Lambda and JDBC explanation

UNIT – IV FILE STREAMS AND DATABASE 9L, 12P
Java I/O- Reading and Writing Files - Regular Expressions – Streams API - Object Serialization – Generic collections – Generic Classes - Lists, set, Maps – GenericMethods - Lambda expressions – Databases with JDBC.

i need only lamda and jdbc

Sure 👍 Here's a clear, **5-mark level explanation** for both **Lambda Expressions** and **JDBC in Java** — simple, concise, and easy to remember.

---

1. Lambda Expressions (Java 8 feature)

Definition:
Lambda expressions are used to represent **anonymous functions** (functions without a name).
They make the code **shorter, cleaner, and more readable**, especially when working with **functional interfaces** (interfaces with a single abstract method).

Syntax:

```
java

(parameters) -> { body }
```

Example:

```
java

// Without Lambda
Runnable r = new Runnable() {
    public void run() {
        System.out.println("Running...");
    }
};

// With Lambda
Runnable r = () -> System.out.println("Running...");
r.run();
```

Key Points:

- Introduced in **Java 8**.
- Works mainly with **functional interfaces** (e.g., Runnable, Comparator).
- Used in **Streams API** and **Collections**.
- Improves **code readability** and **reduces boilerplate**.

Advantages:

- Shorter and expressive code
- Enables **functional programming** style
- Makes it easy to use **streams** and **collection operations**

---

2. JDBC (Java Database Connectivity)

Definition:
JDBC is an **API (Application Programming Interface)** in Java used to **connect and interact with databases**.
It allows Java programs to **execute SQL queries**, **insert**, **update**, **delete**, and **retrieve** data from databases.

Steps in JDBC:

1. **Import the package**

```
java

import java.sql.*;
```

2. **Load the driver**

```
java

Class.forName("com.mysql.cj.jdbc.Driver");
```

3. **Establish the connection**

```
java

Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/dbname", "user",
"password");
```

4. **Create a statement**

```
java

Statement stmt = con.createStatement();
```

5. **Execute the query**

```
java

ResultSet rs = stmt.executeQuery("SELECT * FROM students");
```

6. **Process the results**

```
java

while(rs.next()) {
    System.out.println(rs.getInt(1) + " " + rs.getString(2));
```

```
}
```

7. **Close the connection**

```
java

con.close();
```

Advantages:

- Platform-independent way to access databases
- Supports all major databases (MySQL, Oracle, PostgreSQL, etc.)
- Provides both **Statement** and **PreparedStatement** for secure and efficient queries

---

Would you like me to make it into a **2-mark + 3-mark split version** (like in exams) or keep it as this single short note format?