**20. Способы работы с базами данных**

**JDBC (Java Database Connectivity)**:

JDBC — это стандартный API, который позволяет Java-приложениям взаимодействовать с различными базами данных для выполнения SQL-запросов, обновления данных и получения результатов.. JDBC предоставляет набор интерфейсов и классов, которые используются для выполнения операций с базами данных, таких как создание таблиц, вставка, выборка и обновление данных.

Пример использования JDBC:

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**import** java.sql.SQLException;

**import** java.sql.Statement;

**public** **class** JdbcExample {

**public** **static** **void** main(String[] args) {

        String url = "jdbc:mysql://localhost:3306/mydb";

        String username = "root";

        String password = "password";

**try** {

            Connection connection = DriverManager.getConnection(url, username, password);

            Statement statement = connection.createStatement();

            statement.executeUpdate("CREATE TABLE users (id INT, name VARCHAR(50))");

            statement.executeUpdate("INSERT INTO users (id, name) VALUES (1, 'John Doe')");

            connection.close();

        } **catch** (SQLException e) {

            e.printStackTrace();

        }

    }

}

**JPA (Java Persistence API)**:

JPA — это спецификация Java EE, которая обеспечивает удобный и стандартизированный способ работы с базами данных на Java. JPA использует объектно-реляционное отображение (ORM) для сопоставления Java-объектов с таблицами базы данных. Это позволяет разработчикам работать с объектами и классами, а не с SQL-запросами.

Пример использования JPA:

**import** javax.persistence.EntityManager;

**import** javax.persistence.EntityManagerFactory;

**import** javax.persistence.Persistence;

**public** **class** JpaExample {

**public** **static** **void** main(String[] args) {

        EntityManagerFactory emf = Persistence.createEntityManagerFactory("my-pu");

        EntityManager em = emf.createEntityManager();

        em.getTransaction().begin();

        User user = **new** User();

        user.setId(1);

        user.setName("John Doe");

        em.persist(user);

        em.getTransaction().commit();

        em.close();

        emf.close();

    }

}

**Hibernate**

Hibernate — это популярная реализация JPA, которая предоставляет дополнительные возможности и производительность для работы с базами данных на Java. Hibernate может использоваться как самостоятельная библиотека или интегрироваться с Java EE и Spring фреймворками.

Пример использования Hibernate:

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.cfg.Configuration;

**public** **class** HibernateExample {

**public** **static** **void** main(String[] args) {

        Configuration configuration = **new** Configuration().configure();

        SessionFactory sessionFactory = configuration.buildSessionFactory();

        Session session = sessionFactory.openSession();

        session.beginTransaction();

        User user = **new** User();

        user.setId(1);

        user.setName("John Doe");

        session.save(user);

        session.getTransaction().commit();

        session.close();

        sessionFactory.close();

    }

}