Develop a Java application that converts binary numbers (input as a string) into decimal numbers.

### 1. Overview of the application

This Java-based application will convert binary numbers (input as strings) into decimal numbers, as well as manage expenses. The application will feature a basic interface for user interaction (either a console or GUI) and store expenses in a local database (SQLite, JDBC). It will also generate reports in Excel format using Apache POI, and allow for importing/exporting data using JSON/XML.

**2. User Interface**

* **Option 1: Console-based interface** (Basic functionality)
  + Users can enter binary numbers to convert to decimal.
  + Users can manage expenses by entering them, editing or deleting them.
  + Optionally, generate reports directly from the console.
* **Option 2: GUI Interface (Optional)** using **Swing** or **JavaFX**
  + A simple window with input fields and buttons for interacting with the system.
  + A table or list for displaying expenses, adding/editing/deleting records, and generating reports.

**3. Main Features**

**A. Binary to Decimal Conversion:**

* User inputs a binary string.
* The program converts the binary string to its decimal equivalent.

**B. Expense Management:**

* **Add Expense**: Capture details like amount, category, date, description, etc.
* **Edit Expense**: Modify the details of an existing expense.
* **Delete Expense**: Remove an expense from the system.
* **View Expenses**: List all expenses with options to filter by categories, dates, etc.

**C. Reports:**

* Generate reports based on expenses.
* Generate breakdowns by category, or generate monthly/weekly reports.
* Export reports to **Excel** (using Apache POI) or **JSON/XML** (for export/import).

**D. Database Integration:**

* Use **SQLite** for local data storage.
* Store expenses with details in an SQLite database.
* Retrieve, edit, and delete expense records from the database.

**4. Technologies and Tools**

* **Java**: The core programming language.
* **Swing/JavaFX**: For GUI (optional).
* **JDBC/SQLite**: For interacting with the local database.
* **Apache POI**: For generating Excel reports.
* **JSON/XML**: For exporting or importing data.

**5. Application Flow**

1. **Binary-to-Decimal Converter**:
   * When the user inputs a binary string, the program converts the binary to decimal and displays the result.
2. **Expense Management**:
   * Use commands or GUI buttons to add, edit, or delete expenses.
   * Expenses are stored in a local database.

**6. Detailed Plan and Implementation**

**A. Binary-to-Decimal Conversion Code**

This function will take a binary string and return its decimal equivalent:

import java.util.Scanner;

public class BinaryToDecimal {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter a binary number: ");

String binaryInput = scanner.nextLine();

try {

int decimal = Integer.parseInt(binaryInput, 2);

System.out.println("Decimal equivalent: " + decimal);

} catch (NumberFormatException e) {

System.out.println("Invalid binary number.");

}

}

}

# OUTPUT

Enter a binary number: 1011

Decimal equivalent: 11

**public class Main {**

**public static void main(String[] args) {**

**// Creating some expense objects**

**Expense expense1 = new Expense(1, "Food", 45.67, "2025-02-05", "Lunch at cafe");**

**Expense expense2 = new Expense(2, "Travel", 120.50, "2025-02-03", "Taxi fare to airport");**

**Expense expense3 = new Expense(3, "Entertainment", 80.00, "2025-02-06", "Movie tickets");**

**// Displaying the expense details using toString method**

**System.out.println("Expense Details:\n");**

**System.out.println(expense1);**

**System.out.println("\n-------------------------\n");**

**System.out.println(expense2);**

**System.out.println("\n-------------------------\n");**

**System.out.println(expense3);**

**}**

**}**

**class Expense {**

**private int id;**

**private String category;**

**private double amount;**

**private String date;**

**private String description;**

**// Constructor to initialize all fields**

**public Expense(int id, String category, double amount, String date, String description) {**

**this.id = id;**

**this.category = category;**

**this.amount = amount;**

**this.date = date;**

**this.description = description;**

**}**

**// Getters and Setters for each field**

**public int getId() {**

**return id;**

**}**

**public void setId(int id) {**

**this.id = id;**

**}**

**public String getCategory() {**

**return category;**

**}**

**public void setCategory(String category) {**

**this.category = category;**

**}**

**public double getAmount() {**

**return amount;**

**}**

**public void setAmount(double amount) {**

**this.amount = amount;**

**}**

**public String getDate() {**

**return date;**

**}**

**public void setDate(String date) {**

**this.date = date;**

**}**

**public String getDescription() {**

return description;

**}**

**public void setDescription(String description) {**

**this.description = description;**

**}**

**// Override toString() for easy display of expense details**

@Override

**public String toString() {**

**return "Expense ID: " + id + "\n" +**

### "Category: " + category + "\n" +

**"Amount: " + amount + "\n" +**

**"Date: " + date + "\n" +**

"Description: " + description;

**}**

**}**

# Output :

Expense Details:

Expense ID: 1

Category: Food

Amount: 45.67

Date: 2025-02-05

Description: Lunch at cafe

Expense ID: 2

Category: Travel

Amount: 120.5

Date: 2025-02-03

Description: Taxi fare to airport

Expense ID: 3

Category: Entertainment

Amount: 80.0

Date: 2025-02-06

Description: Movie tickets