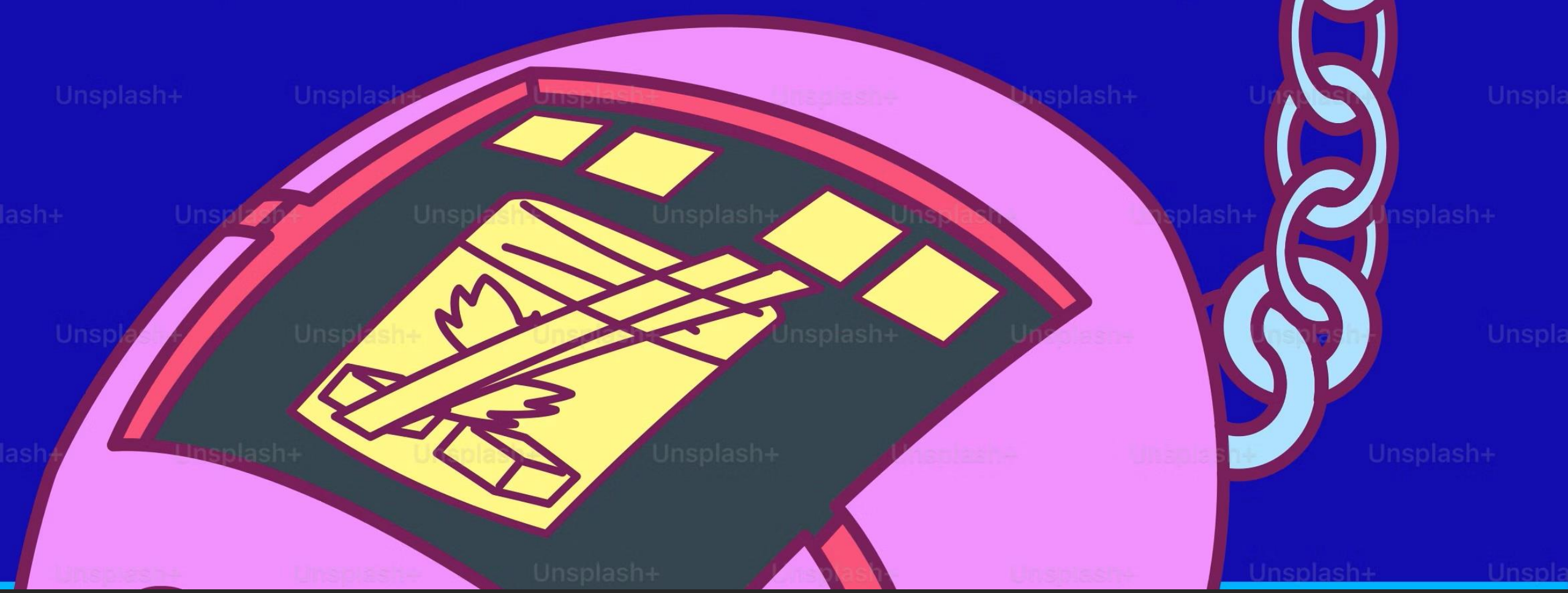

Tamagotchi – Tag 4

12. Dezember 2024

Plan für die Woche

Montag	Dienstag	Mittwoch	Donnerstag	Freitag
• Wiederholung	• Test 60min • JavaFX (Abriss)	• Exceptions in großen Projekten	• Überladen von Methoden • Interfaces • Strings	• Speichern und Laden

Plan für heute

- Speichern und Laden

Strings

Selbststudium

Erarbeite dir in *Deshmukh (2024):Java Foundations Exam 1Z0-811* das Kapitel 13: *Working with the String Class*.

1. Erstelle dir eine A4 Seite mit einer Zusammenfassung und den wichtigsten Informationen zu dem Thema.
2. Löse die Aufgaben in 13.3 Exercise.
3. **Zusatz**: Finde zwei Stellen in unserem Tamagotchi-Code, an denen man die Konzepte anwenden kann.

Zusammenfassungen & Lösungsvorschläge der Aufgaben

Speichern und Laden

Grundlagen

- Schreiben und Lesen aus Dateien
- Verschiedene Formate:
  - Textdateien
  - Binärdateien
- Verwendung von Streams
  - Byte-Streams (Binärdaten z.B. FileInputStream/FileOutputStream)
  - Character-Stream (Textdaten z.B. FileReader/FileWriter)
- Können I/O-Exceptions werfen

Info: Ein **Stream** ist eine Abstraktion, die es ermöglicht, Daten sequenziell zu lesen oder zu schreiben, wobei man sich nicht um die interne Speicherung kümmern muss..

Speichern von Text

```java
import java.io.FileWriter;
import java.io.IOException;

public class FileSaveExample {
    public static void main(String[] args) {
        try (FileWriter writer = new FileWriter("example.txt")) {
            writer.write("Hallo, Welt!");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Laden von Text

```java
import java.io.FileReader;
import java.io.BufferedReader;
import java.io.IOException;

public class FileLoadExample {
    public static void main(String[] args) {
        try (BufferedReader reader = new BufferedReader(new FileReader("example.txt"))) {
            String line;
            while ((line = reader.readLine()) != null) {
                System.out.println(line);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Serialisierung

- Objekte werden in einen Bytestream umgewandelt
- Klasse muss das *Serializable*-Interface implementieren

Beispiel: Serialisierung

```java
public class Person implements Serializable {
    private String name;
    private int age;

    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }

    @Override
    public String toString() {
        return name + ", " + age;
    }
}

public class SerializationExample {
    public static void main(String[] args) {
        Person person = new Person("Max Mustermann", 25);

        try (FileOutputStream fileOut = new FileOutputStream("person.ser");
             ObjectOutputStream out = new ObjectOutputStream(fileOut)) {
            out.writeObject(person);
        } catch (IOException i) {
            i.printStackTrace();
        }
    }
}
```

Beispiel: Deserialisierung

```java
public class DeserializationExample {
    public static void main(String[] args) {
        try (FileInputStream fileIn = new FileInputStream("person.ser");
             ObjectInputStream in = new ObjectInputStream(fileIn)) {
            Person person = (Person) in.readObject();
            System.out.println(person);
        } catch (IOException | ClassNotFoundException e) {
            e.printStackTrace();
        }
    }
}
```

Dateipfade

- möglichst relative Dateipfade angeben z.B. meinText.txt
- absolute Pfade müssen systemabhängig angepasst werden z.B. /home/user/meinText.text

Exkurs: Stream-Api (Java 8)

- funktionale und deklarative Art der Verarbeitung von Daten
- Streams sind nicht direkt mit Dateioperationen verbunden, sondern bieten eine Möglichkeit, Daten zu sammeln, zu transformieren und zu filtern
- Hilfreich bei Listen

```
List<String> names = Arrays.asList("Anna", "Max", "John");
names.stream()
     .filter(name -> name.startsWith("A"))
     .map(String::toUpperCase)
     .forEach(System.out::println);  // Gibt "ANNA" aus
```

Quizfragen

Pursley (2020) - **Question 41-55** (Bitte in OneNode abtippen. – Seitenzahl angeben!)

OneNode – **Question 10, 25, 51, 64, 69, 76, 97**

1. Teilt euch in Kleingruppen auf. (Die Fragen werden aufgeteilt.)
2. Erarbeitet eine Lösung inkl. Begründung der euch zugewiesenen Fragen.
3. Präsentiert eure Ergebnisse euren Kollegen.

Quellen

https://unsplash.com/de/grafiken/ein-rosafarbenes-objekt-an-dem-eine-kette-befestigt-ist-0adN6t6_zzs