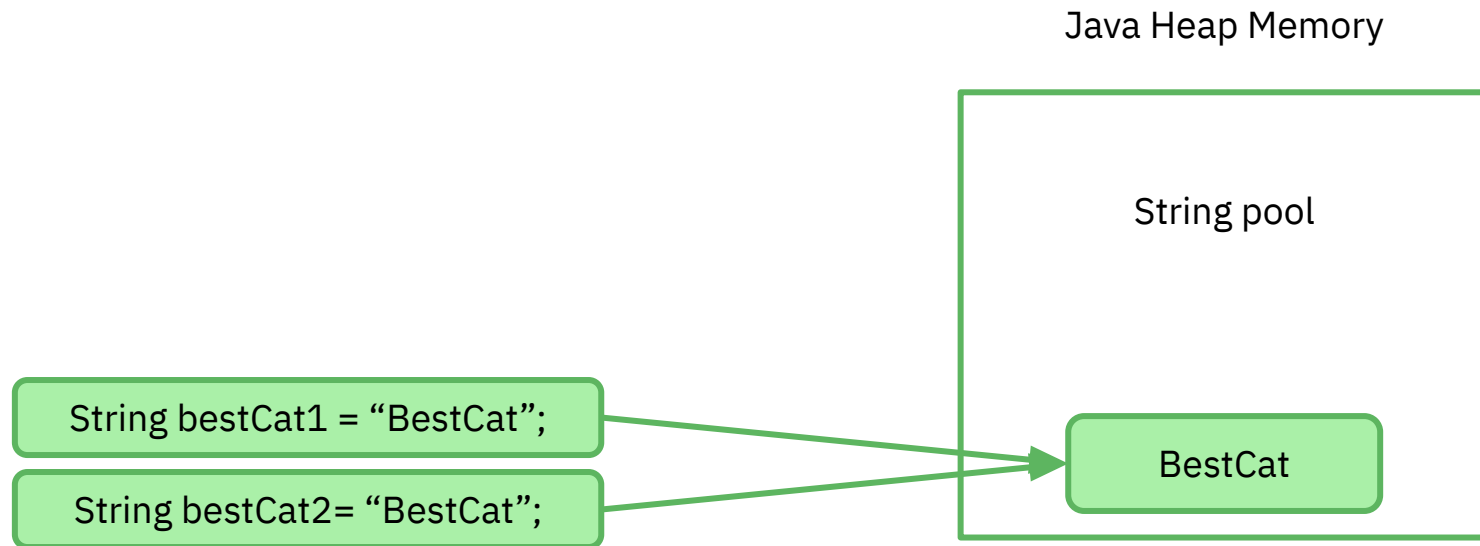




String pool



Интернирование





String pool и конкатенация строк

```
1 String cat0 = "BestCat";
2 String cat1 = "BestCat";
3 String cat2 = "Best" + "Cat";
4 String cat30 = "Best";
5 String cat3 = cat30 + "Cat";
6
7 System.out.println("cat0 equal to cat1? " + (cat0 == cat1));
8 System.out.println("cat0 equal to cat2? " + (cat0 == cat2));
9 System.out.println("cat0 equal to cat3? " + (cat0 == cat3));
10
```

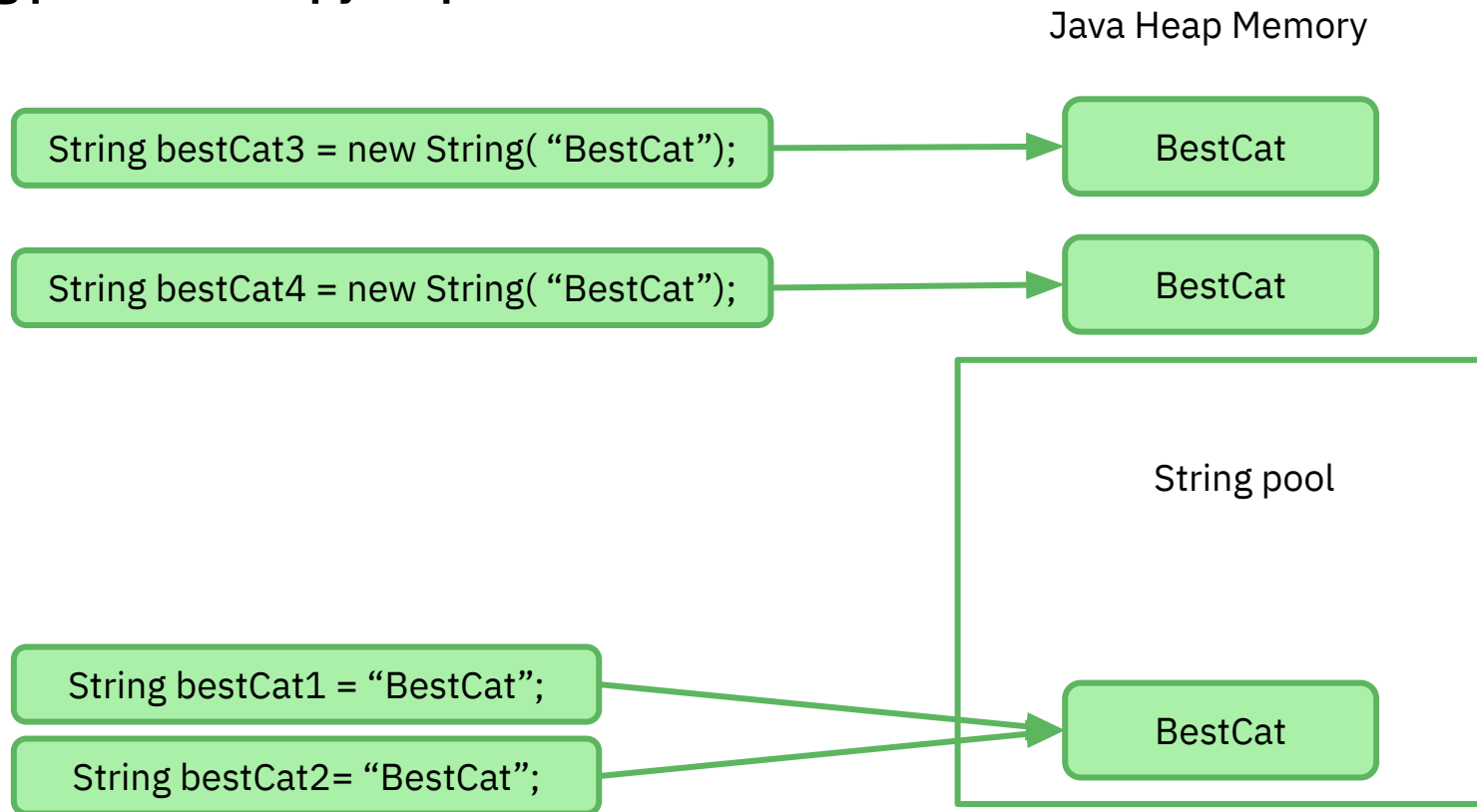
✓ 0.8s

Java

```
cat0 equal to cat1? true
cat0 equal to cat2? true
cat0 equal to cat3? false
```



String pool и конструктор








На этом уроке

- 📌 Загрузчики
- 📌 Файловые системы
 - Windows
 - Linux
- 📌 Файловая система для программы
- 📌 Файлы для Java 1.7+
- 📌 java.io, java.nio
- 📌 String
- 📌 String pool





Практическое задание

-  Создать три текстовых файла. Для упрощения внутри файлов следует писать текст только латинскими буквами.
-  Написать метод, осуществляющий конкатенацию файлов;
-  Написать метод поиска слова внутри файла.





Учитесь так, словно вы постоянно ощущаете нехватку своих знаний, и так, словно вы постоянно боитесь растерять свои знания

Конфуций