Understanding registerOutParameter(int parameterIndex, int sqlType)

- It is mainly used to register output parameters for a CallableStatement.

Syntax:

```
void registerOutParameter(int parameterIndex, int sqlType) throws SQLException
```

- It registers the data type of a parameter that is expected to return a value (OUT or INOUT parameters) from a stored procedure.
- It allows JDBC driver to know what kind of data to expect when the stored procedure executes.
- After calling registerOutParameter(), when the stored procedure is executed, the CallableStatement object can retrieve the value of the output parameter using the appropriate getter methods (e.g., getString(), getInt()).

Transaction Management

Understanding Transaction Management

- The process of combining related operations into a single unit and executing them by applying **do everything** (or) **nothing** principle is called **Transaction management**.
- Every Transaction management contains 3 operations.
  - Begin Transaction **[AutoCommit(false)]**
  - Execute logics
  - Commit/rollback Transaction **[commit() (or)rollback()]**
- A valid Transaction should obey ACID properties

1) Atomicity
2) Consistency
3) Isolation
4) Durability

Atomicity:

Ensures that all operations within a transaction are completed successfully. If even one operation fails, the entire transaction is rolled back, and no changes are applied to the database.

Example: Either the entire IRCTC ticket booking process (seat lock, booking creation, and payment) is completed successfully, or none of it is.

Consistency:

Ensures that the database remains in a consistent state before and after a transaction. The data integrity rules, such as primary and foreign keys, should be maintained.

Example: If a booking fails, the seat count and booking records should remain unchanged, ensuring no inconsistent states.

Isolation:

Ensures that the operations within a transaction are not visible to other transactions until the transaction is completed. This prevents data corruption and ensures that transactions do not interfere with each other.

Example: While one transaction (Seat booking) is in progress, other transactions should not see its intermediate states (e.g., temporary seat locks)

Durability:

Guarantees that once a transaction is committed, its changes are permanent and will not be lost, even in the event of a system failure.

Example: Once the ticket is confirmed, it should be saved permanently in the database, even if a system failure occurs later.

Important methods used in JDBC Transaction Management

Method Name	Description
getAutoCommit()	Returns the current state of the auto-commit mode. If true, each SQL statement is committed automatically.
setAutoCommit(boolean autoCommit)	Enables or disables the auto-commit mode. Set to false to manage transactions manually using commit() and rollback().
setSavepoint()	Creates a savepoint in the current transaction, allowing a partial rollback to this specific point.
releaseSavepoint(Savepoint savepoint)	Deletes the specified savepoint from the transaction.
commit()	Commits all changes made in the current transaction, making them permanent in the database.
rollback()	Undoes all changes made in the transaction since the last commit(), restoring the previous state.

Step-by-Step Implementation Guide for Transaction Management

Step 1: Create Database Tables in Oracle
Step 2: Create a Java Project and Setup JDBC
Step 3: Define the JDBC Connection Properties
Step 4: Create a Database Connection & Disable Autocommit
Step 5: Lock a Seat in TrainSeatAvailability Table ===> SQL Query
Step 6: Create a Savepoint & commit the transaction
Step 7: Insert a New Booking Record ===> SQL Query
Step 8: Check Payment Status of the Customer ===> SQL Query
Step 9: Update Booking Status and Commit/Rollback the Transaction ===> SQL Query
Step 10: Handle Exceptions and Close the Connection

Transaction Management Example tables

Table 1 :

TrainSeatAvailability			
train_id	journey_date	class	available_seats
12345	10-10-2024	Sleeper	10

Table 2 :

BookingDetails				
booking_id	train_id	customer_id	seat_number	status

Table 3:

CustomerPayment			
payment_id	customer_id	amount	payment_status
P1001	C123	500	Success