Java. Exception Handling

Лекція 3

Зміст

Вступ

Exceptions - непередбачені події під час виконання програми, які порушують нормальний потік виконання.

Мета обробки винятків: зберегти контроль над виконанням, коректно звільнити ресурси, повідомити користувача/логувати проблему.

Де застосовується: I/O, робота з мережею, парсинг, операції з масивами/колекціями, взаємодія з БД тощо.

Обробка помилок забезпечує стабільність: програма продовжує працювати або завершиться коректно.

Дозволяє відновитися або виконати запасний план (fallback).

Дозволяє централізовано логувати та діагностувати помилки.

Запобігає витоку ресурсів (файли, сокети, з’єднання з БД).

Покращує UX: зрозумілі повідомлення замість «crash» і stack trace для користувача.

Проблеми без обробки помилок

- Несподівані падіння програми (crash) → поганий UX
- Втрата або пошкодження даних (некоректні транзакції)
- Витоки ресурсів (не закриті файли/потоки) → деградація системи
- ідсутність контекстної інформації/логів → складність відлагодження
- Можливі питання безпеки (необроблені винятки можуть відкривати деталі системи у логах або виводі)

```java
public class Person {
    private String firstName;
    private String lastName;
    private String miidleName;

    public String getFirstName() { return firstName; }

    public void setFirstName(String firstName) { this.firstName = firstName; }

    public String getLastName() { return lastName; }

    public void setLastName(String lastName) { this.lastName = lastName; }

    public String getMiidleName() { return miidleName; }

    public void setMiidleName(String miidleName) { this.miidleName = miidleName; }

    public String fullName(){
        return "%s %c.%c.".formatted(lastName, firstName.charAt(0), miidleName.charAt(0));
    }

    public static void main(String[] args) {
        System.out.println("Start");
        Person p = new Person();
        p.setFirstName("Name");
        System.out.println(p.fullName());
        System.out.println("End");
    }
}
```

```
Start
Exception in thread "main" java.lang.NullPointerException Create breakpoint : Cannot invoke "String.charAt(int)" because "this.miidleName" is null
    at model.Person.fullName(Person.java:33)
    at model.Person.main(Person.java:40)
```

Основні ключові слова

`try` – блок коду, де може виникнути виняток.

`catch` – блок, який обробляє виняток.

`finally` – виконується завжди (навіть якщо є виняток). Використовується для звільнення ресурсів.

`throw` – використовується для генерації винятку.

`throws` – оголошує, що метод може кидати виняток.

```java
public static void main(String[] args) {
    System.out.println("Start");
    Person p = new Person();
    p.setFirstName("Name");
    try{
        System.out.println(p.fullName());
    }
    catch (NullPointerException e){
        System.out.println(e.getMessage());
    }
    finally {
        System.out.println("Finally block");
    }
    System.out.println("End");
}
```

```
8 ms
> Task :classes

> Task :Person.main()
Start
Cannot invoke "String.charAt(int)" because "this.miidleName" is null
Finally block
End
```

```java
public static void main(String[] args) {
    System.out.println("Start");
    Person p = new Person();
    p.setFirstName("FirstName");
    p.setMiidleName("Miidle");
    try{
        System.out.println(p.fullName());
    }
    catch (NullPointerException e){
        System.out.println(e.getMessage());
    }
    finally {
        System.out.println("Finally block");
    }
    System.out.println("End");
}
```

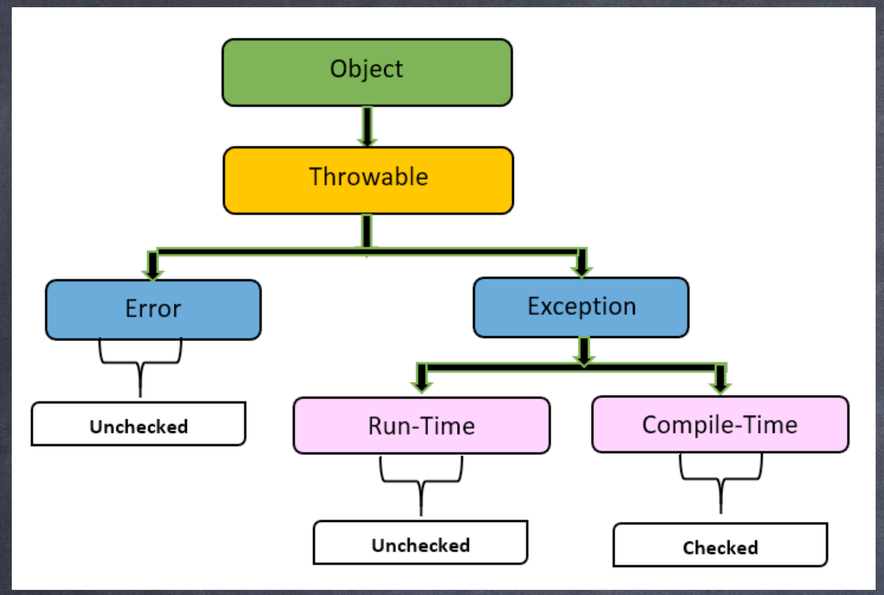
```
0 ms
> Task :classes

> Task :Person.main()
Start
null F.M.
Finally block
End
```

```java
public static void main(String[] args) {
    System.out.println("Start");
    Person p = new Person();
    p.setFirstName("FirstName");
    p.setMiddleName("");
    try{
        System.out.println(p.fullName());
    }
    catch (NullPointerException e){
        System.out.println(e.getMessage());
    }
    finally {
        System.out.println("Finally block");
    }
    System.out.println("End");
}
```

```java
public static void main(String[] args) {
    System.out.println("Start");
    Person p = new Person();
    p.setFirstName("FirstName");
    p.setMiddleName("");
    try{
        System.out.println(p.fullName());
    }
    catch (NullPointerException | StringIndexOutOfBoundsException e){
        System.out.println(e.getMessage());
    }
    finally {
        System.out.println("Finally block");
    }
    System.out.println("End");
}
```

```
Start
Finally block
Exception in thread "main" java.lang.StringIndexOutOfBoundsException Create breakpoint : Index 0 out of bounds for length 0 <7 internal lines>
    at java.base/java.lang.String.checkIndex(String.java:4832)
    at java.base/java.lang.StringLatin1.charAt(StringLatin1.java:46)
    at java.base/java.lang.String.charAt(String.java:1555)
    at model.Person.fullName(Person.java:33)
    at model.Person.main(Person.java:42)
```

```java
/**
 * Returns the formatted full name of a person in the form:
 * <pre>
 *     LastName F.M.
 * </pre>
 * where {@code F} is the first letter of the first name and {@code M} is the first letter
 * of the middle name (if present).
 *
 * @return the formatted full name string
 * @throws RuntimeException if either the first name or the last name is {@code null} or blank
 */
public String fullName() throws RuntimeException {
    if (lastName == null || lastName.isBlank() || firstName == null || firstName.isBlank()) {
        throw new RuntimeException("First name and last name must not be blank");
    }
    return "%s %c.%s.".formatted(
            lastName,
            firstName.charAt(0),
            middleName != null && middleName.length() > 0 ? middleName.charAt(0) : "\b"
    );
}
```



Ієрархія Exceptions

Exception – подія, яка виникає під час виконання програми та порушує нормальний потік інструкцій.

Всі винятки – об'єкти класів, що наслідуються від Throwable.

Ієрархія:

Throwable

- Error – серйозні помилки JVM (наприклад, OutOfMemoryError). Зазвичай не обробляються.
- Exception – винятки, які можна і потрібно обробляти.
  - Checked exceptions – перевіряються на етапі компіляції (наприклад, IOException, SQLException).
  - Unchecked exceptions (Runtime) – перевіряються під час виконання (наприклад, NullPointerException, ArithmeticException).

Custom Exception

RuntimeException

```java
public class PersonNameException extends Exception {
    public PersonNameException(){
        super();
    }
    public PersonNameException(String message){
        super(message);
    }

    public PersonNameException(Throwable throwable){
        super(throwable);
    }

    public PersonNameException(String firstName, String lastName){
        super(buildMessage(firstName, lastName));
    }

    private static String buildMessage(String firstName, String lastName) {
        StringBuilder sb = new StringBuilder();

        if (firstName == null || firstName.isBlank()) {
            sb.append("First name must not be blank (now: '").append(firstName).append("')\n");
        }
        if (lastName == null || lastName.isBlank()) {
            sb.append("Last name must not be blank (now: '").append(lastName).append("')\n");
        }

        return sb.toString().trim();
    }
}
```

```java
    /**
     * Returns the formatted full name of a person in the form:
     * <pre>
     *     LastName F.M.
     * </pre>
     * where {@code F} is the first letter of the first name and {@code M} is the first letter
     * of the middle name (if present).
     *
     * @return the formatted full name string
     * @throws PersonNameException if either the first name or the last name is {@code null} or blank
     */
    public String fullName() throws PersonNameException {
        if (lastName == null || lastName.isBlank() || firstName == null || firstName.isBlank()) {
            throw new PersonNameException(firstName, lastName);
        }
        return "%s %c.%s.".formatted(
                lastName,
                firstName.charAt(0),
                middleName != null && middleName.length() > 0 ? middleName.charAt(0) : "\b"
        );
    }
```

try-with-resources

```java
FileWriter writer = null;
try {
    writer = new FileWriter( fileName: "output.txt");
    writer.write( str: "Hello, world!\n");
    writer.write( str: "This is written to file.");
} catch (IOException e) {
    System.err.println("Помилка запису у файл: " + e.getMessage());
}
finally {
    if (writer != null){
        try {
            writer.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

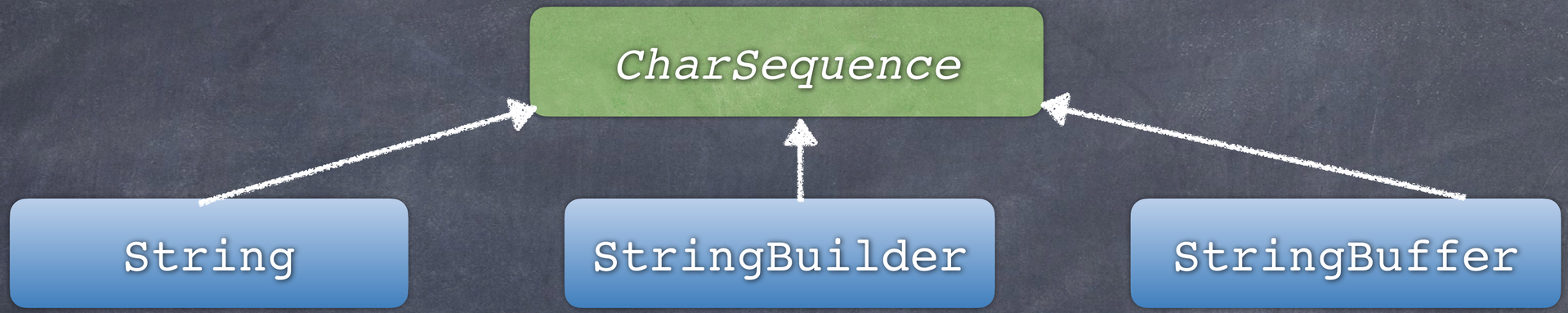
```java
try (FileWriter writer = new FileWriter( fileName: "output.txt")) {
    writer.write( str: "Hello, world!\n");
    writer.write( str: "This is written to file.");
} catch (IOException e) {
    System.err.println("Помилка запису у файл: " + e.getMessage());
}
```

Best practices

- Використовувати специфічні `catch`-блоки, а не загальний `Exception`.
- Уникати порожніх `catch`. Якщо потрібно ігнорувати виняток — залишати лог-запис.
- Використовувати `finally` або `try-with-resources` для закриття ресурсів.
- Не зловживати винятками для управління логікою програми.
- Використовувати стандартні винятки (`IllegalArgumentException`, `IllegalStateException`, `NullPointerException`) замість «голої» `RuntimeException`.
- Створювати власні винятки лише тоді, коли це виправдано бізнес-логікою.
- Давати інформативні повідомлення:
  - Погано → `"Error occurred"`
  - Добре → `"Invalid age: must be >= 18, but was -3"`
- Логувати винятки один раз (краще на «верхньому рівні»).
- Використовувати винятки тільки для виняткових ситуацій (не для контролю потоку).
- Не розкривати зайву інформацію у повідомленнях (`stack trace`, `SQL`-запити) у продакшені.
- Використовувати `multi-catch` (`Java 7+`), якщо обробка однакова.
- Дотримуватися принципу `fail-fast`: перевіряти умови завчасно (`Objects.requireNonNull(...)`).
- У великих застосунках — мати глобальний обробник винятків (`Spring`, `Jakarta EE`, `desktop-UI`).

Рядки в Java

Рядки в Java



Клас	Mutable	Thread-safe	Використання
String	❌	✅	Константні рядки, ключі, константи
StringBuilder	✅	❌	Операції з рядками в одному потоці
StringBuffer	✅	✅	Операції з рядками в багатопотоковому середовищі

Додавання рядків

- String firstName = “Ivan”;
- String lastName = “Bezruchko”;

- String fullName = firstName + “ ” + lastName;
- String fullName = firstName.concat(" “).concat(lastName);
- String fullName = String.format("%s %s", firstName, lastName);
- String fullName = String.join(" ", firstName, lastName);
- String fullName = "%s %s".formatted(firstName, lastName);

Formatter

- Клас для форматування рядків, чисел і дат.
- **Плейсхолдери**:
  - `%s` — рядок (`String`)
  - `%d` — ціле число (`int`, `long`)
  - `%f` — число з плаваючою точкою (`float`, `double`)
  - `%n` — новий рядок (залежний від платформи)
- **Вирівнювання та ширина**:
  - `%10s` — рядок, ширина 10 символів, по правому краю
  - `%-10s` — по лівому краю
- **Локалізація**:
  - Вказує формат чисел, дат, валют тощо через `Locale`.

Основні методи. Не забуваємо, що String immutable

- length()
- charAt(int index)
- codePointAt(int index)
- equals(Object obj)
- equalsIgnoreCase(String another)
- compareTo(String another)
- compareToIgnoreCase(String another)
- contains(CharSequence s)
- indexOf(String s)
- lastIndexOf(String s)
- startsWith(String prefix)
- endsWith(String suffix)
- substring(int beginIndex)
- substring(int begin, int end)
- replace(char oldChar, char newChar)
- replace(CharSequence target, CharSequence replacement)
- toLowerCase()
- toUpperCase()
- trim()
- strip()
- stripLeading()
- stripTrailing()
- split(String regex)
- join(CharSequence delimiter, CharSequence... elements)
- formatted(Object... args)
- format(String format, Object... args)
- toCharArray()
- isEmpty()
- isBlank()

Основні методи. Однакові для StringBuffer та StringBuilder

- append(...)
- insert(int offset, ...)
- delete(int start, int end)
- deleteCharAt(int index)
- replace(int start, int end, String str)
- charAt(int index)
- setCharAt(int index, char ch)
- getChars(int srcBegin, int srcEnd, char[] dst, int dstBegin)
- toString()
- length()
- capacity()
- ensureCapacity(int minimumCapacity)
- setLength(int newLength)
- reverse()

Робота з файлами

Робота з файлами

- File - представляє шлях до файлу або директорії, не для читання/запису.
- FileReader / FileWriter - для читання та запису тексту.
- BufferedReader / BufferedWriter - для ефективного читання/запису рядків.
- FileInputStream / FileOutputStream - для читання/запису байтів.
- Scanner - зручний для читання тексту з файлу.
- Files (java.nio.file) - сучасний API для читання, запису та маніпуляцій з файлами.

Основні операції

Створення файлу/директорії:

```
File file = new File("example.txt");
file.createNewFile();
file.mkdirs();
```

Читання тексту:

```
try (BufferedReader br = new BufferedReader(new
FileReader("example.txt"))) {
    String line;
    while ((line = br.readLine()) != null) {
        System.out.println(line);
    }
} catch (IOException e) {
    e.printStackTrace();
}
```

Запис тексту:

```
try (BufferedWriter bw = new BufferedWriter(new
FileWriter("example.txt"))) {
    bw.write("Hello, Java!");
    bw.newLine();
} catch (IOException e) {
    e.printStackTrace();
}
```

Читання/запис байтів:

```
try (FileInputStream fis = new FileInputStream("file.bin");
     FileOutputStream fos = new FileOutputStream("copy.bin")) {

    int b;
    while ((b = fis.read()) != -1) {
        fos.write(b);
    }
} catch (IOException e) {
    e.printStackTrace();
}
```

Сучасний спосіб з Files (Java 7+):

```
Path path = Paths.get("example.txt");
List<String> lines = Files.readAllLines(path);
Files.write(path, Arrays.asList("Hello", "World"));
```

Читання csv-рядків

23, Some,2020

```
public record Group(
        int number,
        String specialty,
        int startYear
) {}

public static Group parseGroupFromLine(String
line) throws InvalidDataException {
    String[] parts = line.split(",");
    if (parts.length != 3) {
        throw new InvalidDataException(
                "Expected format
'number,specialty,startYear', got: " + line
        );
    }
    try {
        int number =
Integer.parseInt(parts[0].trim());
        String specialty = parts[1].trim();
        int startYear =
Integer.parseInt(parts[2].trim());

        return new Group(number, specialty,
startYear);
    } catch (NumberFormatException e) {
        throw new InvalidDataException(
                "Invalid number format in line: " + line, e
        );
    }
}
```