Recap: Arrays

- Data structure to store multiple values
- Syntax

```java
// create a new array with 4 components
int[] arr = new int[4];

// access 2nd component
System.out.println(arr[1]);
arr[1] = 7;
```

EXPR

- Properties
  - Length is fixed
  - All components are of the same type (e.g., `int`)
  - A reference type

Section 1

Exercise 6

UNIVERSITÄT
ZU KÖLN

Session 7: Strings and Ascii Art 2.0

Softwaretechnologie: Java I

Nils Reiter
nils.reiter@uni-koeln.de

December 27, 2024

INSTITUT FÜR
DIGITAL HUMANITIES
UNIVERSITÄT ZU KÖLN

Section 2

Strings/Zeichenketten

Introduction

- Represents character sequences
- A reference type
- Internally: An array of `char`-values (mostly)

```
String s = "Hi there!"; // String literal with double quotes
```

new String ...

String Operations

- Concatenation (“Aneinanderhängen”)

```
String s1 = "Hi";
String s2 = "there";
String s = s1 + s2; // s now contains "Hithere"
```

  - `+` is the only regular math operator you can use with strings

String Operations

- Concatenation (“Aneinanderhängen”)

```
String s1 = "Hi";
String s2 = "there";
String s = s1 + s2; // s now contains "Hithere"
```

  - `+` is the only regular math operator you can use with strings
- Length: `s.length() //returns 7 (as an int)`
  - Note the round brackets
  - Gives us the length in characters, not in bytes

String Operations

- Concatenation (“Aneinanderhängen”)

```
String s1 = "Hi";
String s2 = "there";
String s = s1 + s2; // s now contains "Hithere"
```

  - `+` is the only regular math operator you can use with strings
- Length: `s.length() //returns 7 (as an int)`
  - Note the round brackets
  - Gives us the length in characters, not in bytes
- Convert case
  - `s2.toLowerCase(); //returns "hi"`
  - `s2.toUpperCase(); //returns "HI"`

Strings and Other Types

- All primitive types can be converted into a string
  - `System.out.println()` does this automatically, as we have seen
- Conversion done implicitly:

```java
int i = 2024;
String s = "Hallo";
System.out.println(s + i); // implicit conversion of i,
                           // then concatenation
```

Strings and Other Types

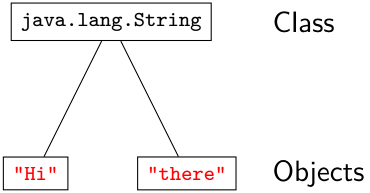
- All primitive types can be converted into a string
  - `System.out.println()` does this automatically, as we have seen
- Conversion done implicitly:

```java
int i = 2024;
String s = "Hallo";
System.out.println(s + i); // implicit conversion of i,
                           // then concatenation
```

- Explicit conversion
  - Many functions `String.valueOf(ARG)`
  - Take all primitive types as arguments

The class String

- `java.lang.String`: Our first class
- Classes and Objects:
  Object-oriented programming



“`Hi` is an instance of class `String`”

What can we do with Strings?

...and how do we find out?

java.lang.String

- Javadoc
  - `char charAt(int index);` – Zeichen an einer Position
  - `int compareTo(String anotherString)` – Vergleich
  - `String concat(String str)` – "Hi" + "There"
  - `boolean endsWith(String suffix)`
  - `boolean isEmpty()`
  - `String substring(int beginIndex, int endIndex)`
  - ...

What can we do with Strings?

...and how do we find out?

java.lang.String

- Javadoc
  - `char charAt(int index);`
  - `int compareTo(String anotherString)`
  - `String concat(String str)`
  - `boolean endsWith(String suffix)`
  - `boolean isEmpty()`
  - `String substring(int beginIndex, int endIndex)`
  - ...
- How to use them? `INSTANCE.METHOD(ARGUMENTS)`
  - Eclipse suggests possible methods/fields in a small window
  - Methods are associated with the specific instance before the `.`

Section 3

ASCII Art 2.0

ASCII Art 2.0

- So far: All functions print out lines of the image directly
- Next version: Should be possible to manipulate the image as a whole (e.g., invert it)
- To do
  - Change all functions such that they return a string instead of printing one
  - Write a function to invert the image

demo

AsciiArt

Section 4

Exercise