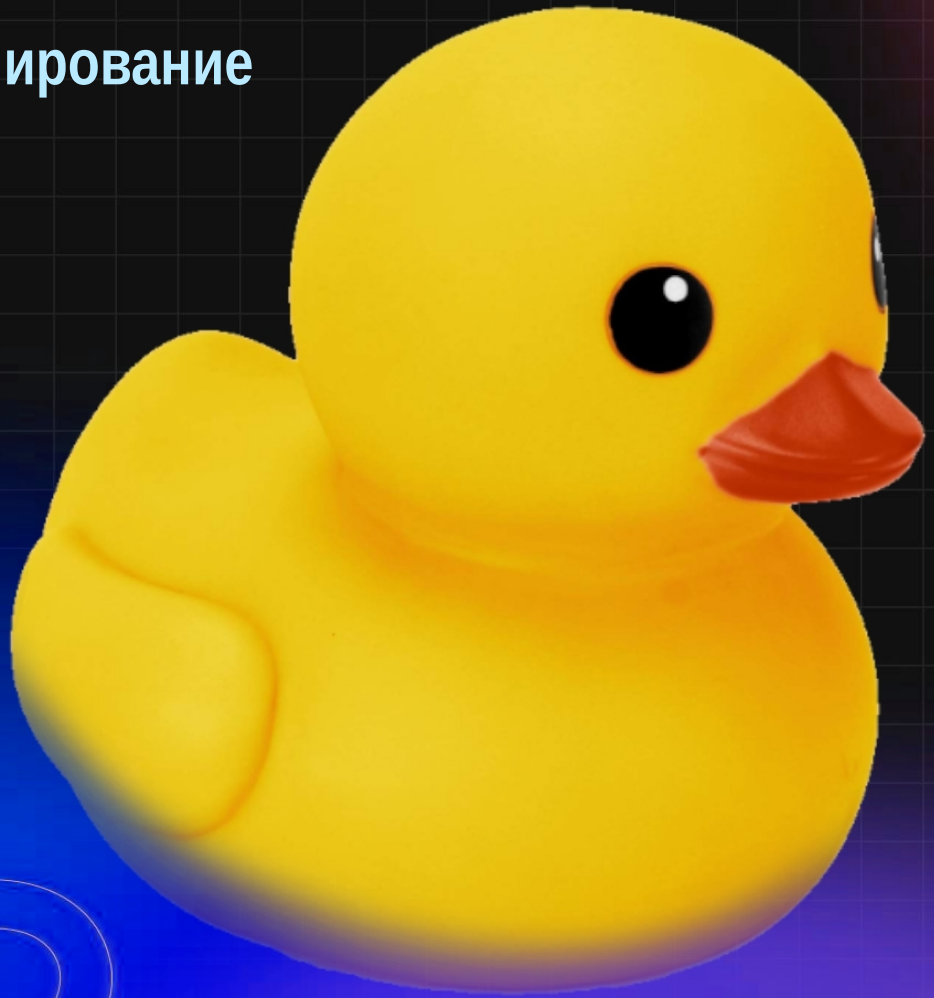
Программирование
2 семестр
2024



Взаимодействие с базами данных

Базы даннных

ИТМО

- БД — структурно организованные данные о предметной области, хранящиеся вместе с информацией о данных и их взаимосвязях.
- СУБД — вычислительная система для создания и использования баз данных.
- Реляционная БД — база данных, основанная на реляционной модели данных.

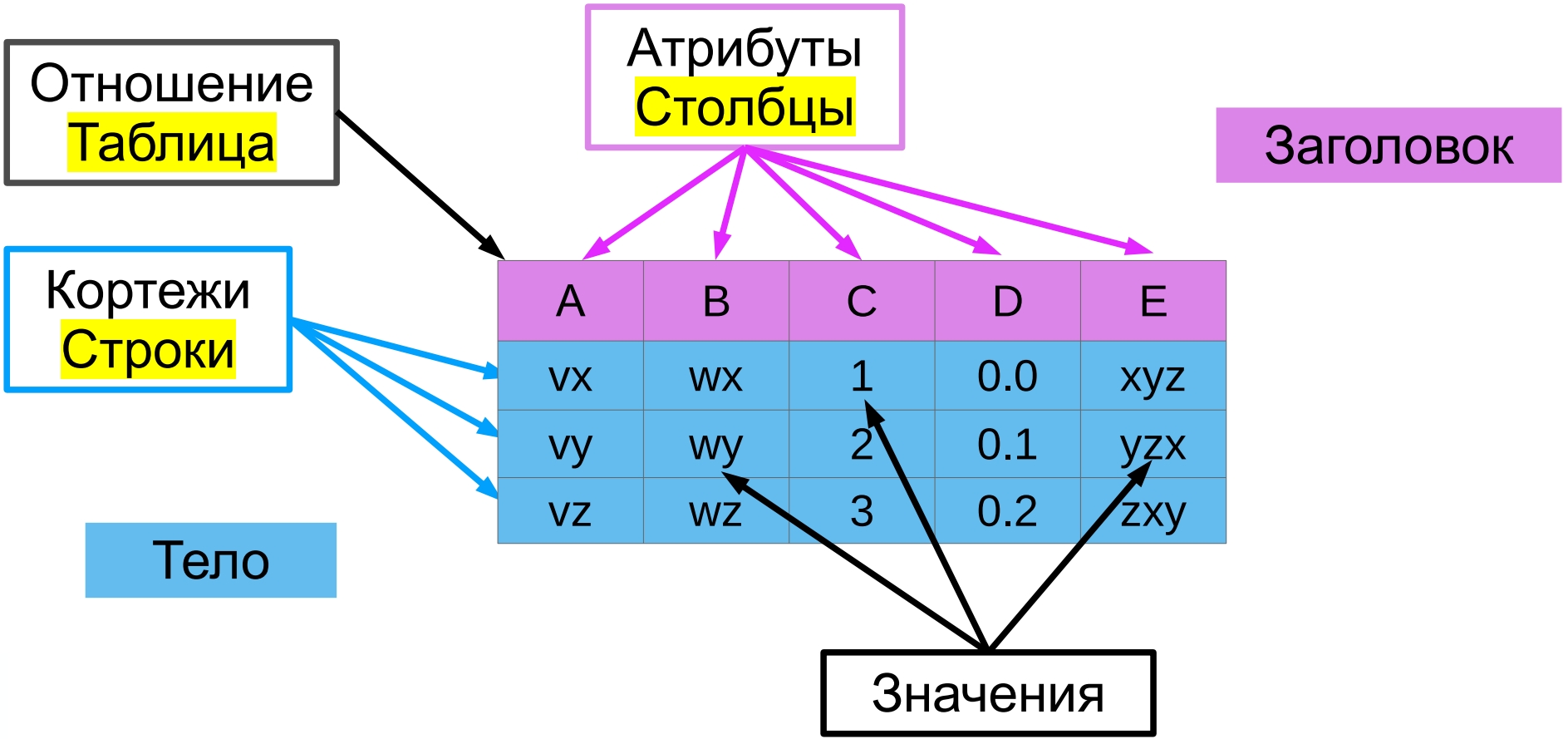
Java

Реляционная модель

ИТМО

- Отношение (relation) — структура данных, состоящая из заголовка и тела.
  - Заголовок отношения — множество атрибутов
  - Тело отношения — множество кортежей, содержащих значения атрибутов
- Свойства отношения
  - Каждый атрибут имеет тип, значения соответствуют типу
  - Атрибуты не повторяются, и их порядок не имеет значения
  - Кортежи не повторяются, и их порядок не имеет значения

Отношение и таблица



Пример базы данных

ИТМО

students

student_id	name	group
289001	Иванов Петр	P3110
289002	Петров Сидор	P3130
289999	Сидоров Иван	R3140

groups

group	faculty
P3110	ПИиКТ
P3130	ПИиКТ
R3140	СУиР

grades

student_id	course_id	grade
289001	1	A
289001	2	B
289002	1	E
289002	2	A
289999	1	C
288999	3	D

courses

course_id	name	semester	type
1	Программирование	1	З
2	ОПД	2	Э
3	Физика	2	Э

Java

Язык запросов

- SQL (Structured Query Language) — декларативный язык для описания, изменения и получения данных из реляционных баз данных.

CREATE TABLE

ИТМО

- DDL — Data definition language

```
CREATE TABLE weather (
    city      VARCHAR(80),
    temp_lo   INT,          -- low temperature
    temp_hi   INT,          -- high temperature
    prcp      REAL,         -- precipitation
    date      DATE
);

CREATE TABLE IF NOT EXISTS persons

DROP TABLE weather;
```

Java

Ограничения

ИТМО

- Ограничения
  - Типы данных
    - INT, SMALLINT, REAL, DOUBLE, CHAR(n), VARCHAR(n), DATE, TIME, ...
  - Возможные значения в столбце
    - NOT NULL, UNIQUE, CHECK (age >= 18)
  - Ключи
    - Первичный ключ
      - PRIMARY KEY (UNIQUE, NOT NULL)
    - Внешние ключи
      - FOREIGN KEY (REFERENCES)

Java

DML

- DML — Data manipulation language
  - `INSERT`
  - `UPDATE`
  - `DELETE`

INSERT

- Вставка данных в таблицу

```
INSERT INTO weather VALUES ('Oslo', 46, 50, 0.25, '2021-11-27');

INSERT INTO weather (date, city, temp_hi, temp_lo)
    VALUES ('2021-11-29', 'Helsinki', 54, 37);

COPY persons TO file;

COPY persons FROM file;
```

UPDATE

- Обновление данных в таблице

```
UPDATE weather
    SET temp_hi = temp_hi - 2,  temp_lo = temp_lo - 2
    WHERE date > '2021-11-28';
```

DELETE

- Удаление данных из таблицы

```
DELETE FROM weather WHERE city = 'Oslo';

DELETE FROM persons;
```

SELECT

ИТМО

- Выборка данных - запрос

```
SELECT * FROM students;
SELECT name, group FROM students;
SELECT * FROM students WHERE group = 'P3110';
SELECT * FROM students ORDER BY name;
SELECT DISTINCT type FROM courses;
SELECT COUNT(*) FROM students;
```

Java

JOIN

ИТМО

- Соединение таблиц

```
SELECT name, faculty FROM students JOIN groups
    ON students.group = groups.group;  --  USING(group);
```

students

student_id	name	group
289001	Иванов Петр	P3110
289002	Петров Сидор	P3130
289999	Сидоров Иван	R3140

groups

group	faculty
P3110	ПИиКТ
P3130	ПИиКТ
R3140	СУиР

JOIN

student_id	name	group	faculty
289001	Иванов Петр	P3110	ПИиКТ
289002	Петров Сидор	P3130	ПИиКТ
289999	Сидоров Иван	R3140	СУиР

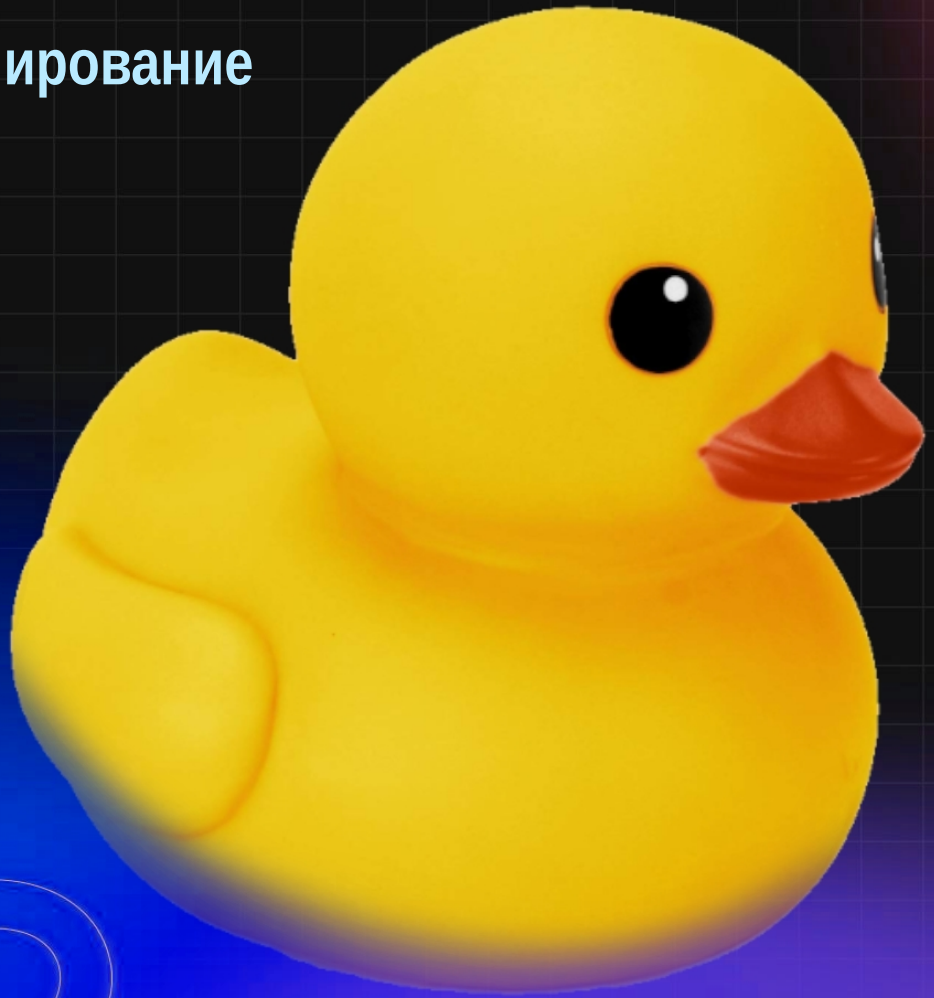
СУБД PostgreSQL

ИТМО

- helios
  - `psql -h pg studs`
  - пароль в файле `.pgpass`
    - `host:port:dbname:login:PaSsw0rD`
  - `драйвер - /usr/local/share/java/classes/postgresql.jar`
- https://postgresql.org
  - \? - помощь по командам psql
  - \h - помощь по командам SQL

Java

Программирование
2 семестр
2024



ИТМО

Основы JDBC

Работа с БД

ИТМО

- Много разных СУБД
  - API для каждой базы отдельно
  - единый интерфейс работы с базами + драйвер для конкретной базы
- Реализации
  - ODBC — Open Database Connectivity
  - JDBC — Java Database Connectivity

Java

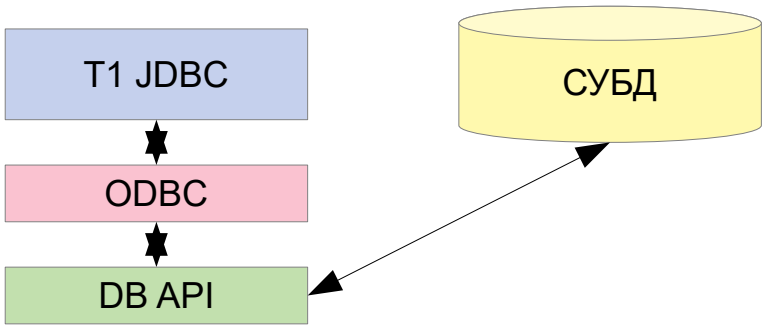
JDBC

ИТМО

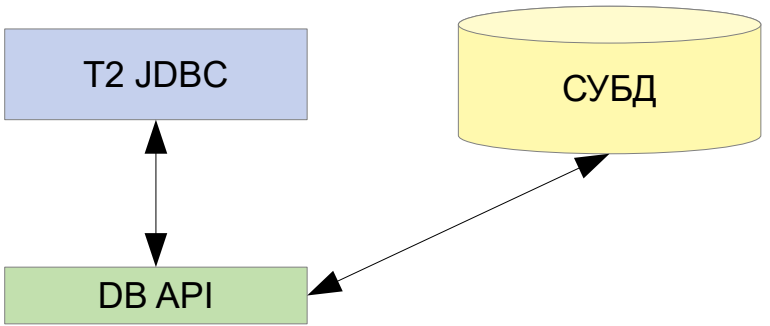
- JDBC — Java DataBase Connectivity
- JDBC API — высокоуровневый интерфейс для доступа к данным
- JDBC Driver API — низкоуровневый интерфейс для создания драйверов
- Пакеты java.sql (Core) и javax.sql (Extension)
- Стандарт взаимодействия с СУБД

Типы взаимодействия

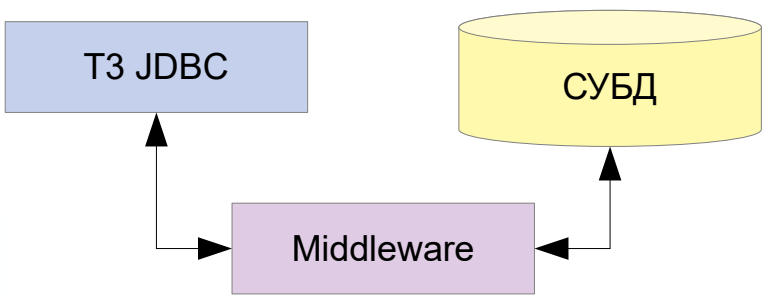
- Тип 1 — мост ODBC



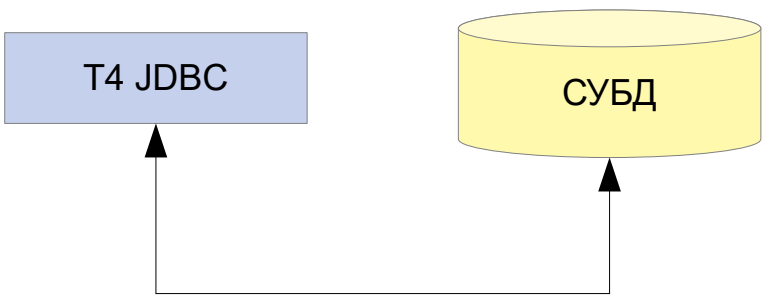
- Тип 2 — DB API



- Тип 3 — middleware



- Тип 4 — pure Java



Процесс взаимодействия

```
Connection connection = DriverManager.getConnection(…);

  Statement statement = connection.createStatement();

    ResultSet resultSet = statement.executeQuery("SELECT …");

    while (resultSet.next()) {
    // получение и обработка данных
    }

    resultSet.close();
  statement.close();
connection.close();
```

Интерфейс Driver

ИТМО

- java.sql.Driver
  - Отвечает за связь с БД
  - Метод Connection connect(String url, Properties info)
  - Используется для написания драйверов для СУБД

Java

Класс DriverManager

- Управляет списком драйверов
- Загрузка драйвера
  - Class.forName()
  - jdbc.drivers=
- Неявная загрузка с помощью ServiceLoader
  - META-INF/services/java.sql.Driver

Класс DriverManager

- Метод Connection getConnection(String url, ...)
  - url = jdbc:*protocol*://*host:port/database*
    - jdbc:postgresql://db:5432/studs
  - getConnection(String url, Properties info)
    - Properties info = new Properties();
    - info.load(new FileInputStream("db.cfg");

      файл db.cfg

      ```
      user = s999999
      password = sss999
      ```
  - getConnection(Stiring url, String name, String pass)

Интерфейс Connection

- Абстракция соединения (сессия)
  - методы:
    - Statement createStatement()
    - PreparedStatement prepareStatement(String sql)
    - CallableStatement prepareCall(String sql)
    - DatabaseMetaData getMetaData()

Семейство интерфейсов Statement

ИТМО

- Statement
  - Статический SQL-запрос
  - Запрос передается как параметр при выполнении

```
String query = "SELECT * FROM table WHERE id = 15";
Statement st = connection.createStatement();
st.executeQuery(query);
```

Семейство интерфейсов Statement

ИТМО

- **PreparedStatement** extends Statement
  - **Динамический запрос** с подстановкой
  - Запрос передается как параметр при создании

```
String query = "SELECT * FROM table WHERE id = ?";
PreparedStatement ps = connection.prepareStatement(query);
ps.setInt(1, 15); // 1 — номер параметра, 15 — значение
SELECT * FROM table WHERE id = 15
```

- Предотвращает SQL-инъекции

Семейство интерфейсов Statement

ИТМО

- CallableStatement extends PreparedStatement
  - Вызов хранимой процедуры

    SQL: CREATE PROCEDURE

```
cs = prepareCall("CALL getResult (?)");
cs.setInt(1, 15);
cs.registerOutParameter(1, Types.INTEGER);
...
int result = cs.getInt(1);
CALL getResult(15);
```

Выполнение запросов - executeQuery

- ResultSet executeQuery(String sql) // Statement
- ResultSet executeQuery() // PreparedStatement (Callable)
  - для выполнения SELECT
  - возвращает ResultSet

```
ResultSet resultSet = statement.executeQuery();
```

Выполнение запросов - executeUpdate

- int executeUpdate(String sql) // Statement
- int executeUpdate() // PreparedStatement (Callable)
  - для выполнения INSERT, UPDATE, DELETE + DDL
  - возвращает количество измененных строк
  - Для запросов DDL возвращает 0

```
int updateCount = statement.executeQuery();
```

Выполнение запросов - execute

- boolean execute(String sql) // Statement
- boolean execute() // PreparedStatement (Callable)
  - для выполнения любых запросов

```
if (statement.execute()) {
    ResultSet resultSet = statement.getResultSet();
} else {
    int updateCount = statement.getUpdateCount();
}
```

Транзакции

- Connection
  - setAutoCommit(true/false)
  - commit()
  - rollback()
  - setSavepoint()
- Statement
  - addBatch(String sql)
  - clearBatch()
  - executeBatch()

Интерфейс ResultSet

- Получение данных из ResultSet

```
ResultSet rs = preparedStatement.executeQuery();

while (rs.next()) {
    String name = rs.getString(1); // по номеру столбца
    int id = rs.getInt("id");      // по имени столбца
}
```

Настройка типа ResultSet

- ResultSet
  - Connection.**createStatement**(sql, **type, concurrency, holdability**)
  - ResultSetType
    - **TYPE_FORWARD_ONLY**
    - TYPE_SCROLL_INSENSITIVE
    - TYPE_SCROLL_SENSITIVE
  - ResultSetConcurrency,
    - **CONCUR_READ_ONLY**
    - CONCUR_UPDATABLE
  - ResultSetHoldability
    - HOLD_CURSORS_OVER_COMMIT
    - CLOSE_CURSORS_AT_COMMIT

Интерфейс ResultSet

- Навигация
  - `next()`
  - `previous()`
  - `first()`
  - `last()`
  - `beforeFirst()`
  - `afterLast()`
  - `relative(int row)`
  - `absolute(int row)`
  - `moveToInsertRow()`

Интерфейс ResultSet

- Получение данных
  - `getString(int)`
  - `getString(String)`
  - `getInt(int)`
  - `getInt(String)`
  - `getBoolean`
  - `getLong`
  - `getDouble`
  - `getArray (SQL Array)`
  - `getDate`
  - `getTimestamp`
  - `getReader`
  - `...`

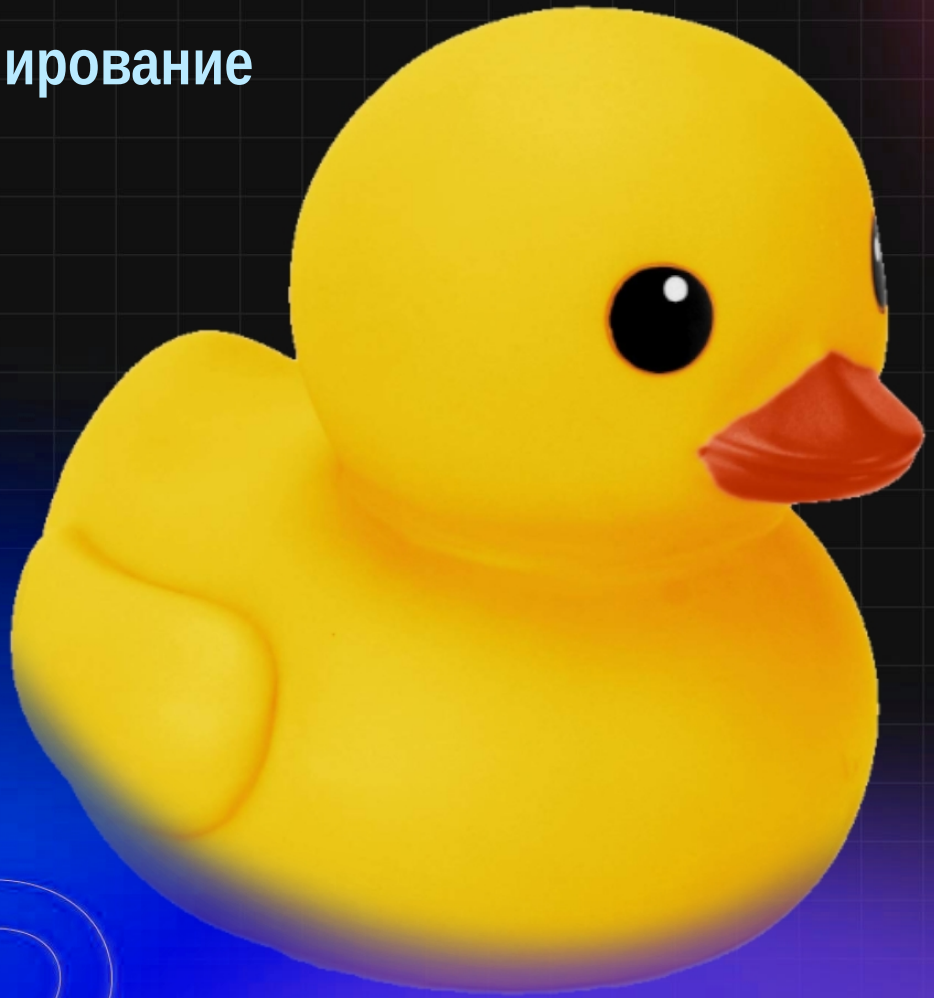
Интерфейс ResultSet

- Обновление строк
  - `updateInt(String, int)`
  - `updateInt(int, int)`
  - `updateString(String, String)`
  - `updateString(int, String)`
  - `updateRow()`
- Добавление строк
  - `moveToInsertRow()`
  - `updateInt(String, int)`
  - `insertRow()`

Метаданные

- ResultSetMetaData ResultSet.getMetaData()
  - `getTableName()`
  - `getColumnCount()`
  - `getColumnName(int n)`
  - `getColumnType(int n)`
- DatabaseMetaData Connection.getMetaData()
  - `getCatalogs()`
  - `getTables()`
  - `getSchemas()`

Программирование
2 семестр
2024



ИТМО

Расширения JDBC

Интерфейс DataSource

- javax.sql.DataSource
  - Позволяет получить соединение с БД
  - org.postgresql.ds.PGSimpleDataSource
- ConnectionPoolDataSource
  - Поддержка пула соединений
  - org.postgresql.ds.PGPoolingDataSource
- XADataSource
  - Поддержка распределенных транзакций

Простой источник данных

```
import org.postgresql.ds.PGSimpleDataSource;

PGSimpleDataSource ds = new PGSimpleDataSource();

ds.setServerName(...);
ds.setDatabaseName(...);
ds.setUser(...);
ds.setPassword(...);

Connection conn = ds.getConnection();
```

Поддержка JNDI

ИТМО

```
import javax.naming.*;

Context context = new InitialContext();
DataSource ds = ...          // подготовка DataSource
context.bind("testDB", ds); // привязка к контексту

// получение DataSource из контекста
Context context = new InitialContext();
DataSource ds = (DataSource)context.lookup("testDB");
```

Интерфейс RowSet

- javax.rowset.*
- Единый интерфейс для всех операций

- RowSet extends ResultSet
  - setUrl(),
  - setUsername(),
  - setPassword(),
  - setCommand("Select * from ...");
  - execute()
  - next()
  - getXXX()

```
var factory = RowSetProvider.newFactory()
var rowset = factory.createJdbcRowSet();
```

Фабрики RowSet

- RowSetFactory factory = RowSetProvider.newFactory();
  - factory.createJdbcRowSet();
  - factory.createCachedRowSet();
  - factory.createWebRowSet();
  - factory.createFilteredRowSet();
  - factory.createJoinRowSet();

Разновидности RowSet. JdbcRowSet

- JdbcRowSet — простая разновидность RowSet
  - Поддерживает соединение с базой данных
  - По умолчанию:
    - Тип: SCROLL_INSENSITIVE
    - Конкурентность: CONCUR_UPDATABLE
    - Включено экранирование спецсимволов

Разновидности RowSet. JdbcRowSet

- JdbcRowSet — простая разновидность RowSet

```
JdbcRowSet rs = factory.createJdbcRowSet();
rs.setUrl(""); …
rs.setCommand("Select * from users");
rs.execute();
rs.last();
rs.getInt("id");
rs.updateString("name", "Pupkin");
rs.UpdateRow();
```

Разновидности RowSet. CachedRowSet

- CachedRowSet
  - Результат запроса может кэшироваться
  - Синхронизация с базой
  - Разрешение конфликтов
  - acceptChanges()

Разновидности RowSet. CachedRowSet

- CachedRowSet

```
CachedRowSet rs = factory.createCachedRowSet();
rs.setUrl(""); …
rs.setCommand("Select * from users");
rs.execute();
rs.last();
rs.getInt("id");
rs.updateString("name", "Pupkin");
rs.acceptChanges();
```

Разновидности RowSet. WebRowSet

- WebRowSet
  - Может записывать и читать результат в виде XML
  - writeXML()
  - readXML()

Разновидности RowSet. WebRowSet

- WebRowSet

```
WebRowSet rs = factory.createWebRowSet();
rs.setUrl(""); …
rs.setCommand("Select * from users");
rs.execute();
rs.writeXML(new FileWriter("data.xml"));
```

Разновидности RowSet. FilteredRowSet

- FilteredRowSet
  - Фильтрация строк (аналог WHERE)
  - setFilter(Predicate p)
- Predicate
  - boolean evaluate(Object value, int/String column)
  - boolean evaluate(RowSet rowset)

Разновидности RowSet. FilteredRowSet

- FilteredRowSet

```
FilteredRowSet rs = factory.createFilteredRowSet();
rs.setUrl(""); …
rs.setCommand("Select * from users");
rs.execute();
Predicate filter = new Predicate();
rs.setFilter(filter);
```

Разновидности RowSet. JoinRowSet

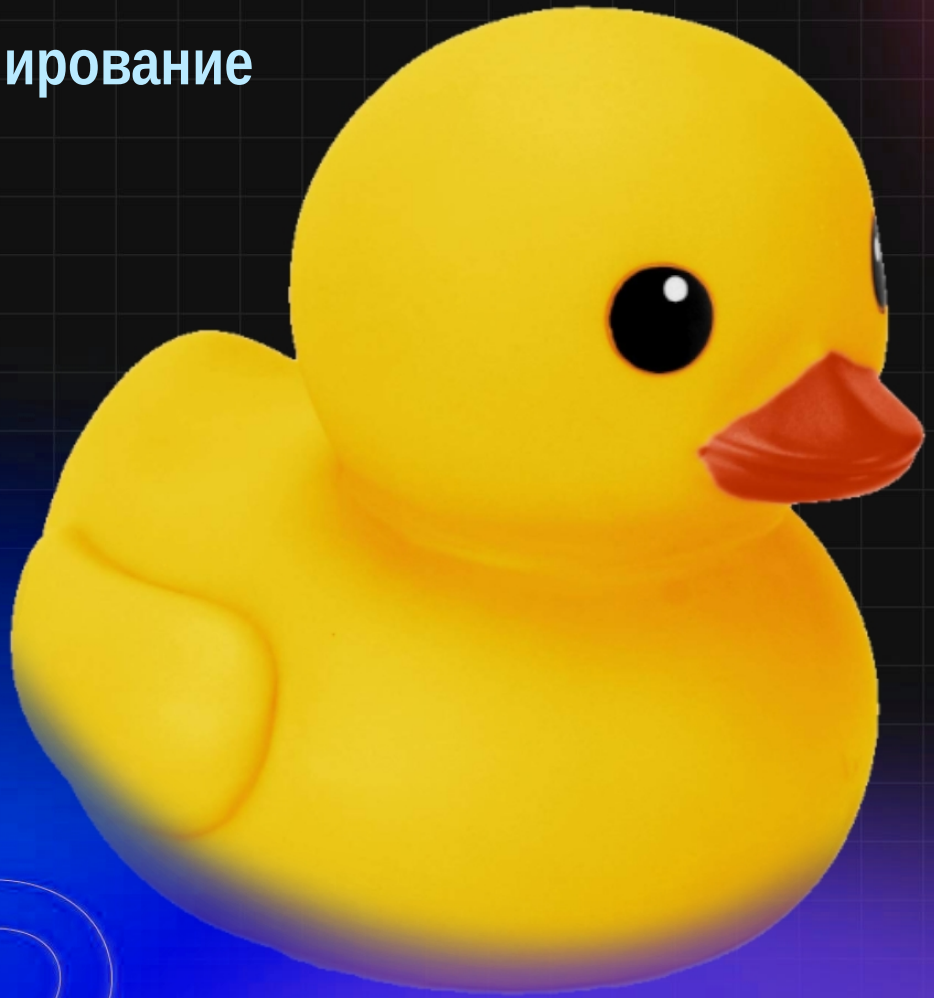
- JoinRowSet
  - Соединение нескольких результатов (JOIN)
  - addRowSet()
  - setJoinType()
  - toCachedRowSet()

Разновидности RowSet. JoinRowSet

- JoinRowSet

```
JoinRowSet js = factory.createJoinRowSet();
CachedRowSet users; // "Select * from users"
CachedRowSet groups; // "Select * from groups"
users.setMatchColumn("uid");
groups.setMatchColumn("uid");
js.addRowSet(users);
js.addRowSet(groups);
```

Программирование
2 семестр
2024



ИТМО

Провайдеры служб

Провайдеры служб

- Служба или сервис
- Разные реалиации
- Нужен механизм поиска и загрузки служб

Шпаргалка на экзамен

ИТМО

```
class CheatSheet {
  public String getAnswer(String question) {
    Map source =
    String answer = source.get(question);
    if (answer != null) return answer;
    else return "Epic fail";
  }
}
```

Java

Нужен источник знаний

```
class CheatSheet {
  public String getAnswer(String question) {
    Map source = knowledge.getSource();
    String answer = source.get(question);
    if (answer != null) return answer;
    else return "Epic fail";
  }
}
interface Knowledge {
  Map<String,String> getSource();
}
```

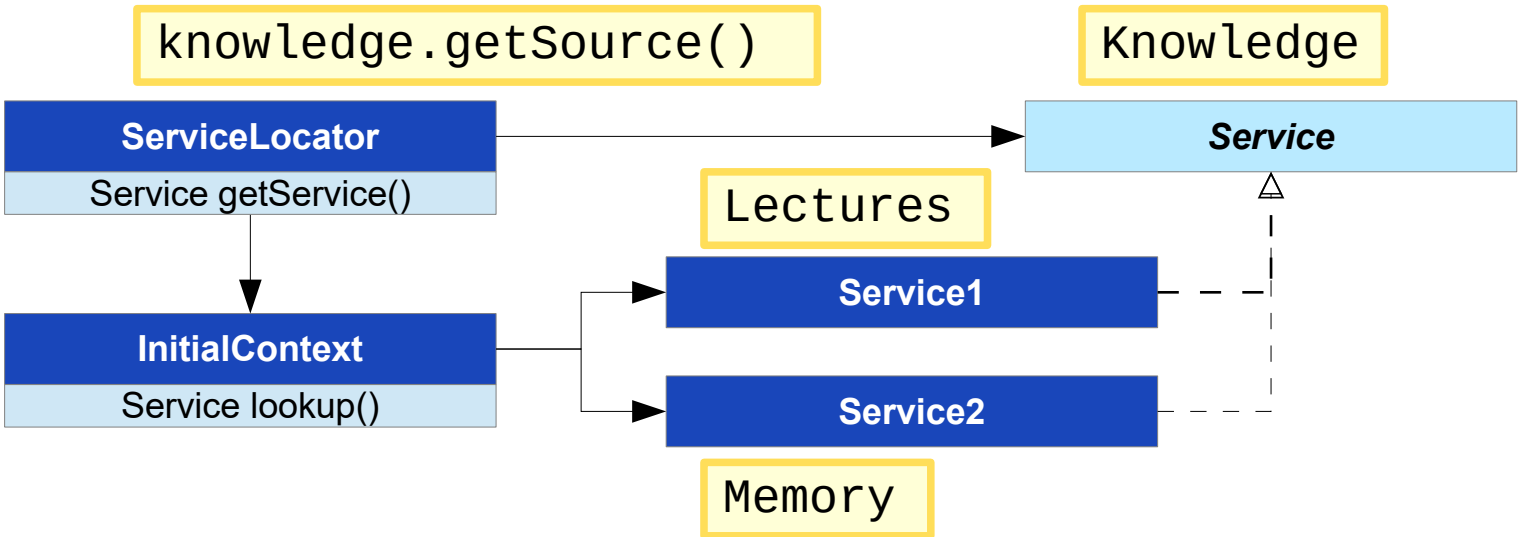
Различные источники

```
class CheatSheet {
  public String getAnswer(String question) {
    Map source = knowledge.getSource();
    String answer = source.get(question);
    if (answer != null) return answer;
    else return "Epic fail";
  }
}
interface Knowledge {
  Map<String,String> getSource();
}
```

```
class Memory implements Knowledge
class Lecture implements Knowledge
class CallFriend implements Knowledge
class Miracle implements Knowledge
```

Как передать объект?

- Шаблон ServiceLocator



Реализация в Java

- Класс java.util.ServiceLoader<Service>
  - static ServiceLoader<Service> load(Service.class)
  - Iterator<Service> iterator()

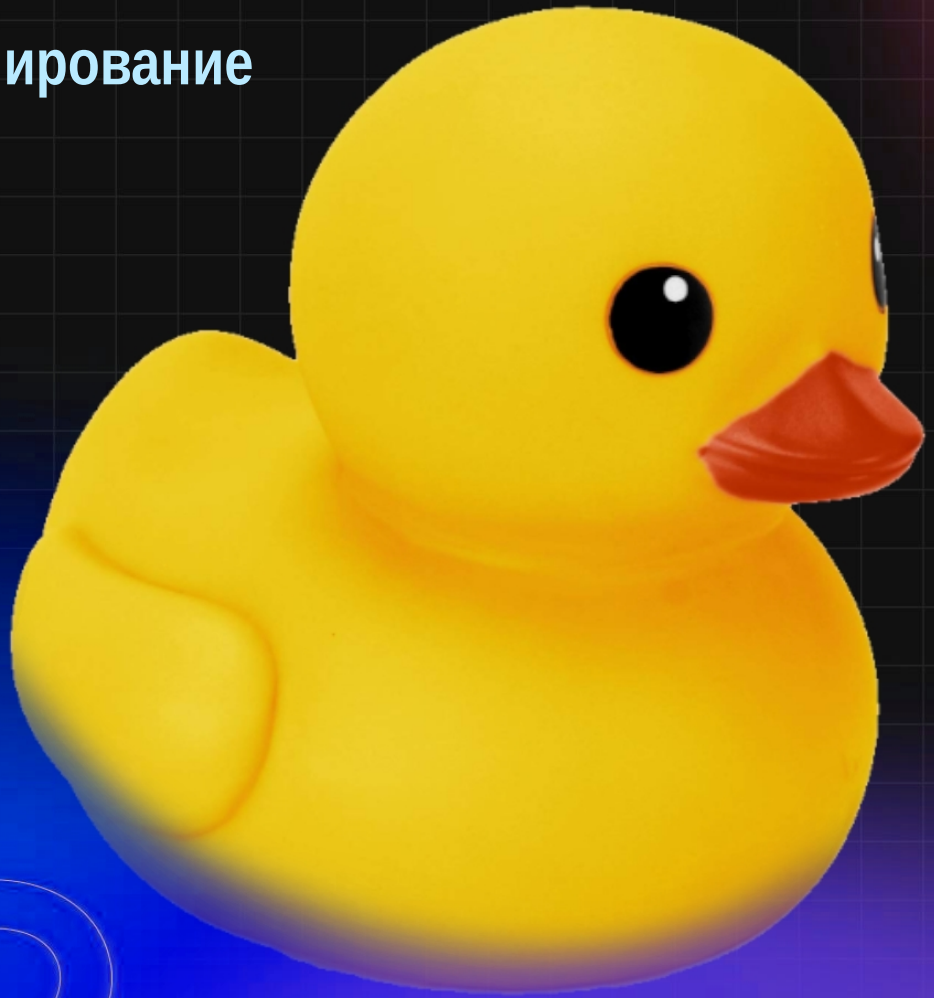
Реализация в Java

- **interface** spi.Service
  - public void execute();
- **public class** spi.DefaultServiceImpl {
  - **public DefaultServiceImpl() { ... }**
  - public void execute() { ... }
- service.jar
  - **каталог META-INF/services/**
    - файл **spi.Service**
      - **spi.DefaultServiceImpl**

Провайдеры

- java.nio.file.spi.FileSystemProvider
- java.nio.channels.spi.AsynchronousChannelProvider
- java.nio.channels.spi.SelectorProvider
- java.nio.charset.spi.CharsetProvider
- java.text.spi.DateTimeFormatProvider
- java.text.spi.NumberFormatProvider
- java.util.spi.CalendarDataProvider
- java.sql.DriverManager

Программирование
2 семестр
2024



ИТМО

Модули

Java до модулей - классы, пакеты, JARы

ИТМО

- Классы
  - public class - доступен всем
  - объединяются в пакеты (доступ внутри пакета)
  - упаковываются в JAR-архивы
- Пакеты ≈ каталоги
  - пакет и JAR - разные сущности
  - файл package-info.java / .class

Java до модулей - JAR-hell

- classpath
  - просмотр всех путей, загрузка первого найденного класса
- конфликт версий
  - не та версия класса, классы разных версий
- кастомные загрузчики классов
- нет зависимостей на уровне JVM
- JRE - Java Runtime Environment
  - rt.jar - содержит всю стандартную библиотеку

Модули

- Явные зависимости (requires и exports)
- модульная библиотека
  - базовый модуль java.base
  - платформенные модули (все остальные)
  - новый формат артефактов - jmod
- jlink - для сборки кастомного образа
  - rt.jar удален в Java 11
- модульный JAR (+module-info.class)

Модули - компиляция и запуск

- Декларация модуля - module-info.java / .class

```
module my.mod {
    requires java.base;
    requires java.sql;
    exports my.package;
}
```

- Опции javac и java

```
javac --module-path mods:libs -d ./out

java --module-path mods:libs --module my.mod
```

Модули - зависимости

- module m0 - наш модуль
- **requires** m1 - m0 требует m1 при компиляции и выполнении
- **requires transitive** m2 - m0 требует m2 и модули, зависящие от m0, тоже требуют m2
- **requires static** m3 - m0 требует m3 при компиляции, но не обязательно при выполнении

Модули - доступ

- module m0 - наш модуль
- **exports** p1 - экспортирует пакет p1 всем (доступ во время компиляции к открытым сущностям пакета)
- **exports** p1 to m1 - доступ только для модуля m1
- **opens** p2 - открывает пакет p2 всем (доступ с помощью рефлексии во время выполнения ко всем сущностям)
- **opens** p2 **to** m2 - открыт только для модуля m2

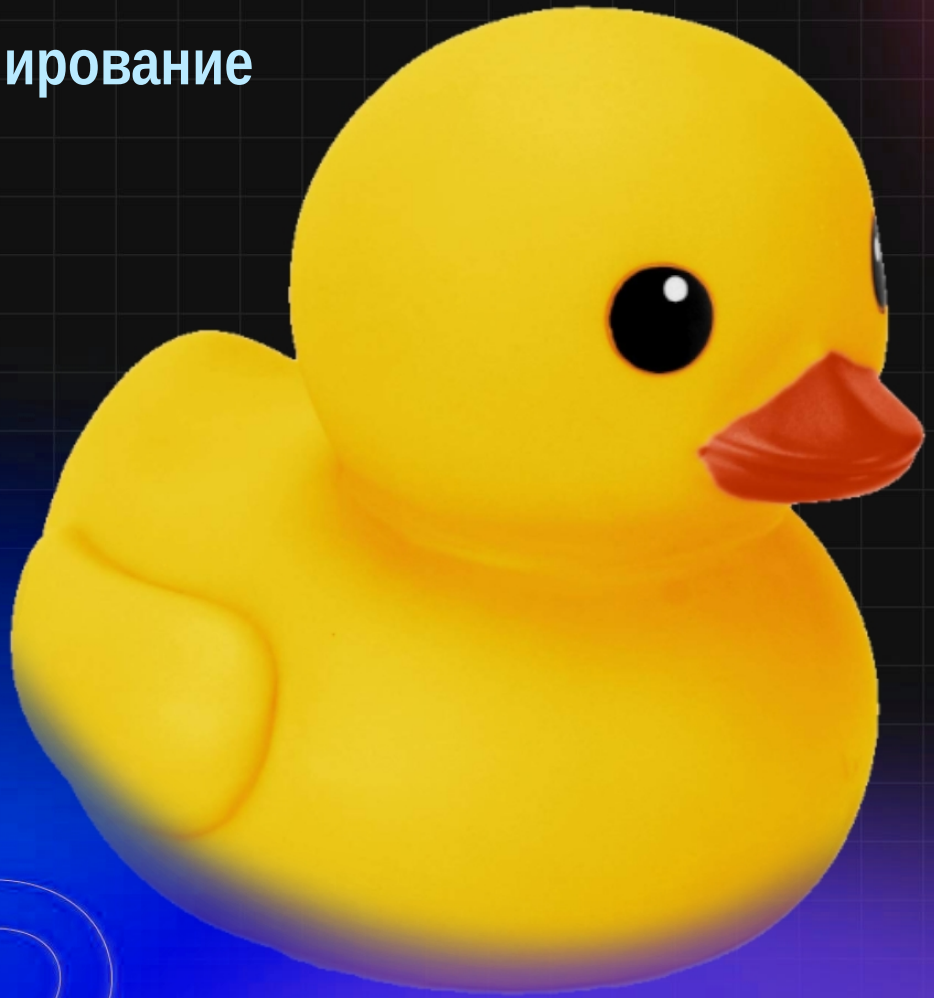
Модули - сервисы

- module m0 - наш модуль
- **uses** i1 - использует сервис с интерфейсом i1 (может загружать его с помощью ServiceLoader)
- **provides** i2 with c1, c2 - предоставляет интерфейс сервиса i2, который реализуют конкретные классы c1 и c2

Модули - совместимость

- Безымянный модуль
  - автоэкспорт всех своих пакетов
  - имеет доступ ко всем другим модулям
  - используется classpath

Программирование
2 семестр
2024



ИТМО

Безопасное хранение паролей

Хранение пароля в базе

- CREATE TABLE users (varchar name, varchar password);

Пароль нельзя хранить в открытом виде

- Доступ к базе
- Одинаковые пароли
- Резервное копирование
- SQL-инъекции и другие методы

SQL-инъекции

ИТМО

```
CREATE TABLE users (varchar name, varchar password);
CREATE TABLE cats (varchar cat, varchar owner);
q1 = "SELECT * FROM users WHERE name = '%s' AND password = '%s';";
q2 = "SELECT cat, owner FROM cats WHERE owner = '%s';";
username = Scanner.readLine(); password = Scanner.readLine();
statement.executeQuery(String.format(q1, username, password));
statement.executeQuery(String.format(q2, username));
```

(C) https://simonscat.com

SQL-инъекции

ИТМО

```
CREATE TABLE users (varchar name, varchar password);
CREATE TABLE cats (varchar cat, varchar owner);
q1 = "SELECT * FROM users WHERE name = '%s' AND password = '%s';";
q2 = "SELECT cat, owner FROM cats WHERE owner = '%s';";
```

Name: **Simon** Password: **123456**

```
SELECT * FROM users WHERE name = 'Simon' AND password = '123456';
SELECT cat FROM cats WHERE owner = 'Simon';
```

(C) https://simonscat.com

SQL-инъекции

```
CREATE TABLE users (varchar name, varchar password);
CREATE TABLE cats (varchar cat, varchar owner);
q1 = "SELECT * FROM users WHERE name = '%s' AND password = '%s';";
q2 = "SELECT cat, owner FROM cats WHERE owner = '%s';";
```

Name: `Simon';--` Password: `1`

```
SELECT * FROM users WHERE name = 'Simon';--' AND password = '1';
Login OK!
```

SQL-инъекции

```
CREATE TABLE users (varchar name, varchar password);
CREATE TABLE cats (varchar cat, varchar owner);
q1 = "SELECT * FROM users WHERE name = '%s' AND password = '%s';";
q2 = "SELECT cat, owner FROM cats WHERE owner = '%s';";
```

Name: `' OR ''='` Password: `' OR ''='`

```
SELECT * FROM users WHERE name = '' OR ''='' AND password = '' OR ''=''
Login OK!
```

(C) https://simonscat.com

SQL-инъекции

```
CREATE TABLE users (varchar name, varchar password);
CREATE TABLE cats (varchar cat, varchar owner);
q1 = "SELECT * FROM users WHERE name = '%s' AND password = '%s';";
q2 = "SELECT cat, owner FROM cats WHERE owner = '%s';";
```

Name: `' UNION SELECT name,password FROM users WHERE name<>'`

```
SELECT cat, owner FROM cats WHERE owner = ''
UNION SELECT name,password FROM users WHERE name<>'';
simon           123456
admin           @#p1nk_m0nkey$%
```

(C) https://simonscat.com

Лучше хранить хеш

ИТМО

- Вместо пароля — хеш пароля
- Функция необратимая — но можно подобрать
- Минимальное число коллизий — тоже можно подобрать
- MD2, MD5, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512
- md5("hello") = 5d41402abc4b2a76b9719d911017c592
- Словарные атаки

  https://md5.gromweb.com/

  http://reversemd5.com/

Java

Добавим к хешу соль

- Вместо пароля — хеш пароля + соль
- Соль — некая случайная последовательность

md5("hello" + "$Ins50D") = fcac10c41f9f67090e450161db4bca5a

md5("hello" + "HOxc3@") = a893f9063e4314635e05f8c46b547f7a

- Храним пароль + соль + алгоритм + сложность
- Все равно — "hello", "12345" — словарные атаки

Добавим перец

- В базе храним хеш пароля + соль
- Перец храним в приложении (отдельно)
- md5("hello" + "kFz<Q%ps" + "$Ins50D")

Пример кода

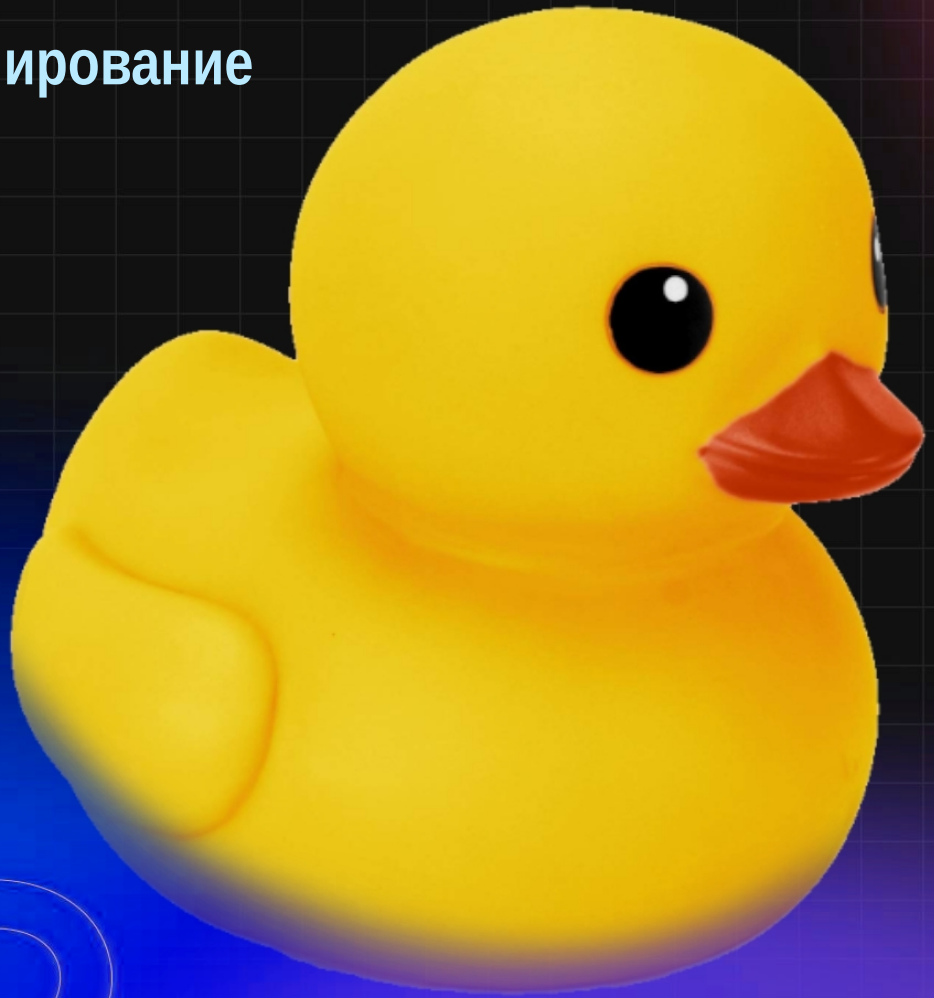
```
  java.security.MessageDigest
MessageDigest md = MessageDigest.getInstance("MD5");
String user   = Console.readLine()
String passwd = Console.readPassword();
String salt   = getRandomString();
String pepper = "*63&^mVLC(#"
byte[] hash = md.digest(
      (passwd + pepper + salt).getBytes("UTF-8"));

insert into users (user, salt, hash);
```

Исходящие пароли

- Входящие пароли - хеш, соль
- Исходящие пароли - отдельно от кода!
  - Запрашивать при старте
  - Конфигурационный файл с ограничением прав
  - Файлы свойств (Properties)
  - Переменная окружения
  - $HOME/.pgpass

Программирование
2 семестр
2024



ИТМО

Дата и время
Date/Time API

Использование даты и времени

ИТМО

- Дата и время — 2 варианта представления:
  - Человеческое время — часы, минуты, дни, недели, месяцы
  - Машинное время — миллисекунды от точки отсчета
    - EPOCH - 1 января 1970 года, 00:00:00 (Java, UNIX)
    - Windows - 1 января 1601 года
  - Y2K problem (2000)
  - Y 2038 problem
    - 19-01-2038 03:14:07 + 1 sec = 13-12-1901 20:45:52

Java

Традиционные классы. Date

ИТМО

- Date 1.0
  - все действия с датой
  - человеческое и машинное представление
  - форматирование даты
- Конструкторы
  - Date
  - Date(long)

- Date 1.1
  - только момент времени
  - почти все методы - deprecated
- Методы
  - long getTime()
  - boolean after(Date)
  - boolean before(Date)

Java

TimeZone

ИТМО

- Временная зона — смещение от стандартного:
- до 1972 года - Гринвич (GMT)
- после 1972 — UTC — всемирное координированное
- Методы
  - getDefault()
  - getAvailableIDs()
  - getRawOffset() - смещение без учета летнего времени
  - getOffset(long date) — с учетом летнего времени
- Класс SimpleTimeZone — реализованный потомок

Calendar

ИТМО

- Абстрактный класс — машинное ↔ человеческое
  - Calendar getInstance()
  - add(int field, int amount);
  - roll(int field, int amount);
  - set(int field, int value);
  - Date getTime()
  - setTime(Date)
- класс GregorianCalendar
  - сочетает 2 календаря (григорианский и юлианский)

Date/Time API - новые классы

ИТМО

- java.time - дата, время, периоды
  - Instant, Duration, Period, LocalDate, LocalTime, LocalDateTime, OffsetTime, OffsetDateTime, ZonedDateTime
- java.time.chrono - календарные системы
- java.time.format - форматирование даты и времени
- java.time.temporal - единицы измерения и отдельные поля
- java.time.zone - временные зоны и правила

Java

Дни недели и месяцы (enums)

ИТМО

- enum DayOfWeek (1 (MONDAY) — 7 (SUNDAY))
- enum Month (1 (JANUARY) — 12 (DECEMBER))
- метод getDisplayName(style, locale)
- стиль — FULL, NARROW, SHORT / STANDALONE

Java

Представление даты и времени

ИТМО

- Year
- YearMonth
- MonthDay
- LocalDate
- LocalTime
- LocalDateTime

Java

Соглашения по именам методов

ИТМО

- Статические
  - of — создает экземпляр на основе входных параметров
    - LocalDate.of(year, month, day), ofYearDay(year, dayOfYear)
  - from — конвертирует экземпляр из другого типа
    - LocalDate.from(LocalDateTime)
  - parse — создает экземпляр из строкового представления
    - LocalDate.parse("2022-02-22")

Соглашения по именам методов

ИТМО

- Методы экземпляра
  - format — форматирует объект в строку
  - get — возвращает поля объекта // getHours()
  - with — возвращает копию с изменением // withYear(2021)
  - plus — возвращает копию с добавлением // plusDays(2)
  - minus — возвращает копию с убавлением // minusWeeks(3)
  - to — преобразует объект в другой тип // toLocalTime()
  - at — комбинирует объект с другим // date.atTime(time)

Временная зона

ИТМО

- ZoneId — идентификатор зоны
  - Europe/Moscow
- ZoneOffset — разница со стандартным временем
  - UTC+01:00, GMT-2
- OffsetTime = LocalTime + ZoneOffset
- OffsetDateTime = LocalDateTime + ZoneOffset
- ZonedDateTime = LocalDateTime + ZoneId
  - использует java.time.zone.ZoneRules

Когда использовать классы

ИТМО

- YearMonth - срок действия банковской карты
- MonthDay - праздники
- LocalDateTime - местное время
- OffsetDateTime - зональное время (веб, базы данных)
- ZonedDateTime - зональное время с учетом летнего

Java

Момент времени

ИТМО

- Класс Instant (timestamp)
  - now()
  - plusNanos()
  - plusMillis()
  - plusSeconds()
  - minusNanos()
  - minusMillis()
  - minusSeconds()

Разбор и форматирование

- java.time.format.DateTimeFormatter
  - константы формата:
    - BASIC_ISO_DATE ('20240229')
    - ISO_LOCAL_DATE ('2024-02-29')
    - ISO_ZONED_DATETIME ('2024-02-29T18:15:30+03:00[Europe/Moscow]')
    - RFC_1123_DATE_TIME ('Thu, 29 Feb 2024 18:15:30 GMT+3')
  - локализованные форматы
    - ofLocalizedDateTime(dateStyle, timeStyle)
    - стиль (FULL, LONG, MEDIUM, SHORT)
  - шаблон
    - ofPattern()
  - методы format() и parse()

Периоды даты и времени

ИТМО

- Duration — продолжительность в часах и менее
  - toNanos(), toMillis(), toSeconds(), toMinutes(), toHours(), toDays()
- Period — период в днях и более
  - getDays(), getMonths(), getYears()
- .between()
- .plus
- .minus

Java

Старые и новые классы

ИТМО

- Соответствия:
  - java.util.Date — java.time.Instant
  - java.util.GregorianCalendar — java.time.ZonedDateTime
  - java.util.TimeZone — ZoneId, ZoneOffset
- Методы:
  - Calendar.toInstant()
  - GregorianCalendar.toZonedDateTime()
  - GregorianCalendar.fromZonedDateTime()
  - Date.fromInstant()
  - Date.toInstant()
  - TimeZone.toZoneId()

Примеры

ИТМО

```
var date1 = LocalDate.of(2024, 03, 27);
var date2 = LocalDate.parse("2024-04-27");
var date3 = LocalDate.now()
              .with(TemporalAdjusters.firstDayOfMonth());

var dateTime1 = LocalDateTime.now();
var dateTime2 = dateTime1.minusHours(2).plusMinutes(30);

var zoneId = ZoneId.of("Europe/Moscow");
var zonedDateTime = ZonedDateTime.of(dateTime, zoneId);

var period = Period.between(date1, date2);
var duration = Duration.between(dateTime1, dateTime2);
```

Java