Exceptions
? wer löst stack trace einer Exception aus?
? wann wird stack trace ausgelöst?
Ausgabe: 3 Ways to print an exception
- **System.out.println(e)**; Java prints it out
- **System.out.println(exception)**, a stack trace is not printed. Just the name of the exception class and the message
- **exception.printStackTrace(),** a complete chain of the names of the methods called, along with the line numbers, is printed

3 Ways to print an exception
- Java prints it out System.out.println(e);
- Print just the message System.out.println(e.getMessage());
- Print where the stack trace comes from e.printStackTrace();

PASSED BY VALUE
- **Primitives** are always passed by value
- **Object "references"** are passed by value. So it looks like the object is passed by reference but actually it is the value of the reference that is passed

THIS
... can only be called in a constructor and that too as a first statement.

CONVERSION Primitives
A narrowing primitive conversion may be used if all of the following conditions are satisfied:

1. The expression is a compile time constant expression of type byte, char, short, or int.
2. The type of the variable is byte, short, or char.
3. The value of the expression (which is known at compile time, because it is a constant expression) is representable in the type of the variable.

FLOW
switch Compile-Time Konstante:
Deklarierung und Zuweisung in Einem.

STRING
Methods
- length()
- charAt()
- indexOf()
- substring()
- toLowerCase()
- toUpperCase()
- equals() / equalsIgnoreCase()

method checks whether two String objects contain exactly the same characters in the same order.

- startsWith() / endWith()
- contains()
- replace()
- trim()

STRINGS
Strings <> References
1. Literal strings within the same class in the same package represent references to the same String object.

2. Literal strings within different classes in the same package represent references to the same String object.

3. Literal strings within different classes in different packages likewise represent references to the same String object.

4. Strings computed by constant expressions are computed at compile time and then treated as if they were literals.

5. Strings computed at run time are newly created and therefore are distinct. (So line 4 prints false.)

6. The result of explicitly interning a computed string is the same string as any pre-existing literal string with the same contents.

We advise you to read section 3.10.5 String Literals in Java Language Specification

STRING BUFFER
Methods
- append()
- delete()
- insert()
- replace()
- reverse()
- keine trim() -Methode

STRING BUILDER
Methods
analolg zu StringBuffer

ARRAY LIST
Methods
- add()
- remove()
- set()
- isEmpty()
- size()

- clear()
- contains()
- equals()

ArrayList has a custom implementation of equals() so you can compare two lists
to see if they **contain the same elements in the same order**.
boolean equals(Object object) s134

Sorting
- Collection.sort(numbers);

LIST

LIST <> ARRAYLIST
List is an Interface. Arraylist is a class.
List is Generic. Arraylist is Specific.
The two can be substituted, but it is not recommended. This is the most recommended syntax:
List list = new ArrayList();

EQUALS
- String (Pool)
- String (Object)
- ArrayList

CONVERTING array and List
ArrayList -> array

3: **List<String> list = new ArrayList<>()**; //Liste erzeugen
4: list.add("hawk");
5: list.add("robin");

8: String[] stringArray = list.toArray(new String[0]);
9: System.out.println(stringArray.length); // 2

Array -> LIst

20: String[] array = { "hawk", "robin" }; // [hawk, robin]
21: **List<String> list = Arrays.asList(array)**; // returns fixed size list

SORTING

ACCESS MODIFIERS
Class -> protected in superclass
- Class inherits from superclass
    ClassMethod
-- without reference (directly)
        -> package access to superclass members
    ClassMethod + new Object
-- with Class Reference
        -> package access to superclass members
-- with superclass Reference
        -> no package access to superclass members //No Compiles