01. Setup

Git-Repository initialisiert	git init
Alle Dateien hinzugefügt	git add -A
Ersten Commit erstellt	git commit -m 'Finished setup for IntroductionExercise.java'
Remote-Repository verknüpft	git remote add origin https://github.com/druxxangie/at.aau.se1.git

02. Implementierung

OpenAI GPT-4 (ChatGPT-4). URL: https://openai.com/chatgpt

https://chatgpt.com/c/670bc464-0f48-8002-a976-d04660d76369

Umrechnen von Binärzahlen in Dezimalzahlen

Git Log

commit		
1	Finished setup for IntroductionExercise.java	using gpt-4
2	Implemented binary to decimal conversion method binary2decimal with int[] binary as input + input validation + using math.pow(2, length - 1 - i) for calculating binary digits	using gpt-4
3	Updated input method to accept binary number from console input and validate user input	using gpt-4
4	Replaced Math.pow with bit shifting for binary to decimal conversion, added overflow check to prevent integer overflow, and improved input validation for binary number entry. Some orange hints occured.	using gpt-4
5	Removed unnecessary null check for input and improved validation for empty input in console	using gpt-4

commit		
6	Improved error handling: added check for invalid characters, input length limit to 31 bits, support for removing whitespace, and handling of negative binary inputs. Fixed loop behavior for invalid input.	using gpt-4

Erklärung der binary2decimal Logik:

```
// Schiebeoperation statt Math.pow
decimal = (decimal << 1) | j;
```

1. Schiebeoperation << 1

decimal << 1 bedeutet: Verschiebe den Wert der Variablen decimal um eine Position nach links in seiner Binärdarstellung. Das hat den Effekt, dass der Wert verdoppelt wird, was dem Multiplizieren mit 2 entspricht.

2. Bitweise ODER-Operation |

Der Ausdruck | j ist eine bitweise ODER-Operation.
Dabei wird das Ergebnis von decimal << 1 (also die links verschobene Binärzahl) mit dem aktuellen Binärwert j (das Bit an der aktuellen Stelle in der Binärzahl) kombiniert.
Das bedeutet, dass das niedrigstwertige Bit von decimal durch den aktuellen Wert von j gesetzt wird (entweder 0 oder 1).

Schreibtischtest:

			lsb –> ungerade
1	**1**	**0**	**1**
3	2	1	0
1x2^3	1x2^2	0x2^1	1x2^0
8	4	0	1
		8+4+0+1=	13

Nehmen wir die Binärzahl `1101` (das entspricht der Dezimalzahl **13**) und wandeln sie mit dem Code um:

1. **Initialisierung**: `decimal = 0`
   - Die Variable `decimal` wird auf 0 gesetzt.
2. **Verarbeitung des ersten Bits (1)**:
   - `decimal = (0 << 1) | 1 = 1`
   - Der Wert von `decimal` ist jetzt 1 (in Binär: `1`).
3. **Verarbeitung des zweiten Bits (1)**:
   - `decimal = (1 << 1) | 1 = 3`
   - Das 1 wird um eine Stelle nach links verschoben (von `1` zu `10` in Binär, was 2 entspricht), dann wird das nächste `1` hinzugefügt, was den Wert 3 ergibt (in Binär: `11`).
4. **Verarbeitung des dritten Bits (0)**:
   - `decimal = (3 << 1) | 0 = 6`
   - Das `11` (3 in Dezimal) wird um eine Stelle nach links verschoben zu `110` (was 6 in Dezimal ist), und das nächste Bit `0` wird hinzugefügt, was den Wert unverändert lässt.
5. **Verarbeitung des vierten Bits (1)**:
   - `decimal = (6 << 1) | 1 = 13`
   - Das `110` (6 in Dezimal) wird um eine Stelle nach links verschoben zu `1100` (was 12 in Dezimal ist), und das letzte Bit `1` wird hinzugefügt, was den Wert auf **13** erhöht (in Binär: `1101`).

03. Codeanalyse / Diskussion

1. Ist die mittels LLM erstellte Implementierung korrekt? Wie sind Sie zu diesem Schluss (korrekt vs. nicht korrekt) gekommen?

LLM hat mir im ersten Schritt eine solide Implementierung vorgeschlagen, hat sich aber für die Binärumrechnung mittels math.pow entschieden —> beim zweiten Durchlauf wurde die Binärumrechnung durch die Verwendung der Schiebeoperation abgeändert, mit der Begründung, dass dies zwar die Lesbarkeit schwächt, aber genauere Ergebnisse erzielen kann. (Ich persönlich bevorzuge die Berechnung siehe S.2, bitweise Potenzen berechnen.)

Ja, die Implementierung war korrekt, es waren aber weitere spezifische Anweisungen zur Implementierung notwendig.

2. Welche Fehler können bei der Ausführung des Programms auftreten? Wie werden diese Fehler in der Implementierung berücksichtigt?

1) Leere oder null-Eingabe über Konsole (bei Ausführung der binary2decimal Methode)

—> wirft eine IllegalArgumentException mit der Nachricht "Die Eingabe darf nicht null oder leer sein."

2) Ungültige Binärwerte im Array

—> wirft eine IllegalArgumentException mit der Nachricht "Binärzahlen dürfen nur 0 oder 1 ethalten."

3) Overflow bei der Umwandlung

—> wirft eine ArithmeticException mit der Nachricht "Overflow: Die Binärzahl ist zu groß, um als Integer umgerechnet zu werden."

Zahl übersteigt den maximalen Wert eines 32 Bit Integers (2^31) - 1 = 2147483647

4) Leere Benutzereingabe

—> gibt mittels System.out.println die Nachricht "Die Eingabe darf nicht leer sein." aus

5) Zu lange Binärzahl (wenn >31 Stellen)

—> Overflow, gibt mittels System.out.println die Nachricht "Die Binärzahl darf maximal 31 Stellen haben." aus.

6) Ungültige Zeichen in der Eingabe (wenn nicht 0 und nicht 1)

—> Gibt mittels System.out.println die Nachricht "Ungültige Eingabe. Nur 0 oder 1 sind erlaubt." aus und startet die Schleife neu.

7) Exception e (wenn unerwarteter Fehler auftritt)

—> Fängt mittels catch(Exception e) alle unerwarteten Ausnahmen ab und gibt eine Fehlermeldug mittels System.out.println aus: "Ein unerwarteter Fehler ist aufgetreten."

3. Beschreiben Sie einen Normalfall der Programmausführung, indem Sie (i) Inputs, (ii) Ausführungspfad und (iii) Outputs für den von Ihnen ausgewählten Fall angeben.

(i) inputs	(ii) Ausführungspfad	(iii) outputs
1101	Eingabeaufforderung Eingabe Überprüfung der Eingabe (Leerzeichen, >31 Zeichen) Überprüfung der Zeichen (sind es 0 und 1) Umwandlung in ein Integer Array {1, 1, 0, 1} Methodenaufruf binary2decimal Überprüfung ob Array leer oder null ist Umwandlung mittels Schiebeoperation (0 << 1) | 1 = 1 (1 << 1) | 1 = 3 (3 << 1) | 0 = 6 (6 << 1) | 1 = 13	Die Dezimalzahl ist: 13

4. Beschreiben Sie einen Fehlerfall der Programmausführung, indem Sie (i) Inputs, (ii) Ausführungspfad und (iii) Outputs für den von Ihnen ausgewählten Fall angeben.

(i) inputs	(ii) Ausführungspfad	(ii) outputs
11012	Eingabeaufforderung Eingabe … … Überprüfung der Zeichen Bemerkt, dass letztes Zeichen keine 0 und keine 1 ist Fehlerfall	Ungültige Eigabe. Nur 0 oder 1 sind erlaubt.