JDBC

OCP DAY 12

CONTENT

- Relational databases
- Basic SQL statements
- JDBC
- Interfaces of JDBC
- Statement: obtaining and executing
- Resultset
- Closing resources
- Exceptions

PLAN VAN DE DAG

9.00-11.00: JDBC

11.00-12.00: ALGEMENE VRAGEN

13.00-14.30: FOUNDATION TEST OCP SAMEN DOORLOPEN

14.30-16.30: FOUNDATION TEST AFMAKEN EN AANDACHTSPUNTEN DOORNEMEN

JDBC

- Java API
- Database: collection of data
- Java Database Connectivity: accessing databases using Java
- Accessing data as rows and columns
- Relational database: consisting of tables with relationships between the tables

RELATIONAL DATABASES AND JAVA

- Two ways to access relational databases in Java:
  - JDBC: access data as rows and columns
  - JPA (Java Persistence API): access data through objecs using an ORM

- SQL: Structured Query Language, used to write commands to the database
- NoSQL: other way of storing and organizing data

Exercise: explain how relational databases work. Give an example.

SQL STATEMENTS

- INSERT
- SELECT
- UPDATE
- DELETE

- Keywords are case insensitive

JDBC INTERFACE

- JDBC has four interfaces to cover the functionality:
  - Driver: obtain connection with DB
  - Connection: communicate with DB
  - Statement: runs SQL
  - ResultSet: returned data by queries
- JDBC drivers have implementations for these
- All databases have their own jar file with an implementation of these interfaces
- You use the interface in code and implementation is coming from the implemented driver

EXAMPLE

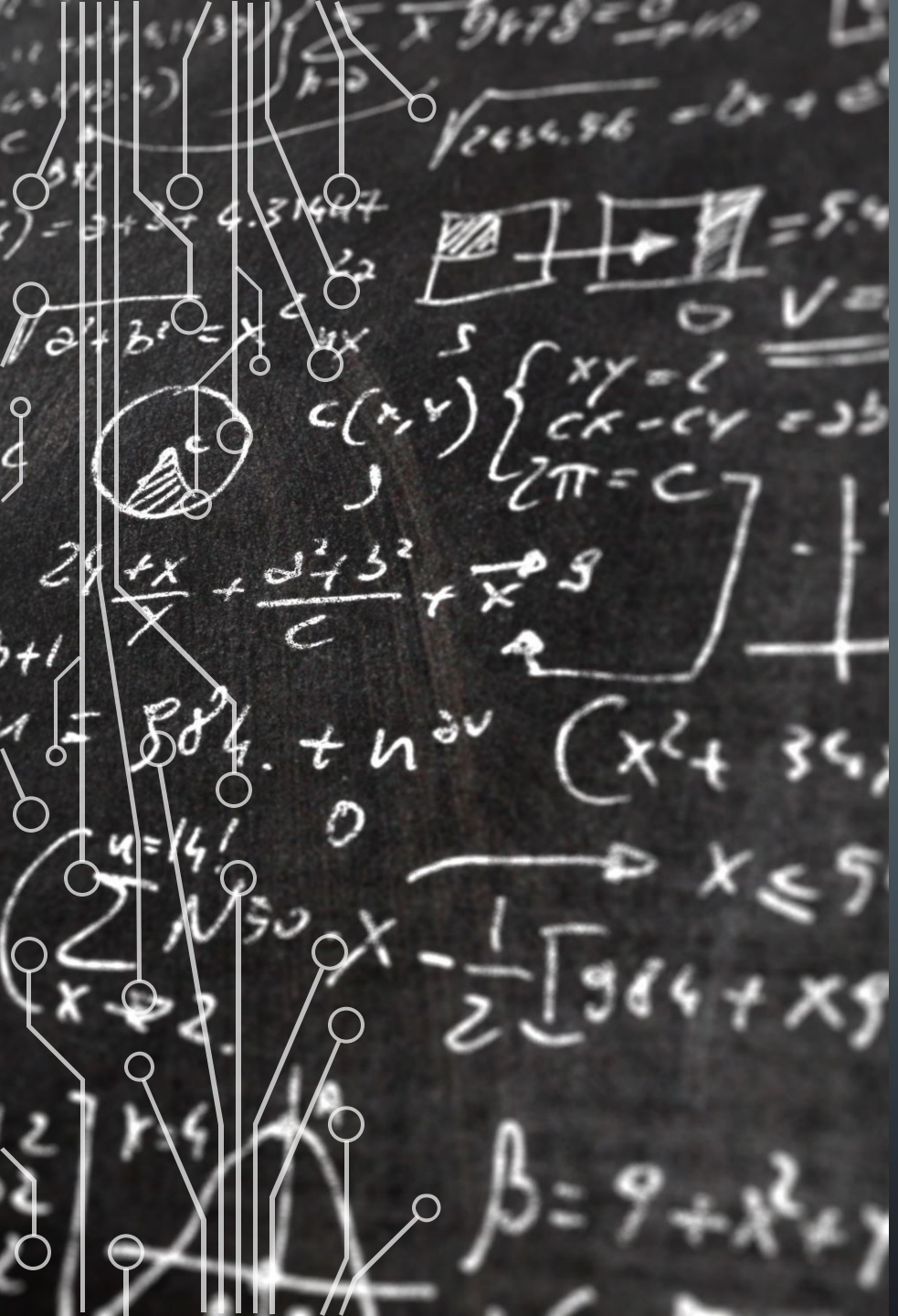
```
public static void main(String[] args) {
        String url = "jdbc:somedb:something";
        try(Connection c = DriverManager.getConnection(url); Statement s = c.createStatement(); ResultSet rs = s.executeQuery()){
                while(rs.next()){
                        System.out.println(rs.getString(1));
                }
        }
}
```

DATABASE CONNECTION

- JDBC URL consists of three parts:
  - jdbc:someVendorName://localhost:1234/databasename
- Two ways to get a connection, only one for OCP exam:
  - DriverManager > for exam!
  - DataSource > for daily life
- Method in DriverManager to get a Connection: getConnection(url) and getConnection(url, username, password)

DATABASE CONNECTION

- DriverManager class finds a Driver in a JAR in the classpath
- If DriverManager cannot find a JAR with a java.SQL.Driver implementation specified, it throws an SQLException.
- Old code: Class.forName(url) to load a class that implements Driver. Driver can then use the Driver, if it cannot be loaded it will throw a ClassNotFoundException instead of the SQLException. It is no longer necessary to use Class.forName since JDBC Driver version 4 and higher



GETTING A STATEMENT

- You get a Statement from a Connection instance
  - Statement s = connection.createStatement();
  - Statement s2 = connection.createStatement(ResultSet.ENUM_TYPE, ResultSet.ENUM_CONCURRENCY);

RESULTSET TYPE

- ResultSet Type enum

ResultSet Type enum	Backward	Latest data	Supported by most drivers
ResultSet.TYPE_FORWARD_ONLY	No	No	Yes
ResultSet.TYPE_SCROLL_INSENSITIVE	Yes	No	Yes
ResultSet.TYPE_SCROLL_SENSITIVE	Yes	Yes	No

RESULTSET CONCURRENCY MODE

- ResultSet Concurrency enum

ResultSet Type enum	Can read	Can update	Supported by all drivers
ResultSet.CONCUR_READ_ONLY	Yes	No	Yes
ResultSet.CONCUR_UPDATABLE	Yes	Yes	No

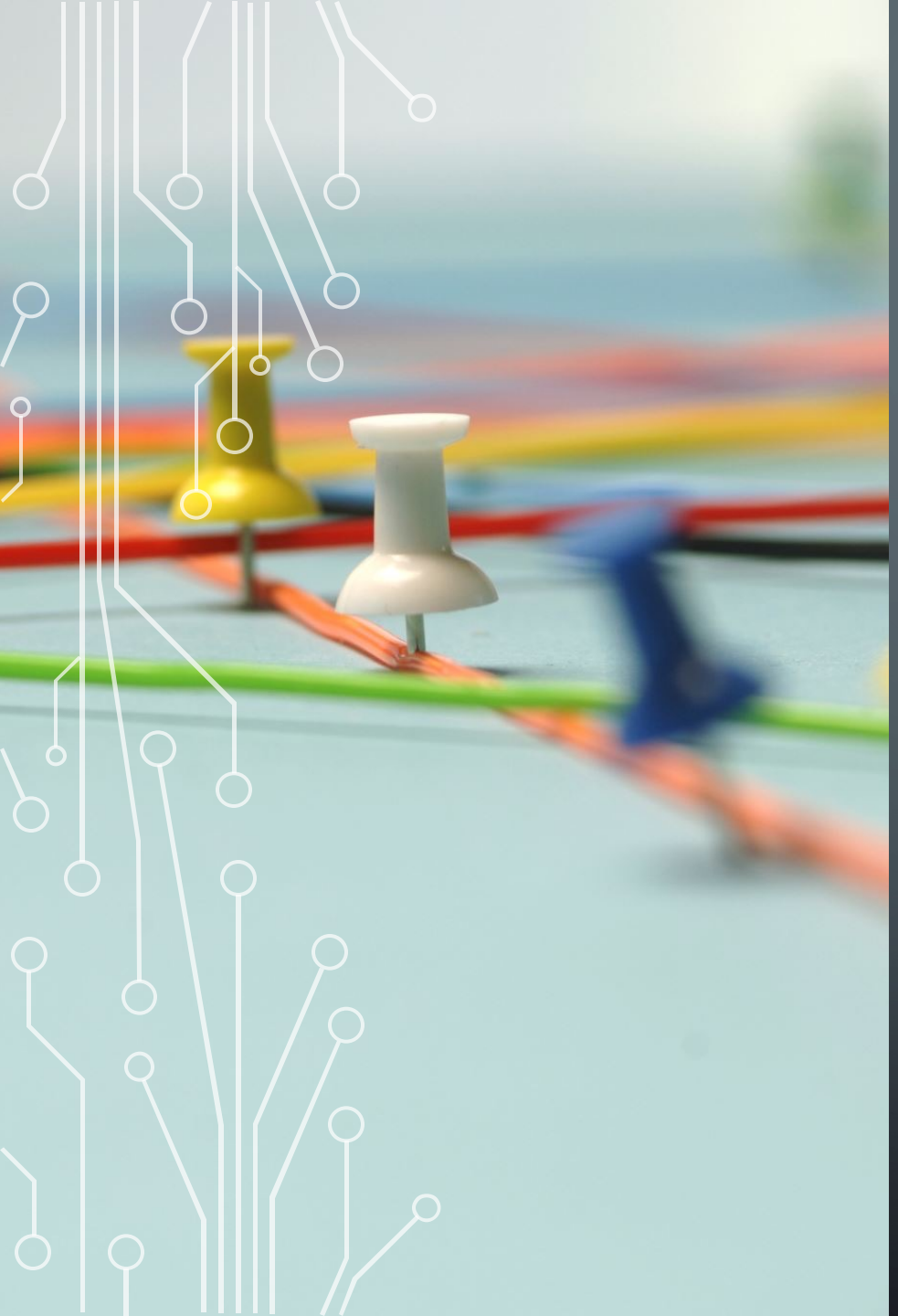
EXECUTING A STATEMENT

- Some SQL query, usually starting with SELECT, DELETE, INSERT or UPDATE
- Executed by calling executeUpdate() (also for DELETE and INSERT) and returns an int for nr of rows that were affected
- Executed by calling executeQuery() (for SELECT) and returns ResultSet
- Executed by calling execute() (for all) returns true if there would be a ResultSet

- Example:

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

int result = stmt.executeUpdate("INSERT into tablename values(1, "hi")); //returns nr of rows that were affected
```



RESULTSET - READING

- Has a cursor that points to the row of data that is selected
- Use loop to iterate over forward only ResultSet to save the results to a Map
- Various methods to read value of row for a column: getInt(), getString, getTimestamp, getEtcetera >> argument is either column name or column index (starts counting at 1)
- .next() to see whether the end of the ResultSet has been reached (returns false for no next)

WHAT'S WRONG WITH THIS CODE?

```
ResultSet rs = statement.executeQuery("some query here");
try {
  while (rs.next()) {
    int id = rs.getInt(0);
    String userName = rs.getString(1);
    Timestamp timeReg = rs.getTimestamp(2);
  }
} finally {
  rs.close();
}
```

WHAT'S WRONG WITH THIS CODE?

```
ResultSet rs = statement.executeQuery("some query here");
try {
    int id = rs.getInt("id");
} finally {
  rs.close();
}
```

WHAT'S WRONG WITH THIS CODE?

```
ResultSet rs = statement.executeUpdate("some query here");
try {
  while (rs.next()) {
    String name = rs.getString(1);
}
} finally {
  rs.close();
}
```

HOW TO DEAL WITH DATES

- getDate() returns a java.sql.Date object
- Convert this Date object to a LocalDate by calling .toLocalDate() on it

- getTime() returns a java.sql.Time object
- Surprisingly: convert this Time to a LocalTime by calling .toLocaltime on it

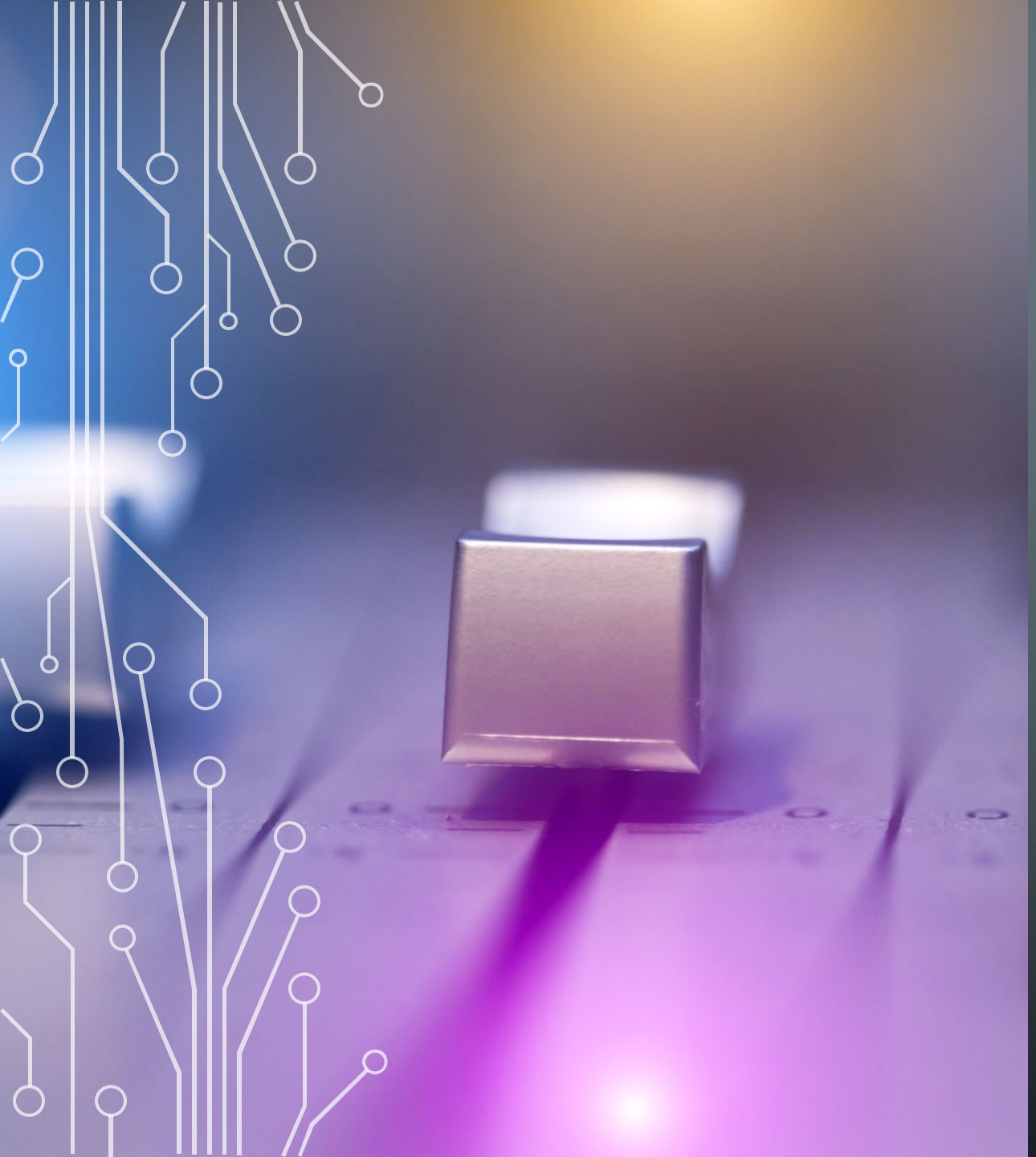
- getTimeStamp() returns a java.sql.Timestamp object
- Shockingly: Convert this to a LocalDateTime by calling .toLocalDateTime on it

HOW TO DEAL WITH OBJECTS

- getObject() on a ResultSet can return any object
- Check if it is an instance of whatever you want to cast it to
- Then cast it to that object

RESULTSET - SCROLLING

- Cursor of the ResultSet can be placed at any row
- .previous() method moves one row backwards
- Other methods that move the cursor:
  - first() > beginning of ResultSet, boolean return type for whether a row was found
  - last() > end of ResultSet, boolean return type for whether a row was found
  - beforeFirst() and afterLast() void return type, because always possible
  - absolute(rowNr) > moves the cursor to the specified row:
    - 0 same as beforeFirst
    - 1 same as first()
    - -1 same as last()
    - -2 one before last, etc
  - relative(nrOfRowsToMoveCursor) > negative number moves backward, positive number moves forward



CLOSING DATABASE SOURCE

- Not closing JDBC recources creates a resource leak, this is bad for the performance of your program
- Connection, Statement and ResultSet are autocloseable and can be used with a try-with-resources
- Also manually with .close(), but then you also need to catch SQLExceptions
- Closing Connection also closes Statement and ResultSet
- Closing Statement also closes ResultSet
- ResultSet is automatically closed when you run another query on a statement that is connected to a ResultSet

DEALING WITH EXCEPTIONS

- Exceptions need to be caught
- Several methods on SQLException:
  - getMessage > easy description of what went wrong
  - getSQLState > code of the problem that you can google, standardized for all databases by ISO/ANSI and Open Group (X/Open) (some databases have a few of their own though)
  - getErrorCode > code of the database implementation, google with code and the database type