Template Method design pattern

Template Method Scenario: Online exam system where steps like startTest(), evaluate(), showResult() are fixed, but questions vary. give java code in depth explaination with anologies , jdk internal examples sprig boot use case jdbctemplate etc

Let's explore the **Template Method** design pattern using your **Online Exam System** scenario.

🧩 What is Template Method Pattern?

Template Method defines the *skeleton* of an algorithm in a superclass but lets subclasses override specific steps without changing the algorithm's structure.

🧠 Analogy: Exam Paper

In a **standardized online exam system**:

- Steps like `startTest()`, `evaluate()`, and `showResults()` are **fixed**.
- But the **actual questions** or `presentQuestions()` vary depending on the exam (Java, Python, Math).

That's a **Template Method** pattern:

- The "template" defines the algorithm steps.
- Subclasses plug in their specific behavior.

✅ Java Implementation of Template Method Pattern

🔶 Abstract Class — `OnlineExam`

```
public abstract class OnlineExam {
    // Template Method
    public final void conductExam() {
        startTest();
        presentQuestions();  // customizable step
        evaluate();
        showResult();
    }

    private void startTest() {
        System.out.println("Test started. Timer is running...");
    }

    protected abstract void presentQuestions();  // variation point

    private void evaluate() {
        System.out.println("Evaluating answers...");
    }
    private void showResult() {
        System.out.println("Displaying result to the student.");
    }
}
```

🔶 Subclass 1 — Java Exam

```
public class JavaExam extends OnlineExam {
    @Override
    protected void presentQuestions() {
        System.out.println("Java Q1: What is JVM?");
        System.out.println("Java Q2: Explain OOP principles.");
    }
}
```

🔶 Subclass 2 — Math Exam

```
public class MathExam extends OnlineExam {
    @Override
    protected void presentQuestions() {
        System.out.println("Math Q1: Solve integration problem.");
        System.out.println("Math Q2: What is Pythagoras Theorem?");
    }
}
```

🔶 Client Code

```
public class ExamClient {
    public static void main(String[] args) {
        OnlineExam exam1 = new JavaExam();
        exam1.conductExam();

        System.out.println("-----");

        OnlineExam exam2 = new MathExam();
        exam2.conductExam();
    }
}
```

✅ Output

```
Test started. Timer is running...
Java Q1: What is JVM?
Java Q2: Explain OOP principles.
Evaluating answers...
Displaying result to the student.
-----
Test started. Timer is running...
Math Q1: Solve integration problem.
Math Q2: What is Pythagoras Theorem?
Evaluating answers...
```

```
Displaying result to the student.
```

🛠️ JDK Internal Example of Template Method

🔹 `java.io.InputStream`, `OutputStream`, `Reader`, `Writer`

```
java
CopyEdit
public abstract class InputStream {
    public int read(byte[] b) {
        // A concrete method using the abstract read()
        int bytesRead = 0;
        for (int i = 0; i < b.length; i++) {
            int result = read(); // abstract method
            if (result == -1) break;
            b[i] = (byte) result;
            bytesRead++;
        }
        return bytesRead;
    }

    public abstract int read();  // to be implemented by subclass
}
```

`read(byte[])` is the **template**, `read()` is the variation point.

🌱 Spring Boot Example – `JdbcTemplate`

🔹 `JdbcTemplate.query(...)` method

`JdbcTemplate` defines the structure of DB access:

```
String sql = "SELECT * FROM users";
List<User> users = jdbcTemplate.query(sql, new UserRowMapper());
```

Internally, this does:

1. Open connection
2. Prepare statement
3. Execute query
4. **Delegate row mapping** to your provided `RowMapper`

🔹 You provide only this "custom step":

```
public class UserRowMapper implements RowMapper<User> {
    public User mapRow(ResultSet rs, int rowNum) throws SQLException {
        return new User(rs.getInt("id"), rs.getString("name"));
    }
}
```

Here, Spring uses the Template Method pattern to **define the DB interaction skeleton**, and lets you provide the row mapping logic.

🧪 Other Use Cases in Spring Boot

Use Case	Template Logic	Your Custom Step
`RestTemplate`	Sends HTTP requests	You provide response extraction code
`TransactionTemplate`	Begin → Commit/Rollback transaction	You write the business logic in between
`JdbcTemplate`	Handles DB connection & exception	You give `RowMapper`
`RetryTemplate`	Handles retries	You provide retryable action

📌 When to Use Template Method

- You want a fixed sequence of steps
- Some steps may vary across implementations
- Reuse common structure, avoid code duplication

✅ Summary

Aspect	Template Method Pattern
Intent	Algorithm skeleton with variation points
Variation Point	Subclasses implement specific steps
Common Use Case	Workflow, testing pipelines, data loading
JDK Usage	InputStream/OutputStream, Executors
Spring Boot Usage	JdbcTemplate, RestTemplate, RetryTemplate
