01.WMC-Test Cheat sheet <3

In your Quarkus-project

Pom.xml

```
<!--JDBC Driver - Derby-->
<dependency>
    <groupId>io.quarkus</groupId>
    <artifactId>quarkus-jdbc-derby</artifactId>
</dependency>

<!--Hibernate ORM -->
<dependency>
    <groupId>io.quarkus</groupId>
    <artifactId>quarkus-hibernate-orm</artifactId>
</dependency>

<!--SmallRye OpenAPI-->
<dependency>
    <groupId>io.quarkus</groupId>
    <artifactId>quarkus-smallrye-openapi</artifactId>
</dependency>

<!-- testing -->
<dependency>
    <groupId>org.assertj</groupId>
    <artifactId>assertj-core</artifactId>
    <version>3.24.2</version>
    <scope>test</scope>
</dependency>

<dependency>
    <groupId>org.assertj</groupId>
    <artifactId>assertj-db</artifactId>
    <version>2.0.2</version>
    <scope>test</scope>
</dependency>
```

Download and set DB

- Go to http://bit.ly/htl-leonding-scripts and download the file ***download-derbydb-and-jdk***
- Run this commands in your Terminal:

```
chmod +x ./download-derbydb-and-jdk.sh
```

```
./download-derbydb-and-jdk.sh
./derbydb-start.sh
```

- Replace the application.properties
- Copy the datasource and add a db with 'import from clipboard'
- password: 'app'

Entities

```
@Entity
@Table(name = "person")
public class Person {

//region properties
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

//region constructors

//region getter and setter

//region toString

}
```

Repositories (control)

```
@ApplicationScoped
public class PersonRepository {

    @Inject
    EntityManager em;

    public Person save(Person person) {
        return em.merge(person);
    }

    public List<Person> findAll() {
        return em.createQuery("select p from Person p", Person.class).getResultList();
    }

    public Person findById(Long id) {
        return em.find(Person.class, id);
    }

    public void delete(Long id) {
```

```
        em.remove(findById(id));
    }

    public void save(Person person) {
        em.persist(person);
    }

    public void update(Long id, Person person) {
        Person p = findById(id);

        p.setName(person.getName());
        p.setAge(person.getAge());

        em.merge(p);
    }
}
```

Resources (boundary)

```
@Path("/person")
public class PersonResource{
    @Inject
    PersonRepository personRepository;

    @GET
    @Produces(MediaType.APPLICATION_JSON)
    @Path("{id}")
    public Person getPerson(@PathParam("id") Long id) {
        return personRepository.findById(id);
    }

    @POST
    @Consumes(MediaType.APPLICATION_JSON)
    @Transactional
    public Response createPerson(Person person) {
        personRepository.save(person);
        return Response.status(Response.Status.CREATED).entity("Person created").
build();
    }

    @PUT
    @Consumes(MediaType.APPLICATION_JSON)
    @Path("{id}")
    @Transactional
    public Response updatePerson(@PathParam("id") Long id, Person person) {
        personRepository.update(id, person);
        return Response.status(Response.Status.OK).entity("Person updated").build();
    }
```

```
}
```

Collections

Hashmap

A Hashmap store items in "key/value" pairs, and you can access them by an index of another type.

- Example:

```
//Create a Hashmap that will store *String* keys and *String* values;
HashMap<String, String> capitalCities = new HashMap<String, String>();

//Add items to it, use the put() method:
capitalCities.put("England", "London");

//To access a value, use the get() method and refer to its !key!:
capitalCities.get("England"); //returns 'London'

//To remove an item, use the remove() method and refer to the !key!:
capitalCities.remove("England");

//To remove all items, use the clear() method:
capitalCities.clear();

//Loop through the items of a HashMap with a for-each loop.

// Use the keySet() method if you only want the keys
for (String i : capitalCities.keySet()) {
  System.out.println(i); //returns 'England'
}

//Use the values() method if you only want the values
for (String i : capitalCities.values()) {
  System.out.println(i); //returns 'London'
}

// Print keys and values
for (String i : capitalCities.keySet()) {
  System.out.println("key: " + i + " value: " + capitalCities.get(i));
  //returns 'key: England value: London'
}
```

equals() and hashCode()

- Example:

```
public class Person {
    private String name;
    private int age;
    //constructor, getters and setters

    //equals() compares two objects for equality.
    @Override
    public boolean equals(Object o) {

        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;

        Person person = (Person) o;

        return age == person.age && Objects.equals(name, person.name);
    }

    @Override
    public int hashCode() {
        return Objects.hash(name, age); //returns a hashcode value for the object
    }
}

public class Main {
    public static void main(String[] args) {

        Person person1 = new Person("John", 20);
        Person person2 = new Person("John", 20);
        Person person3 = new Person("Bertl", 45);

        System.out.println(person1.equals(person2)); //returns 'true'


        Set<Person> persons = new HashSet<>();

        persons.add(person1);
        persons.add(person2);

        System.out.println(persons.size()); //returns '1'

        persons.add(person3);

       //The hashcode() and equals() methods are automatically called by contains()
and remove() methods of a Hashset.
       if(persons.contains(new Person("John", 20))) {
           System.out.println("John is in the set");
       }
    }
```

REGEX (Regular Expressions)

- Example:

```
String name = "Alsoduki";
System.out.println(name.matches("Alsoduki")); //returns 'true'

name = "aaaaabbbbccc";
S.o.p(name.matches("[a-z]+")); //returns 'true' if the string contains lowercase
letters between a and z. The '+' means that the string must contain at least one of
these letters.

name = "Abdul321";
S.o.p(name.matches("[a-zA-Z0-9]+")); //returns 'true'.

name = "abdul";
S.o.p(name.matches("\\p{Upper}[a-z]+")); //returns 'false'.The string must start with
an uppercase letter.

name = "abdi";
S.o.p(name.matches("[^a]+")); //returns 'false'. The string must not contain the
letter 'a'.

String email = "abdul.aldesoky@outlook.com";
S.o.p(email.matches("[a-zA-Z0-9]+\\.?[a-zA-Z0-9]+[@][a-zA-Z0-9]+[.]com$")); //returns
'true'

String number = "E12345678901";
S.o.p(number.matches("E[0-9]{11}")); //returns 'true' if the string starts with 'E'
and is followed by 11 digits.

number = "+4366412345678";
S.o.p(number.matches("^\\+43[0-9]{11}")); //returns 'true' if the string starts with
'+43' and is followed by 11 digits.

number = "a1b2c3";
S.o.p(number.matches("^[0-9a-fA-F]+$")); //returns 'true' if the string contains
hexadecimal characters
```

Read from file

CSV

- Example with BufferedReader:

```
@ApplicationScoped
public class InitBean {
```

```
    @Inject
    PersonRepository personRepository;

    @Transactional
    void startUp(@Observes Startup ev){
       Log.info("+++ It works +++")

        try{
            //Read from file
            BufferedReader reader = new BufferedReader(new FileReader(
"src/files/persons.csv"));

            String line = reader.readLine(); //read the first line

            while (line != null) {
                String[] attributes = line.split(";"); //split the line into an array
of Strings

                Person person = createPerson(attributes); //create a person object

                personRepository.save(person);

                line = reader.readLine(); //read the next line
            }
        }catch(FileNotFoundException e){
            System.out.println("File not found");
            return null;
        }catch (IOException e){
            e.printStackTrace();
        }finally{
            if(reader != null){
                try{
                  reader.close();
                }catch(IOException e){
                    e.printStackTrace();
                }
            }
        }
    }

    private Person createPerson(String[] data) {
        String name = data[0];
        int age = Integer.parseInt(data[1]);

        return new Person(name, age);
    }
}
```