🧩 3. Rejestr wydatków

🎯 Cel programu:

Użytkownik tworzy prosty rejestr swoich wydatków domowych.
Może dodać nowy wydatek, wyświetlić historię i obliczyć sumę wszystkich kosztów.

📋 Wymagania:

Menu programu:

```
1. Dodaj nowy wydatek
2. Pokaż historię wydatków
3. Oblicz sumę wydatków
4. Wyjdź
```

1.
2. Dodawanie wydatku:
    - Pobierz:
        - nazwę/opis wydatku,
        - kategorię (np. „jedzenie”, „transport”),
        - kwotę (`double`).
    - Jeśli kwota ≤ 0 → rzuć wyjątek (`IllegalArgumentException`).

Zapisz dane do pliku `wydatki.txt` w formacie:

```
[2025-11-13 19:20] – Kategoria: jedzenie – Kwota: 25.50 – Opis: obiad
```

    -
3. Wyświetlanie wydatków:
    - Odczytaj i wyświetl każdą linię z pliku `wydatki.txt`.

Jeśli plik nie istnieje — utwórz nowy i pokaż informację:

```
Brak wydatków do wyświetlenia.
```

-

4. Obliczanie sumy:
    - Wczytaj zawartość pliku.
    - Wyszukaj liczby po słowie „Kwota:” i zsumuj je.

Wyświetl wynik:

```
Łączna suma wydatków: 157.30 zł
```

-

5. Obsługa błędów:
    - Obsłuż błędy pliku (`IOException`) i błędne dane użytkownika (`NumberFormatException`).

W `finally` wypisz komunikat:

```
Operacja zakończona pomyślnie.
```

-

🧰 Wymagane biblioteki i funkcje:

Biblioteki:

```
import java.io.*;                          // FileReader, FileWriter, BufferedReader, IOException
import java.time.*;                        // LocalDateTime
import java.time.format.DateTimeFormatter; // formatowanie daty i godziny
import java.util.*;                        // Scanner, ArrayList
```

Funkcje/metody:

- `Scanner.nextLine()` / `Scanner.nextDouble()` – pobieranie danych
- `LocalDateTime.now()` – data i godzina zapisu
- `DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm")` – format daty

- `FileWriter("wydatki.txt", true)` – zapis do pliku
- `BufferedReader.readLine()` – odczyt pliku linia po linii
- `Double.parseDouble(String)` – konwersja tekstu na liczbę
- `String.contains("Kwota:")` / `split(" ")` – analiza tekstu w pliku
- `try/catch/finally` – obsługa błędów