Промежуточная аттестация Модуль 1 «Java Core»

Формулировка задания:

Необходимо реализовать приложение, предоставляющее функциональность работы с пользователем: добавление пользователя в список (регистрация), поиск пользователя по идентификатору, выгрузка информации обо всех пользователях, редактирование пользователя, удаление пользователя/пользователей.

Подробное описание функционала приложения

1. Исходный файл — список пользователей в следующем формате:

f5a8a3cb-4ac9-4b3b-8a65-c424e129b9d2|2023-12-25T19:10:11.556|noisemc_99|789ghs|789ghs|Крылов|Виктор|Павлович|25|true

2. Реализовать классы проекта в папке model:

Класс User

- id типа String – гарантированно уникальный ID пользователя. Состоит из букв и цифр.

- дата LocalDateTime добавления в систему, по умолчанию сегодня, формат: дата и время;

- login типа String, не может быть только из цифр, содержит буквы, цифры, знак подчеркивания, меньше 20 символов

- password и confirmPassword типа String, одинаковые, не может быть только из букв, содержит буквы, цифры, знак подчеркивания, меньше 20 символов

- фамилия - строка, состоит только из букв;

- имя - строка, состоит только из букв;

- отчество - строка, состоит только из букв, может отсутствовать;

- возраст – целое число, может отсутствовать;

- isWorker – является ли сотрудником предприятия, по умолчанию false.

---

3. Реализовать интерфейс по работе с пользователем в папке repositories:

Интерфейс UsersRepository со следующими абстрактными методами:

- Метод void create(User user) – создание пользователя и запись его в файл;

- Метод User findById(String id) — поиск пользователя в файле по

идентификатору;

- Метод List<User> findAll() - выгрузка всех пользователей из файла;

- Метод void update(User user) — обновление полей существующего в файле пользователя;

- Метод void deleteById(String id) — удаление пользователя по идентификатору.

- Метод void deleteAll() — удаление всех пользователей.

4. Реализовать имплементацию UsersRepositoryFileImpl интерфейса UsersRepository по работе с пользователем в папке repositories.

- Метод void create(User user) – создает пользователя и записывает его в список пользователей последним номером. Обновляется файл, туда дописывается пользователь. При формировании объекта User может использоваться функциональный подход с созданием Mapper.

- Метод User findById(String id) — поиск пользователя в файле по идентификатору. Возвращает пользователя или выбрасывает исключение: «Пользователя с заданным идентификатором не существует».

- Метод List<User> findAll() - выгрузка всех пользователей из файла.

- Метод void update(User user) — обновление полей существующего в файле пользователя. Метод находит в файле пользователя с id user-а и заменяет его значения. При отсутствии id в списке формируется сообщение об

отсутствии данных о пользователе и создается новый пользователь. При ошибке в требованиях к полям класса выбрасывается исключение.

Примечание: Реализовать замену данных в файле с полной перезаписью файла (заменить в списке параметры пользователя и записать заново в файл

всех пользователей).

- Метод void deleteById(String id) — удаление пользователя по идентификатору. После удаления пользователя переписать файл и записать список без удаленного пользователя. Если пользователя с таким идентификатором нет — выбросить исключения: «Пользователя с заданным идентификатором не существует».

- Метод void deleteAll() — удаление всех пользователей из списка и из файла.

- Дополнительно. Поиск списка пользователей по заданному полю. Например, выгрузка пользователей по возрасту findByAge, выгрузка пользователей-работников предприятия findByIsWorker.

Пользователи хранятся в коллекции ArrayList. Для работы с пользователями предварительно весь список читается из файла. Ошибки чтения/записи в файл должны быть обработаны.

Наименование файла хранится как константа в имплементации интерфейса UsersRepositoryFileImpl.

5. Реализовать класс App для проверки работоспособности приложения\

a. Сформировать файл с набором пользователей.

b. В методе main создать экземпляр класса UsersRepositoryFileImpl. Проверить все функции класса — создание, поиск по id, выгрузка всех пользователей, обновление данных пользователя, удаление по id, удаление всех пользователей, поиск по полю (при наличии данной функции).

6. Дополнительно. Для удобства работы с приложением может быть создан набор своих custom исключений в отдельной папке.

7. Настроить папку с unit-тестами проекта. Подключить к проекту библиотеку JUnit5. Сгенерировать автоматически каркас для тестов с помощью

Intellij Idea.

8. Создать набор unit-тестов для проверки функционала приложения. Создать минимум 4 теста. Тесты должны быть следующих типов:

- Позитивные тесты. Проверка корректности работы методов с обычными данными.

- Тесты на вызов исключений. Проверка работы методов с ошибкой формата ввода.

```java
package attestation.attestation01.model;

import java.time.LocalDateTime;
import java.util.Objects;

public class User {  23 usages  programakc *

    private String id;  9 usages
    private LocalDateTime date = LocalDateTime.now();  9 usages
    private String login;  9 usages
    private String password;  10 usages
    private String confirmPassword;  9 usages
    private String lastName;  9 usages
    private String firstName;  9 usages
    private String middleName;  9 usages
    private Integer age;  9 usages
    private Boolean isWorker = false;  9 usages

    public User() {  no usages  programakc
    }

    public User(String inputLine) {  3 usages  programakc *
        String[] params = inputLine.split("\\|");

        this.id = params[0];
        this.date = LocalDateTime.parse(params[1]);
        this.login = params[2];
        this.password = params[3];
        this.confirmPassword = params[4];
        this.lastName = params[5];
        this.firstName = params[6];
        this.middleName = params[7];
        this.age = Integer.parseInt(params[8]);
```

```
        this.isWorker = Boolean.parseBoolean(params[9]);
    }

    public User(String id, LocalDateTime date, String login, String password,  no usages  programakc *
                String confirmPassword, String lastName, String firstName,
                String middleName, Integer age, Boolean isWorker) {

        this.id = id;
        this.date = date;

        if (isCorrectLogin(login)) this.login = login;

        if (isCorrectPassword(password)) this.password = password;

        if (isCorrectConfirmPassword(password, confirmPassword))
            this.confirmPassword =
                    confirmPassword;

        if (isCorrectLastName(lastName)) this.lastName = lastName;

        if (isCorrectFirstName(firstName)) this.firstName = firstName;

        if (isCorrectMiddleName(middleName)) this.middleName = middleName;

        if (isCorrectAge(String.valueOf(age))) this.age = age;

        this.isWorker = isWorker;
    }

    public String getId() { return id; }
```

```java
    public void setId(String id) { this.id = id; }

    public LocalDateTime getDate() { return date; }

    public void setDate(LocalDateTime date) { this.date = date; }

    public String getLogin() { return login; }

    public void setLogin(String login) {  no usages  programakc *
        if (isCorrectLogin(login)) this.login = login;
    }

    public String getPassword() { return password; }

    public void setPassword(String password) {  no usages  programakc *
        if (isCorrectPassword(password)) this.password = password;
    }

    public String getConfirmPassword() { return confirmPassword; }

    public void setConfirmPassword(String confirmPassword) {  no usages  programakc *
        if (isCorrectConfirmPassword(this.password, confirmPassword))
            this.confirmPassword = confirmPassword;
    }

    public String getLastName() { return lastName; }

    public void setLastName(String lastName) {  no usages  programakc *
        if (isCorrectLastName(lastName)) this.lastName = lastName;
    }

    public String getFirstName() { return firstName; }
```

```java
115
116     public void setFirstName(String firstName) {  no usages  programakc *
117         if (isCorrectFirstName(firstName)) this.firstName = firstName;
118     }
119
120     public String getMiddleName() { return middleName; }
123
124     public void setMiddleName(String middleName) {  no usages  programakc *
125         if (isCorrectMiddleName(middleName)) this.middleName = middleName;
126     }
127
128     public Integer getAge() { return age; }
131
132     public void setAge(Integer age) {  no usages  programakc *
133         if (isCorrectAge(String.valueOf(age))) this.age = age;
134     }
135
136     public Boolean getWorker() { return isWorker; }
139
140     public void setWorker(Boolean worker) { isWorker = worker; }
143
144 @   public void update(User user) {  1 usage  programakc
145         this.id = user.getId();
146         this.date = LocalDateTime.now();
147         this.login = user.getLogin();
148         this.password = user.getPassword();
149         this.confirmPassword = user.getConfirmPassword();
150         this.lastName = user.getLastName();
151         this.firstName = user.getFirstName();
152         this.middleName = user.getMiddleName();
153         this.age = user.getAge();
154         this.isWorker = user.getWorker();
```

```
155     }
156
157     @Override  programakc
158     public boolean equals(Object o) {
159         if (o == null || getClass() != o.getClass()) return false;
160         User user = (User) o;
161         return Objects.equals(id, user.id) && Objects.equals(date, user.date)
162                 && Objects.equals(login, user.login) && Objects.equals(password
163                 , user.password) && Objects.equals(confirmPassword
164                 , user.confirmPassword) && Objects.equals(lastName
165                 , user.lastName) && Objects.equals(firstName, user.firstName)
166                 && Objects.equals(middleName, user.middleName) && Objects.equals
167                 (age, user.age) && Objects.equals(isWorker, user.isWorker);
168     }
169
170     @Override  programakc
171     public int hashCode() {
172         return Objects.hash(id, date, login, password, confirmPassword, lastName
173                 , firstName, middleName, age, isWorker);
174     }
175
176     @Override  programakc
177     public String toString() {
178         return (id + "|" + date + "|" + login + "|" + password + "|"
179                 + confirmPassword + "|" + lastName + "|" + firstName + "|"
180                 + middleName + "|" + age + "|" + isWorker);
181     }
182
183 @   private Boolean isCorrectLogin(String login) {  2 usages  new *
184         if ((login.length() < 20 && !login.chars().allMatch(Character::isDigit))
185                 && (login.matches("\\w+"))) {
186             return true;
```

```
187         } else {
188             throw new IllegalArgumentException("Недопустимый формат логина!");
189         }
190     }
191
192 @   private Boolean isCorrectPassword(String password) {  2 usages  new *
193         if ((password.length() < 20 && !password.chars().allMatch(Character::isLetter))
194                 && (password.matches("\\w+"))) {
195             return true;
196         } else {
197             throw new IllegalArgumentException("Недопустимый формат пароля!");
198         }
199     }
200
201 @   private Boolean isCorrectConfirmPassword(String password, String confirmPassword) {  2 usages  new *
202         if (password.equals(confirmPassword)) {
203             return true;
204         } else {
205             throw new IllegalArgumentException("Пароли не совпадают!");
206         }
207     }
208
209 @   private Boolean isCorrectLastName(String lastName) {  2 usages  new *
210         if (lastName.chars().allMatch(Character::isLetter)) {
211             return true;
212         } else {
213             throw new IllegalArgumentException("Фамилия пользователя должна " +
214                     "содержать только буквы!");
215         }
216     }
217
```

```
218 @     private Boolean isCorrectFirstName(String firstName) {  2 usages  new *
219           if (firstName.chars().allMatch(Character::isLetter)) {
220               return true;
221           } else {
222               throw new IllegalArgumentException("Имя пользоателя должно " +
223                       "сожержать только буквы!");
224           }
225       }
226
227 @     private Boolean isCorrectMiddleName(String middleName) {  2 usages  new *
228           if (middleName.chars().allMatch(Character::isLetter) || middleName.isEmpty()) {
229               return true;
230           } else {
231               throw new IllegalArgumentException("Отчество пользователя должно " +
232                       "содержать только буквы либо отсутствовать!");
233           }
234       }
235
236 @     private Boolean isCorrectAge(String age) {  2 usages  new *
237           if ((age.chars().allMatch(Character::isDigit) && Integer.parseInt(age) > 0)
238                   || age.isEmpty()) {
239               return true;
240           } else {
241               throw new IllegalArgumentException("Возраст пользователя должен " +
242                       "быть числом больше 0 либо отсутствовать!");
243           }
244       }
245   }
```

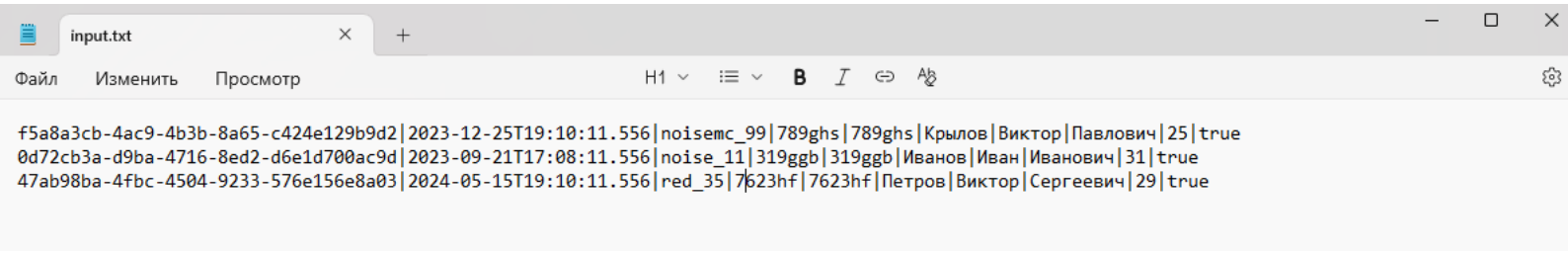
```java
package attestation.attestation01.repository;

import attestation.attestation01.model.User;

import java.util.List;

public interface UsersRepository {  4 usages  1 implementation  programakc
    void create(User user);  1 usage  1 implementation  programakc

    User findById(String id);  1 usage  1 implementation  programakc

    List<User> findAll();  5 usages  1 implementation  programakc

    void update(User user);  1 usage  1 implementation  programakc

    void deleteById(String id);  1 usage  1 implementation  programakc

    void deleteAll();  1 usage  1 implementation  programakc
}
```

```java
package attestation.attestation01.repository.Impl;

import attestation.attestation01.exception.UserNotFoundException;
import attestation.attestation01.model.User;
import attestation.attestation01.repository.UsersRepository;

import java.io.*;
import java.util.ArrayList;
import java.util.List;

public class UsersRepositoryFileImpl implements UsersRepository {  2 usages  programakc *

    private final List<User> USERS = new ArrayList<>();  9 usages
    private final static String FILENAME = "src/attestation/attestation01/input.txt";  4 usages

    public UsersRepositoryFileImpl() { readFile(); }

    @Override  1 usage  programakc
    public void create(User user) {
        USERS.add(user);
        writeFile();
    }

    @Override  1 usage  programakc
    public User findById(String id) {
        return USERS
                .stream() Stream<User>
                .filter( User e -> e.getId().equals(id))
                .findFirst() Optional<User>
                .orElseThrow(() -> new UserNotFoundException("Пользователь " +
                        "с id: " + id + " не найден!"));
    }
```

```
35
36      @Override  5 usages  programakc
37      public List<User> findAll() { return USERS; }
40
41      @Override  1 usage  programakc
42      public void update(User user) {
43
44          try {
45              User foundedUser = USERS.stream() Stream<User>
46                      .filter( User e -> e.getId().equals(user.getId()))
47                      .findFirst() Optional<User>
48                      .orElseThrow(() -> new UserNotFoundException("Пользователь " +
49                              "с id: " + user.getId() + " не найден!"));
50
51              foundedUser.update(user);
52          } catch (Exception e) {
53              System.out.println(e.getMessage());
54              USERS.add(user);
55          }
56
57          writeFile();
58      }
59
60      @Override  1 usage  programakc
61      public void deleteById(String id) {
62          USERS.removeIf( User user -> user.getId().equals(id));
63          writeFile();
64      }
65
66      @Override  1 usage  programakc
67      public void deleteAll() {
68          USERS.clear();
```

```
69          writeFile();
70      }
71
72      private void readFile() {  1 usage   programakc
73          List<User> result;
74
75          try (BufferedReader bufferedReader =
76                       new BufferedReader(new FileReader(FILENAME))) {
77
78              result = bufferedReader.lines()  Stream<String>
79                      .map(User::new)  Stream<User>
80                      .toList();
81
82          } catch (IOException e) {
83              System.out.println("Не удалось найти файл " + FILENAME);
84              result = new ArrayList<>();
85          }
86
87          USERS.addAll(result);
88      }
89
90      private void writeFile() {  4 usages   programakc *
91          try (BufferedWriter bufferedWriter =
92                       new BufferedWriter(new FileWriter(FILENAME))) {
93
94              for (User u : USERS) {
95                  bufferedWriter.write(u.toString() + "\n");
96              }
97
```

```
99                System.out.println("Не удалось найти файл " + FILENAME);
100           }
101       }
102   }
```


input.txt
Файл
Изменить
Просмотр
H1
f5a8a3cb-4ac9-4b3b-8a65-c424e129b9d2|2023-12-25T19:10:11.556|noisemc_99|789ghs|789ghs|Крылов|Виктор|Павлович|25|true
0d72cb3a-d9ba-4716-8ed2-d6e1d700ac9d|2023-09-21T17:08:11.556|noise_11|319ggb|319ggb|Иванов|Иван|Иванович|31|true
47ab98ba-4fbc-4504-9233-576e156e8a03|2024-05-15T19:10:11.556|red_35|7623hf|7623hf|Петров|Виктор|Сергеевич|29|true

```java
package attestation.attestation01;

import attestation.attestation01.model.User;
import attestation.attestation01.repository.Impl.UsersRepositoryFileImpl;
import attestation.attestation01.repository.UsersRepository;

public class App {  programakc *

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

        UsersRepository usersRepository = new UsersRepositoryFileImpl();

        System.out.println("\nСчитывание всех пользователей из " +
                "файла:\n" + usersRepository.findAll());

        User user = new User("00000000-4fbc-4504-9233-576e156e8a03" +
                "|2024-03-11T23:10:11.556|original_41|jd6yen|jd6yen|Семёнов|Виктор|Андреевич|32|true");

        usersRepository.create(user);
        System.out.println("\nСоздание нового пользователя:\n" + usersRepository.findAll());

        System.out.println("\nНахождение пользователя по ID: 0d72cb3a-d9ba-4716-8ed2-d6e1d700ac9d");
        System.out.println(usersRepository.findById("0d72cb3a-d9ba-4716-8ed2-d6e1d700ac9d"));

        User user2 = new User("47ab98ba-4fbc-4504-9233-576e156e8a03|2024-05" +
                "-15T19:10:11.556|gr_35|7623hf|7623hf|Петров|Виктор|Сергеевич|29|true");

        usersRepository.update(user2);
        System.out.println("\nОбновление данных пользователя:");
        System.out.println(usersRepository.findAll());

        usersRepository.deleteById("47ab98ba-4fbc-4504-9233-576e156e8a03");
        System.out.println("\nУдаление пользователя по ID:\n" + usersRepository.findAll());
```

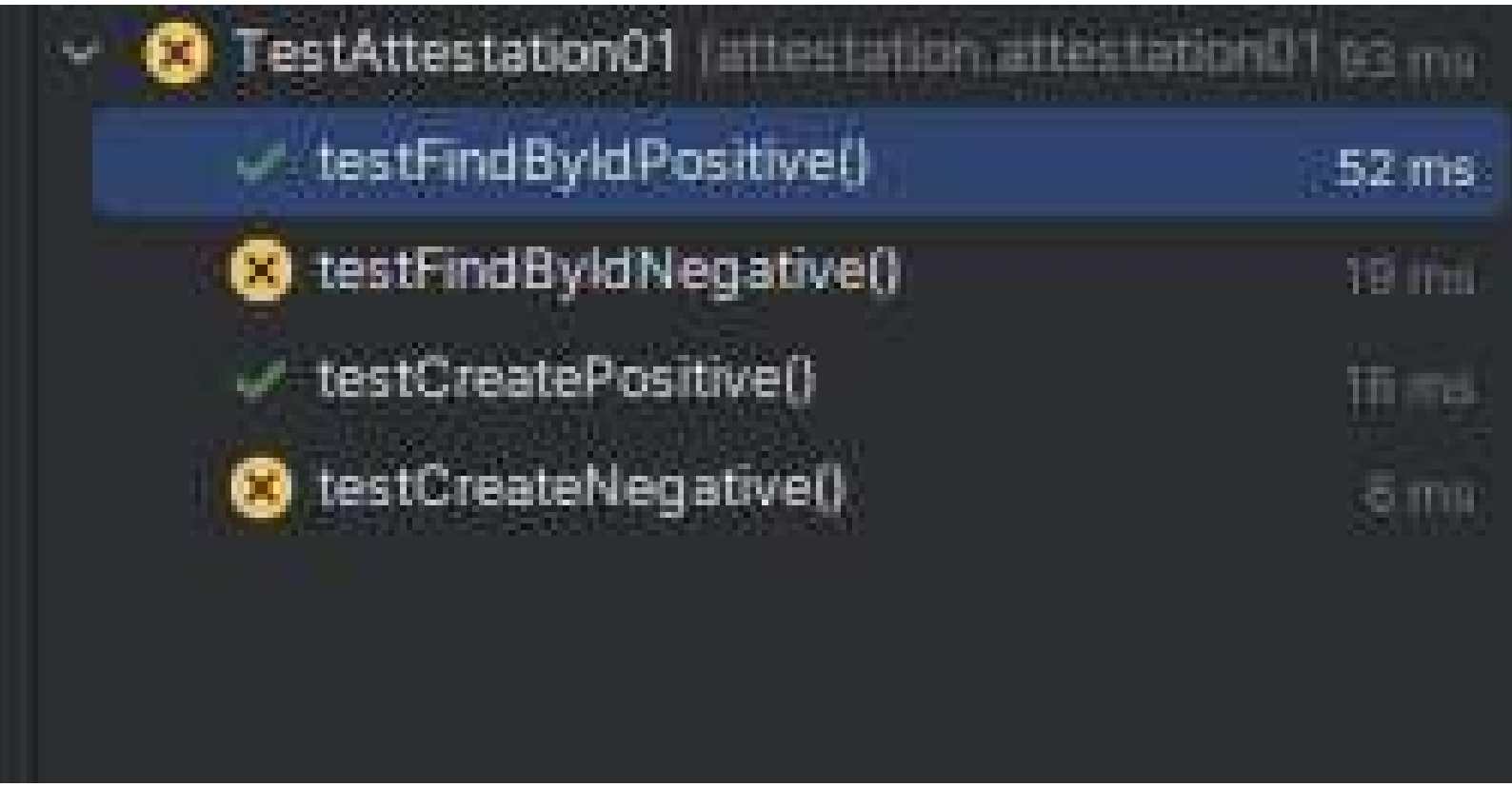
```
34
35          usersRepository.deleteAll();
36          System.out.println("\nУдаление всех пользователей:\n" + usersRepository.findAll());
37      }
38  }
```

```
"C:\Program Files\Java\jdk-21.0.2\bin\java.exe" -javaagent:G:\JetBrains\IntelliJIdea2025.1\lib\idea_rt.jar=60621
 -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8 -classpath
 D:\JavaProject\JavaDevHomeworks\out\production\JavaDevHomeworks attestation.attestation01.App

Считывание всех пользователей из файла:
[f5a8a3cb-4ac9-4b3b-8a65-c424e129b9d2|2023-12-25T19:10:11.556|noisemc_99|789ghs|789ghs|Крылов|Виктор|Павлович|25|true,
 0d72cb3a-d9ba-4716-8ed2-d6e1d700ac9d|2023-09-21T17:08:11.556|noise_11|319ggb|319ggb|Иванов|Иван|Иванович|31|true,
 47ab98ba-4fbc-4504-9233-576e156e8a03|2024-05-15T19:10:11.556|red_35|7623hf|7623hf|Петров|Виктор|Сергеевич|29|true]

Создание нового пользователя:
[f5a8a3cb-4ac9-4b3b-8a65-c424e129b9d2|2023-12-25T19:10:11.556|noisemc_99|789ghs|789ghs|Крылов|Виктор|Павлович|25|true,
 0d72cb3a-d9ba-4716-8ed2-d6e1d700ac9d|2023-09-21T17:08:11.556|noise_11|319ggb|319ggb|Иванов|Иван|Иванович|31|true,
 47ab98ba-4fbc-4504-9233-576e156e8a03|2024-05-15T19:10:11.556|red_35|7623hf|7623hf|Петров|Виктор|Сергеевич|29|true,
 00000000-4fbc-4504-9233-576e156e8a03|2024-03-11T23:10:11.556|original_41|jd6yen|jd6yen|Семёнов|Виктор|Андреевич|32|true]

Нахождение пользователя по ID: 0d72cb3a-d9ba-4716-8ed2-d6e1d700ac9d
0d72cb3a-d9ba-4716-8ed2-d6e1d700ac9d|2023-09-21T17:08:11.556|noise_11|319ggb|319ggb|Иванов|Иван|Иванович|31|true

Обновление данных пользователя:
[f5a8a3cb-4ac9-4b3b-8a65-c424e129b9d2|2023-12-25T19:10:11.556|noisemc_99|789ghs|789ghs|Крылов|Виктор|Павлович|25|true,
 0d72cb3a-d9ba-4716-8ed2-d6e1d700ac9d|2023-09-21T17:08:11.556|noise_11|319ggb|319ggb|Иванов|Иван|Иванович|31|true,
 47ab98ba-4fbc-4504-9233-576e156e8a03|2025-09-25T18:40:20.306901100|gr_35|7623hf|7623hf|Петров|Виктор|Сергеевич|29|true,
 00000000-4fbc-4504-9233-576e156e8a03|2024-03-11T23:10:11.556|original_41|jd6yen|jd6yen|Семёнов|Виктор|Андреевич|32|true]

Удаление пользователя по ID:
[f5a8a3cb-4ac9-4b3b-8a65-c424e129b9d2|2023-12-25T19:10:11.556|noisemc_99|789ghs|789ghs|Крылов|Виктор|Павлович|25|true,
 0d72cb3a-d9ba-4716-8ed2-d6e1d700ac9d|2023-09-21T17:08:11.556|noise_11|319ggb|319ggb|Иванов|Иван|Иванович|31|true,
 00000000-4fbc-4504-9233-576e156e8a03|2024-03-11T23:10:11.556|original_41|jd6yen|jd6yen|Семёнов|Виктор|Андреевич|32|true]

Удаление всех пользователей:
[]

Process finished with exit code 0
```

```java
package attestation.attestation01;

import attestation.attestation01.model.User;
import attestation.attestation01.repository.Impl.UsersRepositoryFileImpl;
import org.junit.jupiter.api.Assertions;
import org.junit.jupiter.api.Test;

import java.time.LocalDateTime;

public class TestAttestation01 {  new *

    UsersRepositoryFileImpl urfi = new UsersRepositoryFileImpl();  4 usages

    @Test  new *
    void testCreatePositive() {
        User user = new User("00000000-4fbc-4504-9233-576e156e8a03" +
                "|2024-03-11T23:10:11.556|original_41|jd6yen|jd6yen|Семёнов|Виктор|Андреевич|32|true");
        urfi.create(user);

        Assertions.assertAll(() -> {
            Assertions.assertNotNull(user);
            Assertions.assertEquals("00000000-4fbc-4504-9233-576e156e8a03", user.getId());
            Assertions.assertEquals(LocalDateTime.parse("2024-03-11T23:10:11.556"), user.getDate());
            Assertions.assertEquals("original_41", user.getLogin());
            Assertions.assertEquals("jd6yen", user.getPassword());
            Assertions.assertEquals("jd6yen", user.getConfirmPassword());
            Assertions.assertEquals("Семёнов", user.getLastName());
            Assertions.assertEquals("Виктор", user.getFirstName());
            Assertions.assertEquals("Андреевич", user.getMiddleName());
            Assertions.assertEquals(Integer.parseInt("32"), user.getAge());
            Assertions.assertEquals(Boolean.parseBoolean("true"), user.getWorker());
        });
    }
```

```
34
35      @Test  new *
36      void testCreateNegative() { // different password
37          User user = new User("00000000-4fbc-4504-9233-576e156e8a03" +
38                  "|2024-03-11T23:10:11.556|original_41|jd6yen|jd6y|Семёнов|Виктор|Андреевич|32|true");
39          urfi.create(user);
40
41          Assertions.assertAll(() -> {
42              Assertions.assertNotNull(user);
43              Assertions.assertEquals("00000000-4fbc-4504-9233-576e156e8a03", user.getId());
44              Assertions.assertEquals(LocalDateTime.parse("2024-03-11T23:10:11.556"), user.getDate());
45              Assertions.assertEquals("original_41", user.getLogin());
46              Assertions.assertEquals("jd6yen", user.getPassword());
47              Assertions.assertEquals("jd6yen", user.getConfirmPassword()); //
48              Assertions.assertEquals("Семёнов", user.getLastName());
49              Assertions.assertEquals("Виктор", user.getFirstName());
50              Assertions.assertEquals("Андреевич", user.getMiddleName());
51              Assertions.assertEquals(Integer.parseInt("32"), user.getAge());
52              Assertions.assertEquals(Boolean.parseBoolean("true"), user.getWorker());
53          });
54      }
55
56      @Test  new *
57      void testFindByIdPositive() {
58          User user = new User("00000000-4fbc-4504-9233-576e156e8a03" +
59                  "|2024-03-11T23:10:11.556|original_41|jd6yen|jd6yen|Семёнов|Виктор|Андреевич|32|true");
60          urfi.findById(user.getId());
61
62          Assertions.assertAll(() -> {
63              Assertions.assertNotNull(user.getId());
64              Assertions.assertEquals("00000000-4fbc-4504-9233-576e156e8a03", user.getId());
65          });
66      }
```

```
67
68      @Test  new *
69      void testFindByIdNegative() { // different password
70          User user = new User("00000000-4fbc-4504-9233-576e156e8a03" +
71                  "|2024-03-11T23:10:11.556|original_41|jd6yen|jd6yen|Семёнов|Виктор|Андреевич|32|true");
72          urfi.findById(user.getId());
73
74          Assertions.assertAll(() -> {
75              Assertions.assertNotNull(user.getId());
76              Assertions.assertEquals("00110000-4fbc-4504-9233-576e156e8a03", user.getId());
77          });
78      }
79  }
```


TestAttestation01 | attestation.attestation01 93 ms
testFindByIdPositive() 52 ms
testFindByIdNegative() 19 ms
testCreatePositive() 16 ms
testCreateNegative() 6 ms