Latvijas Republikas izglītības un zinātnes ministrija

Daugavpils Tehnoloģiju un tūrisma tehnikums

PROJEKTA DARBS

Programmēšanas tehniķis

specialitāte

1. projekta darbs

temats

Testēšanas sistēmas izveidošana

PASKAIDROJOŠAIS RAKSTS

DTTT.P.D.0025 .025PR

Profesijas kods 33 481 031 Programmēšanas nodaļa

PR-21. grupa

Izveidoja Ņ.Ruļevičs

paraksts V.,Uzvārds

Pārbaudīja I. Dortiņa

paraksts V.,Uzvārds

2023./2024. m.g.

# Saturs

[Saturs 2](#_Toc168909160)

[Ievads 3](#_Toc168909161)

[1. Uzdevuma nostādne 4](#_Toc168909162)

[1.1 Sistēmas modelis 4](#_Toc168909163)

[1.1.1. Priekšmetiskās jomas informācijas modelis 4](#_Toc168909164)

[1.1.2. Sistēmas apkārtnes shēma 5](#_Toc168909165)

[1.1.3. Viedokļu analīze 5](#_Toc168909166)

[1.1.4. Datu modelēšana 7](#_Toc168909167)

[1.1.5. Datu vārdnīca 8](#_Toc168909168)

[1.2 Sistēmas evolūcija 9](#_Toc168909169)

[1.3 Sistēmas funkcionālās prasības 10](#_Toc168909170)

[1.4 Sistēmas nefunkcionālās prasības 13](#_Toc168909171)

[2. Priekšmetiskās jomas klašu sistēmas izstrāde 16](#_Toc168909172)

[2.1 Klašu diagramma 16](#_Toc168909173)

[2.2 Klašu realizēšana Java valodā 17](#_Toc168909174)

[3. Testa programmas izstrāde 20](#_Toc168909175)

[3.1 Testēšanas metodikas 20](#_Toc168909176)

[3.2 Testēšanas programmas struktūra 20](#_Toc168909177)

[3.3 Testēšanas rezultāti 23](#_Toc168909178)

[4. Lietotāja rokasgrāmata 23](#_Toc168909179)

[4.1 Lietotāja instrukcija 23](#_Toc168909180)

[4.2 Palīdzības sistēma (Help) 24](#_Toc168909181)

[Secinājumi 24](#_Toc168909182)

[Izmantotās literatūras (informācijas avotu) saraksts 25](#_Toc168909183)

# Ievads

Prasību dokumentā ir aprakstīta tehnikuma uzskaites vadības sistēma, kura izpilda sekojošas

funkcijas:

* datu ievade
* datu apstrāde
* datu izņemšana
* atskaišu ģenerēšana
* adaptīva dialoga izveidošana

Sistēma nodrošina audzēkņa un darbinieka “sadarbību”, t.i. skolotāji ievada datu bāzē vajadzīgo informāciju par savu priekšmetu un izmanto agrāk ievadīto informāciju.

Programma veic skolēnu testēšanas funkciju, izmantojot dažādus skolotāju izveidotus testus.

# Uzdevuma nostādne

## Sistēmas modelis

### Priekšmetiskās jomas informācijas modelis

Apskatāmā sistēmā tika izdalīti objekti: Datu bāze (Fails ar atbildēm), Tests, Rezultāti, Administrators / Skolotājs, Lietotājs. Šie objekti un saites starp tiem ir paradīti ([1.1. att.)](#page4).

Datu bāze

Tests

Rezultāti

Skolotājs

Lietotājs

1.1. att. Vienkāršots testēšanas sistēmas modelis

Saišu nozīme vienkāršotā nodaļas modelī:

* Skolotājs ir testa veidotājs un pārbaudītājs, kuram ir pieeja pie atbildēm, testam un lietotāja rezultātiem;
* Datu bāze satur failus ar jautājumiem un ar pariezām atbildēm;
* Lietotājs ir testa pildītājs, kas beigās var apskatīt savus rezultātus, nepieciešamības gadījumā viņš varēs sazināties ar skolotāju;
* Tests ir instruments, kas palīdzēs pārbaudīt lietotāju zināšanas un novērtēt to ar atzīmi;
* Rezultāti parāda atzīmi, pareizās un nepareizās atbildes skaitliski un procentuāli.

### Sistēmas apkārtnes shēma

Dati, kā atbildes, ko ievada audzēknis tiek saglabāti failā. Tomēr dažreiz atbildes netiek automātiski novērtētas un skolotājam ir iespēja manuāli novērtēt iesniegto atbildi. Saskarni ar lietotājiem nodrošinās interfeisa modulis ([1.2. att.)](#page4).

Lietotājs

Lietotāja saskarsne

Sistēma

Datu bāze

Lietotāja dati un rezultāti

1.2. att. Sistēmas apkārtnes modeļa piemērs

### Viedokļu analīze

Savāktie un identificētie viedokļi par sistēmu ir attēloti ar burbuļdiagrammas palīdzību ([1.3. att.),](#page5) tas faktiski ir viedokļu kopums, kur katra viedokļa nosaukums ir ierakstīts atsevišķa elipsē.

Nefunkcionālie viedokli

Datu viedokli

Funkcionālie viedokli

Viedokļi par lietotāju

1.3. att. Viedokļu burbuļu diagramma

Otrais etaps viedokļu analīzē ir viedokļu klasifikācija, kad viedokļi ir sagrupēti grupās (1.4. att.[)](#page5).

1.4. att. Viedokļu hierarhiskā struktūra

### Datu modelēšana

Datu modelēšanai ir tiek izmantots ER modelis. Testēšanas sistēmas ER modelis ([1.5. att.)](#page6) ietver datu entītiju kopumu un relāciju kopumu starp datu entītijām, pie kam ar šo modeli var attēlot dažāda tipa relācijas starp datu elementiem.

Skolotājs

Audzēknis

Jautājums

 Tests

Satur

Izveido

Pilda

Jautājums

Noņem

1.5. att. Vienkāršots testēšanas sistēmas ER modelis

### Datu vārdnīca

|  |  |  |
| --- | --- | --- |
| Datu tips | Nosaukums | Apraksts |
| VARCHAR | FirstName | Kolonnā, kas satur lietotāja vārdu. |
| VARCHAR | LastName | Kolonnā, kas satur lietotāja uzvārdu. |
| VARCHAR | UserName | Kolonna, kurā ir lietotājvārds, ar kura palīdzību tiek veikta pieslēgšanās programmā. |
| VARCHAR | Password | Kolonna, kurā ir lietotāja parole, tiek izmantota, lai īstenotu pieslēgšanos programmā. |
| VARCHAR | UserType | Mainīgais lielums, kuru pārbaudot, tiks veikts **skolotāja** (ar vairāk iespējām) un **skolēna** (ar mazāk iespējām) GUI sadalījums. |

1.1. Tabula. Datubāzes "users" tabula ar kolonnām.

|  |  |  |
| --- | --- | --- |
| Datu tips | Nosaukums | Apraksts |
| VARCHAR | TestName | Testa nosaukums, kurā tiks izmantots jautājums |
| VARCHAR | TestDescription | Testa apraksts |
| VARCHAR | Question | Jautājuma teksts, kas tiks iekļauts testā. |
| VARCHAR | AnswerFirst | Kolonna ar pirmo (pēc kārtas) atbildi uz jautājumu |
| VARCHAR | AnswerSecond | Kolonna ar otro (pēc kārtas) atbildi uz jautājumu |
| VARCHAR | AnswerThird | Kolonna ar trešo (pēc kārtas) atbildi uz jautājumu |
| VARCHAR | AnswerRight | Dublē vienu no atbildēm, kas ir pareiza. |

1.2. Tabula. Datubāzes "test" tabula ar kolonnām.

|  |  |  |
| --- | --- | --- |
| Datu tips | Nosaukums | Apraksts |
| VARCHAR | UserName | Testa nosaukums, kurā tiks izmantots jautājums |
| VARCHAR | TestDescription |  |
| VARCHAR | Mark | Jautājuma teksts, kas tiks iekļauts testā. |

1.3. Tabula. Datubāzes "marks" tabula ar kolonnām.

## Sistēmas evolūcija

Testēšanas sistēma ir neatkarīga un pašpietiekama programma. Nav paredzēts, ka produkts sastāvēs no vairākiem komponentiem.

Programmai ir četras ārējās saskarnes, kas redzamas 1.4. nodaļā: viena reģistrācijai sistēmā, viena pieteikšanai, viena testa izpildei un viena jauna jautājuma pievienošana. Grafisko lietotāja saskarni var uzlabot, pievienojot dažādas krāsu un dizaina shēmas.

Pašlaik testēšanas sistēma nodrošina divu veidu lietotājus: administratoru un testa dalībnieku. Nākotnē ir plānots pievienot vēl vienu lietotāju grupu ar funkciju veidot jaunus testus un pārbaudīt nokārtotos testus, tādējādi lietotājiem būs divas lomas: skolēns, kas veic testu, un skolotājs, kas veido un pārbauda testus. Administratoram plānots ieviest lietotāja vārda, lietotājvārda vai paroles maiņas funkciju, kā arī lietotāju pievienošanu un dzēšanu un lomu maiņu.

Programmā ir viens tests ar 5 jautājumiem, bet nākotnē sistēmu var pilnveidot, pievienojot jaunas testu tēmas, parādot un izskaidrojot kļūdas pēc testa izpildes, novērtējot lietotāju, parādot lietotāja datus lietotāja izvēlnē vai skolotāju izvēlnē.

Dažiem datu veidiem sistēma izmantos teksta datnē saglabātos datus, nākotnē programmu var uzlabot, lai tā varētu strādāt ar datubāzi (datubāzes savienojums, testa jautājumi un atbildes utt.)

## Sistēmas funkcionālās prasības

1. **Funkcija “Reģistrēties sistēmā”**

***ID: T1***

**Ievads:** Ļauj ievadīt nepieciešamus datus, lai izveidotu savu kontu. Visi ievadīti dati tiek saglabāti failā (datu bāzē).

**Ievade:**

1. Lietotāja vārds
2. Lietotāja uzvārds
3. Lietotāja lietotājvārds
4. Lietotāja parole
5. Ievadītās paroles apstiprināšana.

**Apstrāde:**

1. Lietotāja vārda saglabāšana failā.
2. Lietotāja uzvārda saglabāšana failā.
3. Lietotāja lietotājvārda saglabāšana failā.
4. Lietotāja paroles saglabāšana failā.

Ja ievadītā parole ir nepareiza, tad tiek atvērts kļūdas dialoglodziņš.

**Izvade:** Dialoglodziņš, kurā uzrakstīts “Reģistrēšana ir veiksmīgi pabeigta!”.

1. **Funkcija “Ielogoties sistēmā”**

***ID: T2***

**Ievads:** Ļauj ievadīt nepieciešamus datus, lai ieiet sava kontā.

**Ievade:**

1. Lietotāja lietotājvārds
2. Lietotāja parole

**Apstrāde:**

1. Lietotāja lietotājvārda pārbaude un atrašana DB.
2. Lietotāja paroles pārbaude un atrašana DB.

Ja ievadītā parole vai/un lietotājvārds ir nepareiza, tad tiek atvērts kļūdas dialoglodziņš.

**Izvade:** Dialoglodziņš, kurā uzrakstīts “Ieeja ir veiksmīgi pabeigta!”.

1. **Funkcija “Sākt testu”**

***ID: T3***

**Ievads:** Ļauj administratoram sākt testu.

**Ievade:** Administrators noklikšķina uz pogu “Sākt testu”. Pēc tam sistēma uzdod jautājumu: "Vai Jūs vēlaties sākt testu?" Administratoram ir divi varianti atbildēm "Jā" vai "Nē".

**Apstrāde:** Sistēma pārbauda atbildi uz uzdoto jautājumu un pēc tam sāk testu, ja uz uzdoto jautājumu atbilde "Jā". Tad tests ir pieejams lietotājiem to veikšanai.

**Izvade:** Administratora ekrānā tiek parādīts dialoglodziņš ar tekstu "Tests ir atļauts!"

1. **Funkcija “Pildīt testu”**

***ID: T4***

**Ievads:** Ļauj lietotājam sākt testa izpildi.

**Ievade:** Lietotājs noklikšķina uz pogu “Pildīt testu”.

**Apstrāde:** Ja tests ir pieejams izpildei, tad pēc testa izvēles tiek parādīts dialoglodziņš ar tekstu "Tests ir pieejams! Vai vēlaties izpildīt testu?". Lietotājam ir divi varianti atbildēm "Jā" vai "Nē". Ja tests nav pieejams izpildei, tad pēc testa izvēles tiek parādīts dialoglodziņš ar tekstu "Tests nav pieejams!"

**Izvade:** Lietotājā ekrānā tiek parādīts dialoglodziņš ar uzrakstu "Veiksmi testā!"

1. **Funkcija “Atbildēt uz testa jautājumu”**

***ID: T5***

**Ievads:** Ļauj ievadīt atbildi uz testa jautājumus.

**Ievade:** Lietotājs izvēlas 1 no 3 piedāvātajām atbildēm uz uzdoto jautājumu.

**Apstrāde:** Sistēma pārbauda atbildi uz jautājumu. Pēc katras pareizās atbildes sistēma pievieno 1.

**Izvade:** Pēc izvēlētās atbildes lietotājs noklikšķina uz pogas "Nākamais", pēc kura tiek parādīts nākamā izvēlne ar testa jautājumiem.

1. **Funkcija “Parādīt rezultātu”**

***ID: T6***

**Ievads:** Ļauj uzzināt nokārtotā testa rezultātus.

**Ievade:** Lietotājs noklikšķina uz pogas "Parādīt rezultātu".

**Apstrāde:** Sistēma saskaita pareizo atbilžu skaitu, nepareizo atbilžu skaitu un neatbildēto jautājuma skaitu, tad aprēķina pareizo atbilžu procentuālo vērtību, pēc tam nolasa vērtējumu par 10 balles skalu un atrod komentāru par saņemto atzīmi.

**Izvade:** Tiek atvērts dialoglodziņš, kurā tiek rakstīts šāds teksts:

* 1. Pareizo atbilžu procentuālā daļa.
  2. Atzīme.

1. **Funkcija "Pievienot jaunu testu"**

**ID: T7**

**Ievads:** Ļauj skolotājam pievienot jaunu jautājumu esošajam testam.

**Ievade:**

1. Testa nosaukums
2. 5 Jautājumi
3. 3 Atbildes varianti katram jautājumam (viens pareizs)
4. Pareizā atbilde
5. Testa apraksts

**Apstrāde:** Sistēma pieņem ievadīto informāciju par jauno testu, pārbauda, vai ir norādīts un vai ir norādīta pareizā atbilde.

**Izvade:** Paziņojums par jaunā testa veiksmīgu pievienošanu vai kļūdas paziņojums, ja ievadītajā informācijā ir kļūdas vai trūkumi.

1. **Funkcija "Izdzest testu"**

**ID: T7**

**Ievads:** Ļauj skolotājam pievienot jaunu jautājumu esošajam testam.

**Ievade:**

1. Testa nosaukums

**Apstrāde:** Sistēma pieņem ievadīto testa nosaukumu un izdzēš to.

**Izvade:** Paziņojums par jaunā testa veiksmīgu izdzēšanu vai kļūdas paziņojums, ja ievadītajā informācijā ir kļūdas vai trūkumi.

## Sistēmas nefunkcionālās prasības

1. Prasības produktam:

1.1. Lietotāju saskarne ar sistēmu notiek latviešu valodā.

1.2. Produktam nav paredzētas vairākas sastāvdaļas. Produktam ir trīs ārējās saskarnes: reģistrēšanai sistēmā, ielogošanai sistēmā, testa pildīšanai.

1.3. Programma ir paredzēta vienam lietotājam, kurš saprot latviešu valodu. Lietotāja vecuma vai izglītības ierobežojumi nav paredzēti.

1. Ārējās saskarnes prasības

2.1. Lietotāja saskarne: produkts paredz vienotu lietotāja saskarni.

2.2. Ekrāna formāti: minimālais ekrāna izmērs ir 10” ar minimālo izšķirtspēju –800x600 punkti.

2.3. Lietotāja saskarne “Lietotāja reģistrēšanas saskarne”. Saskarnes struktūra: skat. 1.6. att.

Reģistrēties

Vārds:

Uzvārds:

Lietotājvārds:

Parole:

Parole (atkārtoti):

1.6. att. Lietotāja reģistrēšanas saskarne

Saskarnes elementi:

1. teksta lauks “Vārds”
2. teksta lauks “Uzvārds”
3. teksta lauks “Lietotājvārds”
4. teksta lauks “Parole”
5. teksta lauks “Parole (atkārtoti)”
6. poga “Reģistrēties”

2.4. Lietotāja saskarne “Lietotāja ielogošanas saskarne”. Saskarnes struktūra: skat. 1.7. att.

Ielogoties

Lietotājvārds:

Parole:

Reģistrēties

1.7. att. Lietotāja ielogošanas saskarne

Saskarnes elementi:

1. teksta lauks “Lietotājvārds”
2. teksta lauks “Parole”
3. poga “Ielogoties”
4. poga “Reģistrēties”

2.5. Lietotāja saskarne “Testa izpilde”. Saskarnes struktūra: skat. [1.8. att.](#page11)

a)

b)

c)

Sākt testu

Tests "Nosaukums"

1. jautājums „Jautājuma teksts”

1. atbilde

2. atbilde

3. atbilde

Tālāk

Beigt

Rezultāti

Procenti: X%

Atzīme: A

1.8. att. Testa izpildes saskarne: testa izvēle/sakums; b) atbilde uz jautājumu/-iem ; c) rezultāti

1. Saskarnes elementi:
2. teksta lauks (virsraksts) “Tests "Nosaukums" ”
3. poga “Sākt testu”
4. Saskarnes elementi(Satur tikai viena jautājuma piemēru. Patiesībā vienā testa logā būs 5 šādi jautājumi):
5. teksta lauks (virsraksts) “1. jautājums "Jautājuma teksts" ”
6. radio poga “1. atbilde”
7. radio poga “2. atbilde”
8. radio poga “3. atbilde”
9. poga “Sākt testu”
10. Saskarnes elementi:
11. teksta lauks (virsraksts) “Rezultāti”
12. teksta lauks “Procenti: ”
13. teksta lauks “Atzīme: ”
14. poga “Beigt”

2.6. Lietotāja saskarne “Jauna jautājuma pievienošana”. Saskarnes struktūra: skat. 1.9. att.

Jauna test pievienošana

Jautājuma virsraksts

1. atbildes variants

2. atbildes variants

 3. atbildes variants

Pievienot

1.9. att. Jauna jautājuma pievienošanas lietotāja saskarne

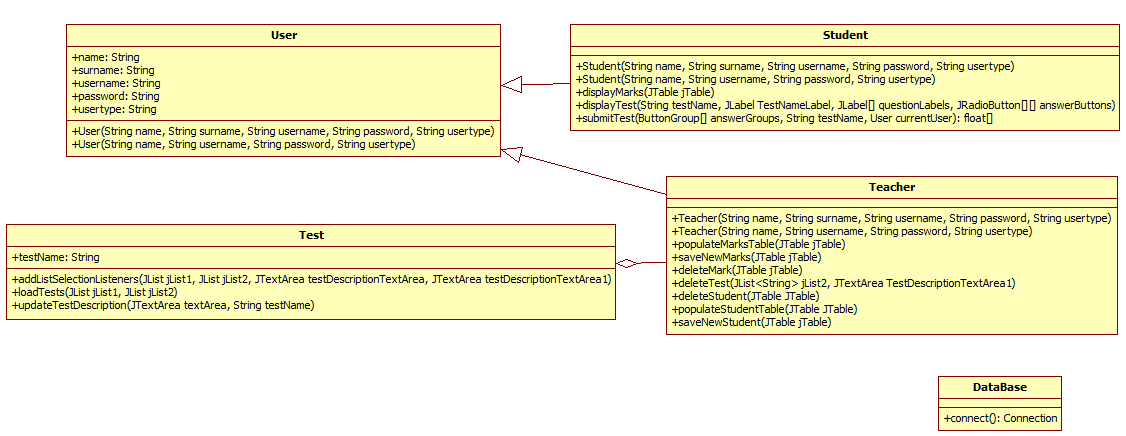
Saskarnes elementi(Satur tikai viena jautna jautājuma pievienošnas piemēru. Patiesībā vienā jauna testa pievienošanas logā būs 5 šādi jautājumi):

1. teksta lauks (virsraksts) “ Jauna jautājuma pievienošana”
2. teksta lauks “ Jautājuma virsraksts”
3. teksta lauks “1. atbildes variants”
4. radio poga “”, kas izmatota lai parradīt, ka 1. atbīlde ir pareiza.
5. teksta lauks “2. atbildes variants”
6. radio poga “”, kas izmatota lai parradīt, ka 2. atbīlde ir pareiza.
7. teksta lauks “3. atbildes variants”
8. radio poga “”, kas izmatota lai parradīt, ka 3. atbīlde ir pareiza.
9. poga “ Pievienot”

# Priekšmetiskās jomas klašu sistēmas izstrāde

## Klašu diagramma

Klašu diagrammā ir realizēts vispārīgs klašu apraksts – klašu hierarhijas vispārējā struktūru, to sadarbība, atribūti, metodes, saskarnes un to savstarpējās attiecības, izmantojot UML valodu.



2.1. att. Testēšanas sistēmas klašu diagramma UML valodā

Balstoties uz izstrādāto sistēmas modeli var veikt sekojošu klašu iedalījumu (2.1. att.):

* Klase **User** – ir atbildīga par darbu ar lietotājiem.
* **public** User (name : String, surname : String, username : String, password : String, userType: String) – konstruktors, kurš izveido jaunu lietotāju ar norādītiem datiem;
* **public** User (username : String, password : String, usertype: String) – konstruktors, kurš izveido jaunu lietotāju tikai ar lietotājvārdu, paroli un lietotāja tīpu.
* Klase **Teacher** – nodrošina darbu ar lietotajiem un testiem.
* **public** Teacher (name : String, surname : String, username : String, password : String, userType: String) – konstruktors, kurš izveido jaunu skolotāju ar norādītiem datiem;
* **public** Teacher (name : String, surname : String, username : String, password : String, usertype: String) – konstruktors, kurš izveido jaunu skolotāju tikai ar lietotājvārdu, paroli un lietotāja tīpu;
* **public void** populateMarksTable(JTable jTable) – metode, kura aizpilda atzimju tabulu;
* **public void** saveNewMarks(JTable jTable) – metode, kura aizpilda atzimju tabulu;
* **public void** deleteTest(JList<String> jList2, JTextArea TestDescriptionTextArea1) – metode, kura aizpilda atzimju tabulu;
* **public void** deleteStudent(JTable jTable) – metode, kura nodrošina lietotāja izņemšanu no administratora puses;
* **public void** saveNewUser(JTable jTable) – metode, kura nodrošina lietotāja pievienošanu ar norādītiem datiem no administratora puses;
* **public void** populateStudentTable(JTable jTable) – metode, kura aizpilda lietotāja tabulu.
* Klase **Student** – nodrošina audzēkņa testa pildīšanas darbus.
* **public** Student (name : String, surname : String, username : String, password : String, userType: String) – konstruktors, kurš izveido jaunu studentu ar norādītiem datiem;
* **public** Student (name : String, surname : String, username : String, password : String, usertype: String) – konstruktors, kurš izveido jaunu studentu tikai ar lietotājvārdu, paroli un lietotāja tīpu;
* **public int** displayMarks(jTable jTable) – metode, kura nodrošina tabulas ar atzīmju aizpildīšanu;
* **public** **void** displayTest(String testName, JLabel TestNameLabel, JLabel[] questionLabels, JRadioButton[][] answerButtons) – metode, kura nodrošina testa aizpildīšanu ar jautājumu;
* **public float[]** submitTest(ButtonGroup[] answerGroups, String testName, User currentUser) – metode, kura nodrošina datu saglabāšanu datubāzē.
* Klase **Test** – ir atbildīga par darbu ar testu.
* **public void** addListSelectionListeners(JList jList1, JList jList2, JTextArea testDescriptionTextArea, JTextArea testDescriptionTextArea1)– metode, kas sarakstiem pievieno darbību noklausīšanos.
* **public void** loadTests(JList jList1, JList jList2) – metode, kura aipild sarakstu ar testiem no datu bāzes;
* **public void** updateTestDescription(JTextArea textArea, String testName) – metode, kura nodrošina testa apraksta ieladēšanu.

Klase **DataBase** – ir atbildīga par darbu ar datu bāzem.

* **public void** connect() – metode, kura nodrošina pieslēgšanu datu bazei.

## Klašu realizēšana Java valodā

Izstrādātās klašu diagrammas (2.1. att.) realizēšanai ***NetBeans*** vidē tika izveidots projekts ar nosaukumu ***TestingSystem***, kurā izstrādātās klases tika aprakstītas ***Java*** valodā. Šeit tiek piedāvāts klašu vispārīgā apraksta (klašu deklarācijas) programmas kods un metožu apraksti.

Pilns avota programmas kods ar klašu metožu definīcijām ir ievietots pielikumā “Pielikums A. Klašu sistēma”, bet programmas kods, kas realizē šīs klašu sistēmas testēšanu (klase ar grafisko lietotāja saskarni) ir ievietots pielikumā “Pielikums B. testa programmas kods”.

User klase:

public class User {

String name;

String surname;

String username;

String password;

String usertype;

public User(String name, String surname, String username, String password, String usertype) {

}

public User(String name, String username, String password, String usertype) {

}

}

Teacher klase:

public class Teacher extends User {

public Teacher(String name, String surname, String username, String password, String usertype) {

super(name, surname, username, password, usertype);

}

public Teacher(String name, String username, String password, String usertype){

super(name, username, password, usertype);

}

public void populateMarksTable(JTable jTable) {

}

public void saveNewMarks(JTable jTable){

}

public void deleteMark(JTable jTable) {

}

public void deleteTest(JList<String> jList2, JTextArea TestDescriptionTextArea1) {

}

public void deleteStudent(JTable JTable){

}

public void populateStudentTable(JTable JTable){

}

public void saveNewStudent(JTable jTable){

}

}

Student klase:

public class Student extends User {

public Student(String name, String surname, String username, String password, String usertype) {

super(name, surname, username, password, usertype);

}

public Student(String name, String username, String password, String usertype) {

super(name, username, password, usertype);

}

public void displayMarks(JTable jTable1) {

}

public void displayTest(String testName, JLabel TestNameLabel, JLabel[] questionLabels, JRadioButton[][] answerButtons) {

}

public float[] submitTest(ButtonGroup[] answerGroups, String testName, User currentUser) {

}

Test klase:

public class Test {

String testName;

public void addListSelectionListeners(JList jList1, JList jList2, JTextArea testDescriptionTextArea, JTextArea testDescriptionTextArea1) {

}

public void loadTests(JList jList1, JList jList2, DefaultListModel listModel) {

}

DataBase klase:

public class DataBase {

public Connection connect() {

Connection con = null;

return con;

}

}

# Testa programmas izstrāde

## Testēšanas metodikas

Pirms priekšmetiskās jomas klašu sistēmas testēšanas programmas izstrādes, ir jānosaka testēšanas metodika.

Ir jāatšķir koda testēšana un atkļūdošana. Atkļūdošanu veic programmētājs, izmantojot iebūvētos izstrādes vides rīkus un balstoties uz pieredzi programmas koda rakstīšanā. Būtībā runa iet par sintaktisko un semantisko kļūdu identificēšanu programmas tekstā.

Testēšana ir process, kurā nepieciešams plānot un īstenot vairākas sākotnējās procedūras, no kurām galvenā ir testa piemēru kopas izstrāde, kas veido testa plānu. Testpiemēri vairumā gadījumu ir balstīti uz sistēmas funkcionālajām prasībām un var ietekmēt dažādus attīstības līmeņus (vienības testēšana, integrācijas testēšana, sistēmas testēšana).

Projekta darba laikā nepieciešams veikt vienkāršotu vienības testēšanas versiju, kas realizē visu izstrādāto klašu metožu testēšanu. Ar testēšanas metodiku projekta darbā pieņem dažādu klases metožu izsaukšanas iespēju sarakstu ar gaidāmajiem rezultātiem un šo testēšanas izsaukumu izpildes kārtību.

## Testēšanas programmas struktūra

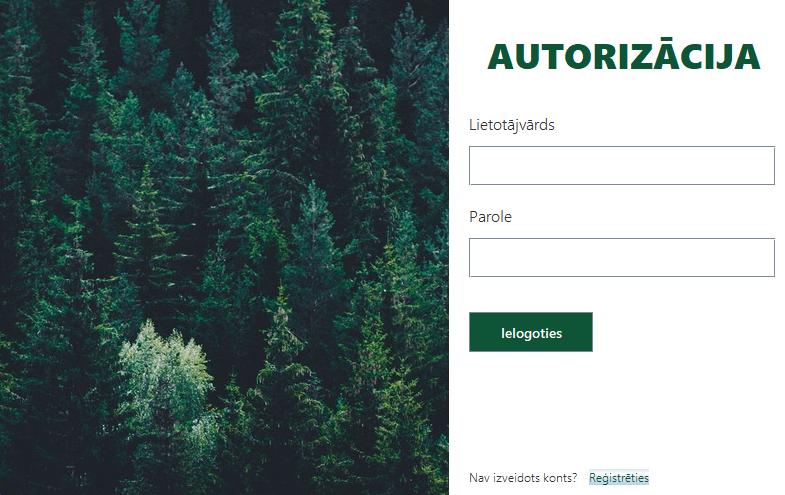
Testēšanas programmas projektēšana un programmatūras izstrāde ir ļoti līdzīga iepriekš aprakstītajam klašu sistēmas izstrādes procesam.

Jāatzīmē tie momenti un prasības, kas pastāv, izstrādājot testēšanas programmu.

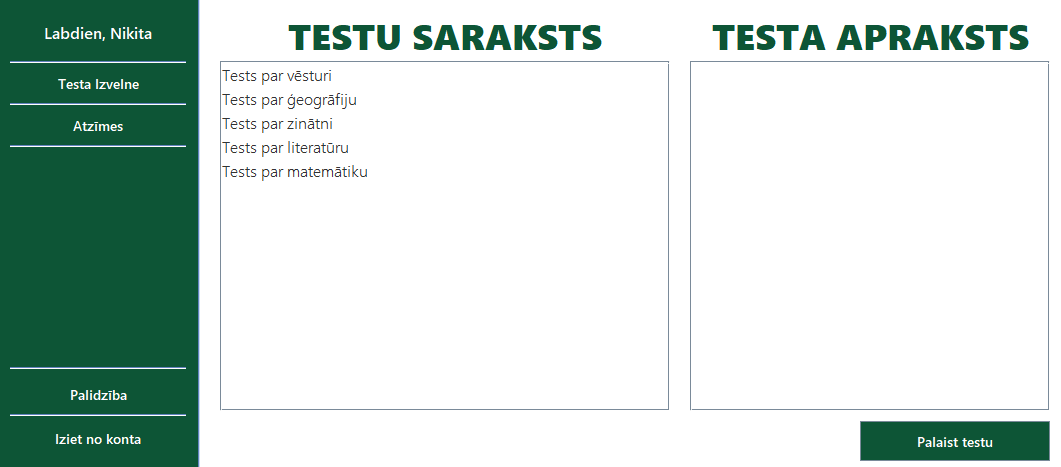
* Testēšanas programmai ir loga interfeiss, kas ietver sevī izvēlnes un interaktīvā interfeisa rīkus, ar kuru palīdzību var pārbaudīt izveidoto klašu sistēmu
* Interfeisa elementu (pogas, saraksti, ievades lauki, dialoglodziņi, grafiskie elementi utt.) komplekts tiek noteikts, pamatojoties uz iepriekšējā sadaļā aprakstīto testēšanas metodiku.
* Testēšanas programmas interfeisa kvalitātei jābūt vienkāršai lietošanā un interaktīvai. Galvenā prasība ir spēja realizēt testēšanas metodes.

1. Ielogošanas procesa apraksts:

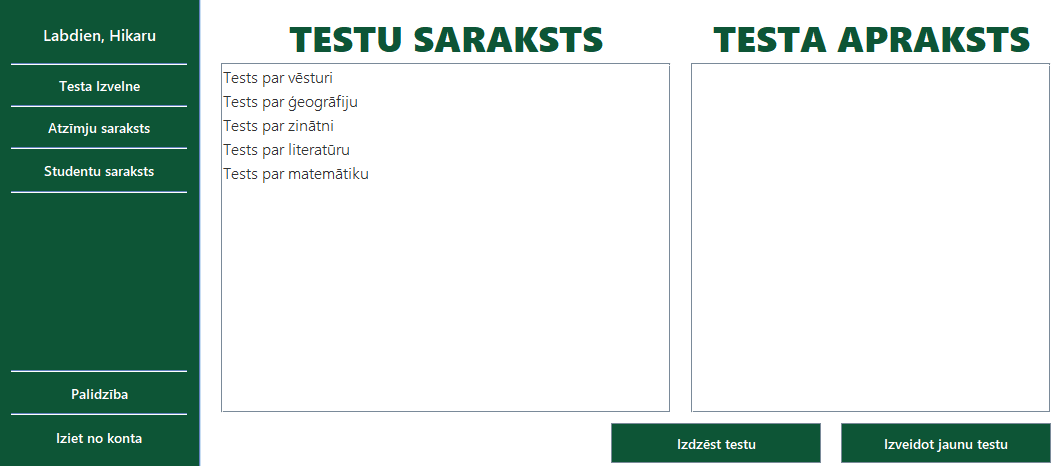
* Pieslēgšanās process ir paredzēts, lai pievienotu sistēmā reģistrētus lietotājus. Lai lietotājs varētu piekļūt testam, lietotājam ir jābūt pieteicies. Pieteikšanās procesā lietotājs ievada iepriekš reģistrētos datus - lietotāja vārdu un paroli (3.1. att.). Ja lietotāja dati sakrīt ar testa datiem, lietotājs nonāk galvenajā lietojumprogrammas logā, bet tā izskats un funkcijas ir atkarīgas no tā, kāda tipa lietotājs ir pieteicies kontā. Ja lietotājs ir skolēns, atveras **skolēna galvenais lietojumprogrammas logs** (3.2. att.), ja lietotājs ir skolotājs, atveras **skolotāja galvenais lietojumprogrammas logs** (3.3. att.).
* Ielogošanas procesā ir iespējamas vairākas kļūdas – neaizpildīti lauki, neprecīzi ievadīti lietotāja dati.



3.1. att. Testēšanas sistēmas ielogošanas forma



3.2. att. Galvenais programmas logs (students)

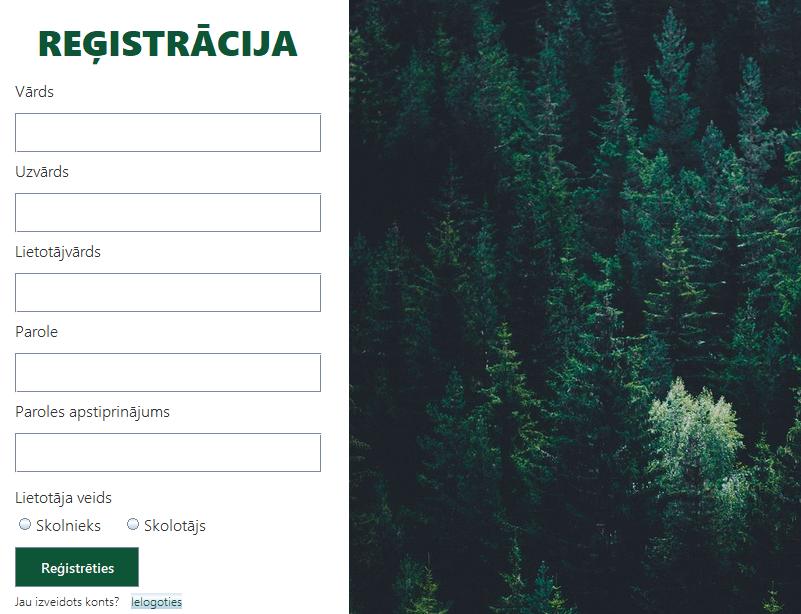


3.3. att. Galvenais programmas logs (skolotājs)

1. Reģistrācijas procesa apraksts:

* Reģistrācijas process ir paredzēts, lai sistēmā pievienotu jaunus lietotājus. Reģistrācijas procesā lietotājs ievada datus, kas turpmāk tiks izmantoti, pieslēdzoties kontam, - vārdu, uzvārdu, lietotājvārdu, paroli, paroles apstiprinājumu un lietotāja veidu (3.4. att.). Ja lietotājs ir ievadījis datus pareizi, proti, ievērojot šos trīs noteikumus:
  + Visu datu esamība,
  + Lietotājvārds ir unikāls (nav iepriekš reģistrētu lietotāju ar tādu pašu lietotājvārdu),
  + Atbilstošas paroles,

tad lietotājs tiek veiksmīgi reģistrēts sistēmā, saņem par to paziņojumu un atgriežas pie pieteikšanās loga.



3.4. att. Testēšanas sistēmas registrācijas forma

## Testēšanas rezultāti

Programmai tika izstrādāts tikai viens vienkāršs JUnit tests, jo izstrādājamā programma ir darbvirsmas lietojumprogramma, kas darbojas ar logiem, un šādu testu īstenot JUnit testos ir ārkārtīgi grūti.

JUnit Testa kods:

public class TeacherTest {

public TeacherTest() {

}

@BeforeClass

public static void setUpClass() {

}

@AfterClass

public static void tearDownClass() {

}

/\*\*

\* Test of Teacher(String name, String username, String password, String usertype) constructor, of class Teacher.

\*/

@Test

public void testTeacherCreation(){

System.out.println("Teacher(String name, String username, String password, String usertype) constructor of class Teacher test");

Teacher teacher = new Teacher("John", "mcLovin", "password", "teacher");

assertEquals("John", teacher.name);

assertEquals("mcLovin", teacher.username);

assertEquals("password", teacher.password);

assertEquals("teacher", teacher.usertype);

Teacher teacher2 = new Teacher("Ivan", "ivan", "QWERTY", "teacher");

assertEquals("Ivan", teacher2.name);

assertEquals("ivan", teacher2.username);

assertEquals("QWERTY", teacher2.password);

assertEquals("teacher", teacher2.usertype);

Teacher teacher3 = new Teacher("Kilian", "KMBP88", "123456", "teacher");

assertEquals("Kilian", teacher3.name);

assertEquals("KMBP88", teacher3.username);

assertEquals("123456", teacher3.password);

assertEquals("teacher", teacher3.usertype);

Teacher teacher4 = new Teacher("Jose", "JOSE2004", "pas123456789", "teacher");

assertEquals("Jose", teacher4.name);

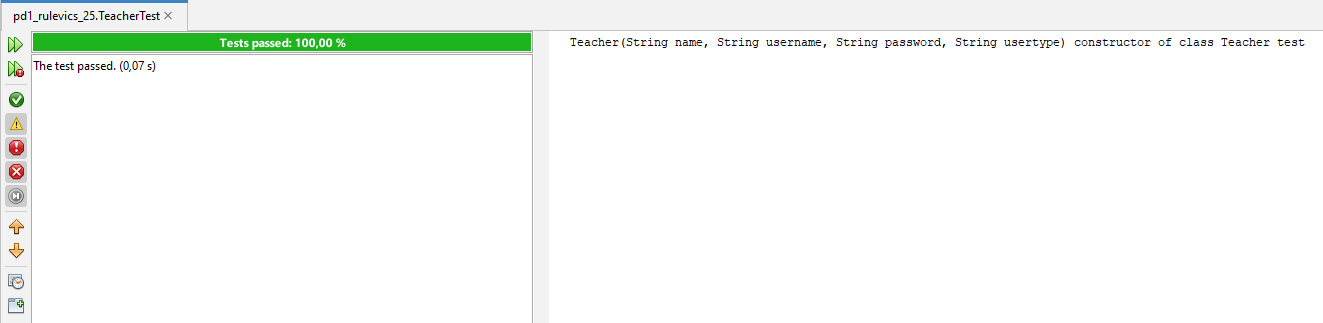
assertEquals("JOSE2004", teacher4.username);

assertEquals("pas123456789", teacher4.password);

assertEquals("teacher", teacher4.usertype);

}

}



3.5. att. Klases “Teacher” konstruktora tests

# Lietotāja rokasgrāmata

## Lietotāja instrukcija

1. Lietotājs palaiž programmu.
2. Piesakās sistēmā, izmantojot savu lietotājvārdu un paroli, ja lietotājam nav sava konta, tad viņš var reģistrēt to.
   1. Reģistrācijas logā lietotājs aizpilda visus nepieciešamus datus un noklikšķina uz pogas **“Reģistrēties”**.
3. Studenta izvēlnē ir 4 pogas, 1 saraksts un 1 tabula.
   1. Sarakstā lietotājs var izvēlēties veicamo testu. Noklikšķinot uz konkrēta testa, ekrāna kreisajā pusē tiks parādīts testa apraksts.
   2. Poga **“Testa izvēlne”** ļauj lietotājam pārslēgties uz testa izvēles ekrānu.
   3. Poga **“Atzīmes”** ļauj lietotājam pārslēgties uz logu ar novērtēšanas tabulu.
   4. Poga **“Iziet no konta”** ļauj lietotājam iziet no konta un atgriezties autorizācijas logā.
   5. Poga **“Palaist testu”** ļauj sākt pildīt izvelēto testu.
   6. Tabulā tiek parādīts saraksts ar visām skolēna atzīmēm.
4. Skolotāja izvēlnē ir 11 pogas, 1 saraksts un 2 tabulas.
   1. Sarakstā skolotājs var izvēlēties testu. Noklikšķinot uz konkrēta testa, ekrāna labājā pusē tiks parādīts testa apraksts.
   2. Poga **“Testa izvēlne”** ļauj lietotājam pārslēgties uz testa izvēles ekrānu.
   3. Poga **“Atzīmju saraksts”** ļauj skolotājam pārslēgties uz logu ar visu studentu novērtēšanas tabulu.
   4. Poga **“Studentu saraksts”** ļauj skolotājam pārslēgties uz logu ar visu studentu datu tabulu.
   5. Poga **“Iziet no konta”** ļauj lietotājam iziet no konta un atgriezties autorizācijas logā.
   6. Poga **“Pievienot jaunu testu”** ļauj pievienot jaunu testu.
   7. Poga **“Izdzēst testu”** ļauj izdzēst izvelēto testu.
   8. Poga **“Izdzēst”** ļauj dzēst atlasīto elementu no atzīmju tabulas.
   9. Poga **“Saglabāt”** ļauj saglabāt atzīmju tabulā veiktās izmaiņas.
   10. Poga **“Izdzēst studentu”** ļauj dzēst atlasīto elementu no studentu tabulas.
   11. Poga **“Pievienot jaunu studentu”** ļauj pievienot jaunu ierakstu studentu tabulā.
   12. Poga **“Saglabāt”** ļauj saglabāt studentu tabulā veiktās izmaiņas.
   13. Tabulā **“Atzīmes”** ir parādīts saraksts ar visu skolēnu vērtējumiem. Skolotājs var mainīt lietotāju vērtējumus no tabulas.
   14. Tabulā **“Studentu saraksts”** ir parādīts saraksts ar visu skolēnu datiem. Skolotājs var mainīt lietotāju datus no tabulas.

## Palīdzības sistēma (Help)

Laipni lūgti mūsu lietotnes palīdzības sadaļā! Šeit Jūs atradīsiet informāciju par to, kā izmantot mūsu lietotni un risināt biežāk sastopamās problēmas. Lietotāja un skolotāja izvēlnēs ir vairākas pogas, saraksti un tabulas, kas palīdz Jums viegli piekļūt nepieciešamajām funkcijām. Pogas ļauj Jums pārslēgties starp dažādiem ekrāniem, piemēram, testa izvēles ekrānu, atzīmju pārskata logu vai autorizācijas logu. Saraksti ļauj Jums izvēlēties dažādus testus vai citas opcijas. Atlasot konkrētu elementu no saraksta, ekrāna kreisajā pusē tiks parādīts detalizēts apraksts. Tabulas parāda dažādu informāciju, piemēram, skolēnu atzīmes. Skolotāji var arī mainīt vērtējumus tieši tabulā.

**BIEŽĀK UZDOTIE JAUTĀJUMI**

**Vai mani ievadītie dati tiks aizsargāti?**

Jā, jūsu ievadītie dati tiks aizsargāti atbilstoši visaugstākajiem drošības standartiem. Mūsu sistēma izmanto šifrēšanu, lai nodrošinātu, ka jūsu personiskā informācija un dati par testiem ir drošībā.

**Ko darīt, ja es saskaros ar autorizācijas problēmām?**

Pārliecinieties, ka Jūs ievadāt pareizu lietotājvārdu un paroli. Ja problēma saglabājas, mēģiniet atiestatīt paroli vai sazinieties ar administraciju. Turpmāk ir norādīta administrācijas kontaktinformācija.

**Kā pievienot vai izdzēst testu?**

Administrators var pievienot jaunu testu, noklikšķinot uz “Pievienot jaunu testu” un aizpildot nepieciešamos datus. Lai izdzēstu testu, izvēlieties testu no saraksta un noklikšķiniet uz “Izdzēst testu”.

**Kā es varu pieprasīt jaunu funkcionalitāti vai ziņot par kļūdu?**

Sazinieties ar administrāciju, norādot vēlamo funkcionalitāti vai detalizētu kļūdas aprakstu. Turpmāk ir norādīta administrācijas kontaktinformācija.

**Kā rīkoties, ja programma uzkaras vai darbojas lēni?**

Pārliecinieties, ka Jūsu ierīce atbilst sistēmas prasībām, un mēģiniet restartēt programmu. Ja problēma saglabājas, sazinieties ar administrāciju. Turpmāk ir norādīta administrācijas kontaktinformācija.

Administrācijas e-pasts: **nikitarulevics@gmail.com**

# Secinājumi

Projekts “Testēšanas sistēma” tika izstrādāts **NetBeans** vidē. Tika izpildītas visas programmatūras pamatprasības, pat tika pievienotas daudzas papildu funkcijas. Projekta gaitā radās dažādas problēmas, kas neļāva sasniegt sākotnēji vēlamo rezultātu, taču ar laiku tika atrasts kārtīgs risinājums, un bija interesanti uzzināt jaunu informāciju, kas noderēs nākotnē. Man īpaši patika strādāt ar **Derby** datubāzēm **NetBeans** vidē. Šajā vidē es uzzināju daudz jaunas un noderīgas informācijas, kā arī iemācījos pilnīgi jaunas darba metodes. Šī bija mana pirmā pieredze, izstrādājot tik liela mēroga projektu. Diemžēl ir dažas nerealizētas idejas, kas manu projektu būtu varējušas padarīt vēl labāku, taču briesmīgā laika trūkuma dēļ tām nav lemts tikt realizētām. Svarīgākais ir tas, ka pamatprasības ir īstenotas un viss darbojas pareizi un ļoti labi. Cerams, jo tālāk, jo labāk!

# Izmantotās literatūras (informācijas avotu) saraksts

1. **Zaiceva, L.** *Programmatūras izstrādes tehnoloģija.* Rīga : RTU, 2002.

2. **Kodors, Sergejs.** *IEVADS PRASĪBU INŽENIERIJĀ. Mācību līdzeklis.* Rēzekne : Rēzeknes Tehnoloģiju akadēmija, 2019. 978-9984-44-226-6.

3. **Taylor Johnson, Dung X. Nguyen.** UML and more JAVA Syntax. [Tiešsaiste] 2007. gada 16. 1. [Citēts: 2024. gada 9. 5.] https://www.clear.rice.edu/comp212/07-spring/labs/01.5/.

4. **Minkyu Lee, Hyunsoo Kim, Jeongil Kim, Jangwoo Lee, Deukkyu Gum.** StarUml 5.0 User Guide. [Tiešsaiste] 2005. gada. [Citēts: 2024. gada 10. 6.] https://staruml.sourceforge.net/docs/user-guide(en)/toc.html.

5. **Dortiņa, I. un Dauģerts, A.** 1.1.1. Ievads Java. Java pamati. [Tiešsaiste] 2022. gada 15. 2. [Citēts: 2024. gada 26. 5.] http://e.daugvt.lv/mod/resource/view.php?id=3321.

6. **—.** 1.1.2. Java sazarotas struktūras operātori. [Tiešsaiste] 2022. gada 21. 02. [Citēts: 2024. gada 03. 06.] http://e.daugvt.lv/mod/resource/view.php?id=3330.

7. —. 4.1.2. Java GUI izveidošana NetBeans vidē. [Tiešsaiste] 2023. gada 03. 03. [Citēts: 2024. gada 15. 05.] http://e.daugvt.lv/mod/resource/view.php?id=3870.

8. —. 4.1.1. Java. Grafiskais lietotāja interfeiss. [Tiešsaiste] 2023. gada 23. 02. [Citēts: 2024. gada 16. 05.]

9. —. 2.2. Java izņēmumi un to apstrāde. [Tiešsaiste] 2022. gada 24. 11. [Citēts: 2024. gada 02. 06.] http://e.daugvt.lv/mod/resource/view.php?id=3742.

10. —. 2.1.3. Java pakotnes un interfeisi. [Tiešsaiste] 2022. gada 04. 11. [Citēts: 2024. gada 06. 06.] http://e.daugvt.lv/mod/resource/view.php?id=3676.

11. —. 2.1.1.2. Java. Metodes un konstruktori. [Tiešsaiste] 2022. gada 16. 09. [Citēts: 2024. gada 29. 05.] http://e.daugvt.lv/mod/resource/view.php?id=3567.

12. —. 2.1.1.1. Java. OOP koncepcija. [Tiešsaiste] 2022. gada 01. 06. [Citēts: 2024. gada 20. 05.] http://e.daugvt.lv/mod/resource/view.php?id=3432.

13. —. 1.2.1.3. Java dinamiskie masīvi. [Tiešsaiste] 2022. gada 13. 04. [Citēts: 2024. gada 17. 05.] http://e.daugvt.lv/mod/resource/view.php?id=3398.

14. —. 1.2.2. Java ieraksti. [Tiešsaiste] 2022. gada 27. 04. [Citēts: 2024. gada 05. 30.] http://e.daugvt.lv/mod/resource/view.php?id=3408.

15. —. 1.2.3. Java Ievades/izvades sistēma. [Tiešsaiste] 2022. gada 11. 05. [Citēts: 2024. gada 11. 05.] http://e.daugvt.lv/mod/resource/view.php?id=3419.

16. —. 1.2.1.2. Java rakstzīmju masīvi un rakstzīmju virknes. [Tiešsaiste] 2022. gada 06. 04. [Citēts: 2024. gada 17. 05.] http://e.daugvt.lv/mod/resource/view.php?id=3383.

17. —. 1.2.1.1. Java masīvi. [Tiešsaiste] 2022. gada 25. 03. [Citēts: 2024. gada 25. 05.] http://e.daugvt.lv/mod/resource/view.php?id=3347.

18. —. 1.1.3. Java cikliskās struktūras operatori. [Tiešsaiste] 2022. gada 01. 03. [Citēts: 2024. gada 09. 05.] http://e.daugvt.lv/mod/resource/view.php?id=3340.

19. **Kultūras informācijas sistēmu centrs.** Latvijas Nacionālais terminoloģijas portals. [Tiešsaiste] 2005-2024. gada. [Citēts: 2024. gada 10. 06.] https://termini.gov.lv/.

20. **Samsotha, Paul.** java pass data to JTable's row upon button click. *stackoverflow.* [Tiešsaiste] 2014. gada 07. 01. [Citēts: 2024. gada 11. 06.] https://stackoverflow.com/questions/20967741/java-pass-data-to-jtables-row-upon-button-click.

21. **Zukowski, John.** *Definitive Guide to Swing for Java 2.* bez viet. : APress, 2000. 189311578X.

22. **splungebob.** Removing items from JList. *stackoverflow.* [Tiešsaiste] 2012. gada 08. 11. [Citēts: 2024. gada 01. 06.] https://stackoverflow.com/questions/13298711/removing-items-from-jlist.

23. **Yuan, Kai.** Make Division of Two Integers Result in a Float . *Baeldung.* [Tiešsaiste] 2024. gada 08. 01. [Citēts: 2024. gada 11. 06.] https://www.baeldung.com/java-integer-division-float-result.

# Pielikums A. Klašu sistēma

User klase:

public class User {

String name;

String surname;

String username;

String password;

String usertype;

public User(String name, String surname, String username, String password, String usertype) {

this.name = name;

this.surname = surname;

this.username = username;

this.password = password;

this.usertype = usertype;

}

public User(String name, String username, String password, String usertype) {

this.name = name;

this.username = username;

this.password = password;

this.usertype = usertype;

}

}

Teacher klase:

public class Teacher extends User {

public Teacher(String name, String surname, String username, String password, String usertype) {

super(name, surname, username, password, usertype);

}

public Teacher(String name, String username, String password, String usertype){

super(name, username, password, usertype);

}

public void populateMarksTable(JTable jTable4) {

DataBase db = new DataBase();

ResultSet rs = null, rs2 = null;

PreparedStatement pst = null, pst2 = null;

try {

Connection con = db.connect();

String sql = "SELECT UserName, TestName, Mark FROM marks";

pst = con.prepareStatement(sql);

rs = pst.executeQuery();

DefaultTableModel model = (DefaultTableModel) jTable4.getModel();

model.setRowCount(0);

while (rs.next()) {

String userName = rs.getString("UserName");

String testName = rs.getString("TestName");

String mark = rs.getString("Mark");

sql = "SELECT FirstName, LastName FROM users WHERE UserName=? ";

pst2 = con.prepareStatement(sql);

pst2.setString(1, userName);

rs2 = pst2.executeQuery();

while (rs2.next()) {

String firstName = rs2.getString("FirstName");

String lastName = rs2.getString("LastName");

model.addRow(new Object[]{firstName, lastName, testName, mark});

}

rs2.close();

pst2.close();

}

jTable4.setModel(model);

con.close();

} catch (Exception e) {

System.out.println(e.getMessage());

}

}

public void saveNewMarks(JTable jTable4){

DataBase db = new DataBase();

Connection con = null;

PreparedStatement pst = null;

boolean hasInvalidMarks = false;

Logu\_Redaktors loguRedaktors = new Logu\_Redaktors();

try {

con = db.connect();

DefaultTableModel model = (DefaultTableModel) jTable4.getModel();

int rowCount = model.getRowCount();

String sql = "UPDATE marks SET Mark=? WHERE UserName=? AND TestName=?";

pst = con.prepareStatement(sql);

for (int i = 0; i < rowCount; i++) {

String mark = (String) model.getValueAt(i, 3);

try {

int markValue = Integer.parseInt(mark);

if (markValue < 1 || markValue > 10) {

hasInvalidMarks = true;

String firstName = (String) model.getValueAt(i, 0);

String lastName = (String) model.getValueAt(i, 1);

loguRedaktors.showError("Nepareizs vērtējums par lietotāju - \"" + firstName + " " + lastName + "\".");

}

} catch (NumberFormatException e) {

hasInvalidMarks = true;

String firstName = (String) model.getValueAt(i, 0);

String lastName = (String) model.getValueAt(i, 1);

loguRedaktors.showError("Nepareizs vērtējums par lietotāju - \"" + firstName + " " + lastName + "\".");

}

}

if (!hasInvalidMarks) {

for (int i = 0; i < rowCount; i++) {

String firstName = (String) model.getValueAt(i, 0);

String lastName = (String) model.getValueAt(i, 1);

String testName = (String) model.getValueAt(i, 2);

String mark = (String) model.getValueAt(i, 3);

String userName = loguRedaktors.getUserName(con, firstName, lastName);

pst.setString(1, mark);

pst.setString(2, userName);

pst.setString(3, testName);

pst.executeUpdate();

}

loguRedaktors.showInfo("Jūs esat veiksmīgi nomainījuši atzīmes!");

}

} catch (Exception e) {

System.out.println(e.getMessage());

}

populateMarksTable(jTable4);

}

public void deleteMark(JTable jTable4) {

Logu\_Redaktors loguRedaktors = new Logu\_Redaktors();

int selectedRow = jTable4.getSelectedRow();

if (selectedRow != -1) {

DefaultTableModel model = (DefaultTableModel) jTable4.getModel();

String firstName = (String) model.getValueAt(selectedRow, 0);

String lastName = (String) model.getValueAt(selectedRow, 1);

String testName = (String) model.getValueAt(selectedRow, 2);

DataBase db = new DataBase();

Connection con = null;

PreparedStatement pst = null;

try {

con = db.connect();

String userName = loguRedaktors.getUserName(con, firstName, lastName);

String sql = "DELETE FROM marks WHERE UserName=? AND TestName=?";

pst = con.prepareStatement(sql);

pst.setString(1, userName);

pst.setString(2, testName);

pst.executeUpdate();

model.removeRow(selectedRow);

loguRedaktors.showInfo("Vērtējums ir veiksmīgi izdzēsts!");

} catch (Exception e) {

System.out.println(e.getMessage());

}

} else {

loguRedaktors.showError("Nav izvēlēta neviena rinda!");

}

}

public void deleteTest(JList<String> jList2, JTextArea TestDescriptionTextArea1) {

String selectedTest = jList2.getSelectedValue();

Logu\_Redaktors loguRedaktors = new Logu\_Redaktors();

if (selectedTest != null) {

DataBase db = new DataBase();

Connection con = null;

PreparedStatement pst = null;

try {

con = db.connect();

String sql = "DELETE FROM test WHERE TestName = ?";

pst = con.prepareStatement(sql);

pst.setString(1, selectedTest);

pst.executeUpdate();

con.close();

DefaultListModel<String> model = (DefaultListModel<String>) jList2.getModel();

model.removeElement(selectedTest);

loguRedaktors.showInfo("Tests ir veiksmīgi izdzēsts!");

TestDescriptionTextArea1.setText("");

} catch (Exception e) {

System.out.println(e.getMessage());

}

}else{

loguRedaktors.showError("Tests nav izvelēts!");

}

}

public void deleteStudent(JTable JTable) {

Logu\_Redaktors loguRedaktors = new Logu\_Redaktors();

int selectedRow = JTable.getSelectedRow();

if (selectedRow != -1) {

DefaultTableModel model = (DefaultTableModel) JTable.getModel();

String userName = (String) model.getValueAt(selectedRow, 2);

DataBase db = new DataBase();

Connection con = null;

PreparedStatement pst = null, pst2 = null;

try {

con = db.connect();

String deleteMarksSql = "DELETE FROM marks WHERE UserName=?";

pst = con.prepareStatement(deleteMarksSql);

pst.setString(1, userName);

pst.executeUpdate();

pst.close();

String deleteUserSql = "DELETE FROM users WHERE UserName=? ";

pst2 = con.prepareStatement(deleteUserSql);

pst2.setString(1, userName);

pst2.executeUpdate();

pst2.close();

loguRedaktors.showInfo("Lietotājs ir izdzēsts!");

model.removeRow(selectedRow);

} catch (Exception e) {

}

} else {

loguRedaktors.showError("Nav izvēlēta neviena rinda!");

}

}

public void populateStudentTable(JTable JTable) {

DataBase db = new DataBase();

ResultSet rs = null;

PreparedStatement pst = null;

try {

Connection con = db.connect();

String sql = "SELECT FirstName, LastName, UserName, Password, UserType FROM users";

pst = con.prepareStatement(sql);

rs = pst.executeQuery();

DefaultTableModel model = (DefaultTableModel) JTable.getModel();

model.setRowCount(0);

while (rs.next()) {

String firstName = rs.getString("FirstName");

String lastName = rs.getString("LastName");

String userName = rs.getString("UserName");

String password = rs.getString("Password");

String userType = rs.getString("UserType");

if(userType.equals("teacher")){

continue;

}else{

model.addRow(new Object[]{firstName, lastName, userName, password});

}

}

JTable.setModel(model);

con.close();

} catch (Exception e) {

System.out.println(e.getMessage());

}

}

public void saveNewStudent(JTable jTable) {

DataBase db = new DataBase();

Connection con = null;

PreparedStatement pst = null;

Logu\_Redaktors loguRedaktors = new Logu\_Redaktors();

DefaultTableModel model = (DefaultTableModel) jTable.getModel();

int rowCount = model.getRowCount();

try {

con = db.connect();

con.setAutoCommit(false); // Begin transaction

for (int i = 0; i < rowCount; i++) {

String firstName = (String) model.getValueAt(i, 0);

String lastName = (String) model.getValueAt(i, 1);

String userName = (String) model.getValueAt(i, 2);

String password = (String) model.getValueAt(i, 3);

// Check if the user already exists

String checkSql = "SELECT COUNT(\*) FROM users WHERE UserName=?";

pst = con.prepareStatement(checkSql);

pst.setString(1, userName);

ResultSet rs = pst.executeQuery();

rs.next();

int count = rs.getInt(1);

pst.close();

if (count > 0) {

String updateSql = "UPDATE users SET FirstName=?, LastName=?, Password=?, UserType=? WHERE UserName=?";

pst = con.prepareStatement(updateSql);

pst.setString(1, firstName);

pst.setString(2, lastName);

pst.setString(3, password);

pst.setString(4, "student");

pst.setString(5, userName);

pst.executeUpdate();

pst.close();

} else {

// User does not exist, insert a new record

String insertSql = "INSERT INTO users (FirstName, LastName, UserName, Password, UserType) VALUES (?, ?, ?, ?, ?)";

pst = con.prepareStatement(insertSql);

pst.setString(1, firstName);

pst.setString(2, lastName);

pst.setString(3, userName);

pst.setString(4, password);

pst.setString(5, "student");

pst.executeUpdate();

pst.close();

}

}

con.commit();

con.setAutoCommit(true);

loguRedaktors.showInfo("Izmaiņas veiksmīgi saglabātas!");

} catch (Exception e) {

System.out.println(e.toString());

}

}

}

Student klase:

public class Student extends User {

public Student(String name, String surname, String username, String password, String usertype) {

super(name, surname, username, password, usertype);

}

public Student(String name, String username, String password, String usertype) {

super(name, username, password, usertype);

}

public void displayMarks(JTable jTable1) {

DataBase db = new DataBase();

ResultSet rs = null;

PreparedStatement pst = null;

try {

Connection con = db.connect();

String sql = "SELECT TestName, Mark FROM marks WHERE UserName=?";

pst = con.prepareStatement(sql);

pst.setString(1, this.username);

rs = pst.executeQuery();

DefaultTableModel model = (DefaultTableModel) jTable1.getModel();

model.setRowCount(0);

while (rs.next()) {

String testName = rs.getString("TestName");

String mark = rs.getString("Mark");

model.addRow(new Object[]{testName, mark});

}

jTable1.setModel(model);

con.close();

} catch (Exception e) {

System.out.println(e.getMessage());

}

}

public void displayTest(String testName, JLabel TestNameLabel, JLabel[] questionLabels, JRadioButton[][] answerButtons) {

DataBase db = new DataBase();

ResultSet rs = null;

PreparedStatement pst = null;

if (testName == null) {

Logu\_Redaktors loguRedaktors = new Logu\_Redaktors();

loguRedaktors.showError("Tests nav izvelēts!");

return;

}

try {

Connection con = db.connect();

TestNameLabel.setText(testName);

String sql = "SELECT QuestionText, AnswerFirst, AnswerSecond, AnswerThird FROM test WHERE TestName=?";

pst = con.prepareStatement(sql);

pst.setString(1, testName);

rs = pst.executeQuery();

int questionIndex = 0;

while (rs.next() && questionIndex < 5) {

String questionText = rs.getString("QuestionText");

String answerFirst = rs.getString("AnswerFirst");

String answerSecond = rs.getString("AnswerSecond");

String answerThird = rs.getString("AnswerThird");

questionLabels[questionIndex].setText(questionText);

answerButtons[questionIndex][0].setText(answerFirst);

answerButtons[questionIndex][1].setText(answerSecond);

answerButtons[questionIndex][2].setText(answerThird);

questionIndex++;

}

con.close();

} catch (Exception e) {

System.out.println(e.getMessage());

}

}

public float[] submitTest(ButtonGroup[] answerGroups, String testName, User currentUser) {

DataBase db = new DataBase();

PreparedStatement pst = null;

ResultSet rs = null;

int pointsForTest = 0;

float markForTest = 0, percentsForTest = 0;

int i = 0;

Logu\_Redaktors loguRedaktors = new Logu\_Redaktors();

try {

Connection con = db.connect();

String sql = "SELECT AnswerRight FROM test WHERE TestName=?";

pst = con.prepareStatement(sql);

pst.setString(1, testName);

rs = pst.executeQuery();

while (rs.next() && i < 5) {

String answerRight = rs.getString("AnswerRight");

if (loguRedaktors.getSelectedButtonText(answerGroups[i]) == null) {

pointsForTest += 0;

} else if (loguRedaktors.getSelectedButtonText(answerGroups[i]).equals(answerRight)) {

pointsForTest++;

}

i++;

}

percentsForTest = ((float) pointsForTest / 5) \* 100;

switch (pointsForTest) {

case 0: {

markForTest = 1;

break;

}

case 1: {

markForTest = 2;

break;

}

case 2: {

markForTest = 3;

break;

}

case 3: {

markForTest = 6;

break;

}

case 4: {

markForTest = 8;

break;

}

default: {

markForTest = 10;

}

}

String sqlCheck = "SELECT \* FROM marks WHERE UserName=? AND TestName=? ";

pst = con.prepareStatement(sqlCheck);

pst.setString(1, currentUser.username);

pst.setString(2, testName);

rs = pst.executeQuery();

if (rs.next()) {

loguRedaktors.showError("Jūms jau ir atzīme par šo testu!");

} else {

sql = "INSERT INTO marks VALUES(?, ?, ?)";

pst = con.prepareStatement(sql);

pst.setString(1, currentUser.username);

pst.setString(2, testName);

pst.setString(3, Float.toString(markForTest));

pst.executeUpdate();

}

con.close();

} catch (SQLException ex) {

Logger.getLogger(Logu\_Redaktors.class.getName()).log(Level.SEVERE, null, ex);

}

return new float[] {percentsForTest, markForTest};

}

}

Test klase:

public class Test {

String testName;

public void addListSelectionListeners(JList jList1, JList jList2, JTextArea testDescriptionTextArea, JTextArea testDescriptionTextArea1) {

jList1.addListSelectionListener(new ListSelectionListener() {

@Override

public void valueChanged(ListSelectionEvent e) {

if (!e.getValueIsAdjusting()) {

testName = (String) jList1.getSelectedValue();

if (testName != null) {

updateTestDescription(testDescriptionTextArea, testName);

}

}

}

});

jList2.addListSelectionListener(new ListSelectionListener() {

@Override

public void valueChanged(ListSelectionEvent e) {

if (!e.getValueIsAdjusting()) {

testName = (String) jList2.getSelectedValue();

if (testName != null) {

updateTestDescription(testDescriptionTextArea1, testName);

}

}

}

});

}

public void loadTests(JList jList1, JList jList2, DefaultListModel listModel) {

DataBase db = new DataBase();

try {

Connection con = db.connect();

Statement stm = con.createStatement();

ResultSet rs = stm.executeQuery("SELECT \* FROM test");

Set<String> uniqueTestNames = new HashSet<>();

while (rs.next()) {

String testName = rs.getString("TestName");

if (uniqueTestNames.add(testName)) {

listModel.addElement(testName);

}

}

jList1.setModel(listModel);

jList2.setModel(listModel);

con.close();

} catch (Exception e) {

System.out.println(e.getMessage());

}

}

public void updateTestDescription(JTextArea textArea, String testName) {

DataBase db = new DataBase();

try {

Connection con = db.connect();

String sql = "SELECT TestDescription FROM test WHERE TestName=?";

PreparedStatement pst = con.prepareStatement(sql);

pst.setString(1, testName);

ResultSet rs = pst.executeQuery();

if (rs.next()) {

String testDescription = rs.getString("TestDescription");

textArea.setText(testDescription);

}

con.close();

} catch (Exception e) {

System.out.println(e.getMessage());

}

}

}

# Pielikums B. testa programmas kods

public class Logu\_Redaktors extends javax.swing.JFrame {

static Logu\_Redaktors LoginFrame = new Logu\_Redaktors();

User currentUser;

Test test = new Test();

/\*\*

\* Creates new form Logu\_Redaktors

\*/

public Logu\_Redaktors() {

initComponents();

}

/\*\*

\* This method is called from within the constructor to initialize the form.

\* WARNING: Do NOT modify this code. The content of this method is always

\* regenerated by the Form Editor.

\*/

// Ienākšana reģistrācijas logā

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {

RegisterFrame.setVisible(true);

RegisterFrame.pack();

RegisterFrame.setLocationRelativeTo(null);

this.dispose();

}

// Reģistrēšanas funkcija

private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {

String passwordRepeat = RegPasswordRepeatField.getText();

String usertype = "";

if (StudentTypeRadioButton.isSelected()) {

usertype = "student";

} else if (TeacherTypeRadioButton.isSelected()) {

usertype = "teacher";

}

User NewUser = new User(RegNameField.getText(), RegSurnameField.getText(), RegUsernameField.getText(), RegPasswordField.getText(), usertype);

DataBase db = new DataBase();

if (NewUser.name.equals("") || NewUser.surname.equals("") || NewUser.username.equals("") || NewUser.password.equals("") || passwordRepeat.equals("") || usertype.equals("")) {

showError("Ne visas ailītes ir aizpildītas!");

} else if (!NewUser.password.equals(passwordRepeat)) {

showError("Paroles nesakrīt!");

} else {

try {

Connection con = db.connect();

String sqlCheck = "SELECT \* FROM users WHERE UserName=?";

PreparedStatement pst = con.prepareStatement(sqlCheck);

pst.setString(1, NewUser.username);

ResultSet rs = pst.executeQuery();

if (rs.next()) {

showError("Jau pastāv lietotājs ar šo lietotājvārdu!");

} else {

String sql = "INSERT INTO users VALUES(?, ?, ?, ?, ?)";

pst = con.prepareStatement(sql);

pst.setString(1, NewUser.name);

pst.setString(2, NewUser.surname);

pst.setString(3, NewUser.username);

pst.setString(4, NewUser.password);

pst.setString(5, usertype);

pst.executeUpdate();

RegisterFrame.dispose();

this.show();

clearRegistrationFields();

con.close();

showInfo("Jūs esat veiksmīgi reģistrējies!");

}

} catch (Exception e) {

e.printStackTrace();

}

}

}

// Ielogošanas funkcija

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {

DataBase db = new DataBase();

String username = UsernameField.getText();

String password = PasswordField.getText();

ResultSet rs = null;

PreparedStatement pst = null;

if (username.equals("") || password.equals("")) {

showError("Ne visas ailītes ir aizpildītas");

} else {

// connection

try {

Connection con = db.connect();

String sql = "SELECT \* FROM users WHERE UserName=? AND Password=?";

pst = con.prepareStatement(sql);

pst.setString(1, username);

pst.setString(2, password);

rs = pst.executeQuery();

if (rs.next()) {

String userType = rs.getString("UserType");

String FirstName = rs.getString("FirstName");

if ("teacher".equals(userType)) {

MainFrameTeacher.setVisible(true);

MainFrameTeacher.pack();

MainFrameTeacher.setLocationRelativeTo(null);

jLabel21.setText("Labdien, " + FirstName);

currentUser = new Teacher(FirstName, username, password, userType);

} else {

MainFrameStudent.setVisible(true);

MainFrameStudent.pack();

MainFrameStudent.setLocationRelativeTo(null);

// Izmainisīm tekstu uz saveicinājumu ar lietotāju

jLabel13.setText("Labdien, " + FirstName);

currentUser = new Student(FirstName, username, password, userType);

}

this.dispose();

showInfo("Jūs esat veiksmīgi pieteicies savā kontā!");

} else {

showError("Nevarēja piekļūt kontam!");

}

UsernameField.setText("");

PasswordField.setText("");

con.close();

} catch (Exception e) {

System.out.println(e.getMessage());

}

}

test.addListSelectionListeners(jList1, jList2, TestDescriptionTextArea, TestDescriptionTextArea1);

test.loadTests(jList1, jList2, new DefaultListModel());

}

// Funkcija rezultātu lejupielādei no datubāzes

private void jButton7ActionPerformed(java.awt.event.ActionEvent evt) {

jTabbedPane1.setSelectedIndex(1);

if (currentUser instanceof Student) {

((Student) currentUser).displayMarks(jTable1);

}

}

// Funkcija rezultātu lejupielādei no datubāzes

private void jButton8ActionPerformed(java.awt.event.ActionEvent evt) {

jTabbedPane1.setSelectedIndex(0);

}

// Konta izrakstīšanās funkcija

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

LoginFrame.setVisible(true);

MainFrameStudent.dispose();

TestDescriptionTextArea.setText("");

jTabbedPane1.setSelectedIndex(0);

}

// Kļūdu loga aizvēršanas funkcija

private void jButton18ActionPerformed(java.awt.event.ActionEvent evt) {

ErrorFrame.dispose();

}

private void jButton20ActionPerformed(java.awt.event.ActionEvent evt) {

LoginFrame.setVisible(true);

RegisterFrame.dispose();

}

private void jButton21ActionPerformed(java.awt.event.ActionEvent evt) {

InfoFrame.dispose();

}

// Testa pieladēšana no Datu bāzes

private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {

DataBase db = new DataBase();

ResultSet rs = null;

PreparedStatement pst = null;

String testName = jList1.getSelectedValue();

JLabel[] questionLabels = {QuestionLabel1, QuestionLabel2, QuestionLabel3, QuestionLabel4, QuestionLabel5};

JRadioButton[][] answerButtons = {

{Answer11, Answer12, Answer13},

{Answer21, Answer22, Answer23},

{Answer31, Answer32, Answer33},

{Answer41, Answer42, Answer43},

{Answer51, Answer52, Answer53}

};

// Ja neviens tests nav izvelets

if (testName == null) {

showError("Tests nav izvelēts!");

return;

}

if (currentUser instanceof Student) {

((Student) currentUser).displayTest(testName, TestNameLabel, questionLabels, answerButtons);

}

MainFrameStudent.dispose();

MainFrameTeacher.dispose();

TestFrame.setVisible(true);

TestFrame.pack();

TestFrame.setLocationRelativeTo(null);

}

// Testu atbilžu ielādēšana un pārbaude

private void jButton16ActionPerformed(java.awt.event.ActionEvent evt) {

ButtonGroup[] answerGroups = {AnswerGroup1, AnswerGroup2, AnswerGroup3, AnswerGroup4, AnswerGroup5};

String testName = jList1.getSelectedValue();

if (currentUser instanceof Student) {

float[] Marks = ((Student)currentUser).submitTest(answerGroups, testName, currentUser);

showResult(Marks[0],(int)Marks[1]);

}

}

// Dzēst visas atlasītās atbildes

private void jButton15ActionPerformed(java.awt.event.ActionEvent evt) {

TestFrame.dispose();

MainFrameStudent.show();

ButtonGroup[] answerGroups = {AnswerGroup1, AnswerGroup2, AnswerGroup3, AnswerGroup4, AnswerGroup5};

for(ButtonGroup currentGroup : answerGroups){

currentGroup.clearSelection();

}

}

// Dzēst visas atlasītās atbildes

private void jButton26ActionPerformed(java.awt.event.ActionEvent evt) {

ButtonGroup[] answerGroups = {AnswerGroup1, AnswerGroup2, AnswerGroup3, AnswerGroup4, AnswerGroup5};

for(ButtonGroup currentGroup : answerGroups){

currentGroup.clearSelection();

}

TestFrame.dispose();

ResultFrame.dispose();

MainFrameStudent.show();

}

// Dzēst visas atlasītās atbildes

private void jButton25ActionPerformed(java.awt.event.ActionEvent evt) {

ButtonGroup[] answerGroups = {AnswerGroup1, AnswerGroup2, AnswerGroup3, AnswerGroup4, AnswerGroup5};

for(ButtonGroup currentGroup : answerGroups){

currentGroup.clearSelection();

}

}

// Vērtēšanas tabulas aizpildīšanas funkcija

private void jButton27ActionPerformed(java.awt.event.ActionEvent evt) {

jTabbedPane2.setSelectedIndex(1);

if (currentUser instanceof Teacher) {

((Teacher)currentUser).populateMarksTable(jTable4);

}

}

// Konta izrakstīšanās funkcija

private void jButton28ActionPerformed(java.awt.event.ActionEvent evt) {

LoginFrame.setVisible(true);

MainFrameTeacher.dispose();

TestDescriptionTextArea1.setText("");

jTabbedPane2.setSelectedIndex(0);

}

// Funkcijas atkārtošana testa saraksta ielādei

private void jButton29ActionPerformed(java.awt.event.ActionEvent evt) {

test.addListSelectionListeners(jList1, jList2, TestDescriptionTextArea, TestDescriptionTextArea1);

test.loadTests(jList1, jList2, new DefaultListModel());

jTabbedPane2.setSelectedIndex(0);

}

// Testa izdzēšanas funkcija

private void jButton30ActionPerformed(java.awt.event.ActionEvent evt) {

if (currentUser instanceof Teacher) {

((Teacher)currentUser).deleteTest(jList2, TestDescriptionTextArea1);

}

}

// Atzīmes labošanas funkcija

private void jButton32ActionPerformed(java.awt.event.ActionEvent evt) {

if (jTable4.isEditing()) {

jTable4.getCellEditor().stopCellEditing();

}

if (currentUser instanceof Teacher) {

((Teacher)currentUser).saveNewMarks(jTable4);

}

}

// Atzīmes izdzēšanas funkcija

private void jButton33ActionPerformed(java.awt.event.ActionEvent evt) {

if (currentUser instanceof Teacher) {

((Teacher)currentUser).deleteMark(jTable4);

}

}

private void jButton34ActionPerformed(java.awt.event.ActionEvent evt) {

NewTestFrame.setVisible(true);

NewTestFrame.pack();

NewTestFrame.setLocationRelativeTo(null);

MainFrameTeacher.dispose();

}

// Jauna testa pievienošana

private void jButton19ActionPerformed(java.awt.event.ActionEvent evt) {

DataBase db = new DataBase();

Connection con = null;

PreparedStatement pst = null;

try {

con = db.connect();

String testName = TestNameField.getText();

String testDescription = NewTestDesriptionField.getText();

String sql = "INSERT INTO test (TestName, QuestionText, TestDescription, AnswerFirst, AnswerSecond, AnswerThird, AnswerRight) VALUES (?, ?, ?, ?, ?, ?, ?)";

pst = con.prepareStatement(sql);

for (int i = 1; i <= 5; i++) {

String questionText = ((JTextField) this.getClass().getDeclaredField("NewQuestionField" + i).get(this)).getText();

String answerFirst = ((JTextField) this.getClass().getDeclaredField("NewAnswerField" + i + "1").get(this)).getText();

String answerSecond = ((JTextField) this.getClass().getDeclaredField("NewAnswerField" + i + "2").get(this)).getText();

String answerThird = ((JTextField) this.getClass().getDeclaredField("NewAnswerField" + i + "3").get(this)).getText();

String answerRight = null;

for (int j = 1; j <= 3; j++) {

JRadioButton radioButton = (JRadioButton) this.getClass().getDeclaredField("RadioButton" + i + j).get(this);

if (radioButton.isSelected()) {

answerRight = ((JTextField) this.getClass().getDeclaredField("NewAnswerField" + i + j).get(this)).getText();

break;

}

}

if (answerRight == null) {

showError("Lūdzu, izvēlieties pareizo atbildi jautājumam " + i);

return;

}

pst.setString(1, testName);

pst.setString(2, questionText);

pst.setString(3, testDescription);

pst.setString(4, answerFirst);

pst.setString(5, answerSecond);

pst.setString(6, answerThird);

pst.setString(7, answerRight);

pst.addBatch();

}

pst.executeBatch();

NewTestFrame.dispose();

MainFrameTeacher.setVisible(true);

MainFrameTeacher.pack();

MainFrameTeacher.setLocationRelativeTo(null);

showInfo("Tests veiksmīgi pievienots!");

} catch (Exception e) {

System.out.println("Radās kļūda: " + e.getMessage());

}

try {

DefaultListModel listModel = new DefaultListModel();

con = db.connect();

Statement stm = con.createStatement();

ResultSet rs = stm.executeQuery("SELECT \* FROM test");

//

Set<String> uniqueTestNames = new HashSet<>();

while (rs.next()) {

String testName = rs.getString("TestName");

if (uniqueTestNames.add(testName)) { //

listModel.addElement(testName);

}

}

jList2.setModel(listModel);

con.close();

} catch (Exception e) {

System.out.println(e.getMessage());

}

}

private void jButton22ActionPerformed(java.awt.event.ActionEvent evt) {

NewTestFrame.dispose();

MainFrameTeacher.setVisible(true);

MainFrameTeacher.pack();

MainFrameTeacher.setLocationRelativeTo(null);

}

private void jButton6ActionPerformed(java.awt.event.ActionEvent evt) {

MainFrameStudent.dispose();

HelpFrame.setVisible(true);

HelpFrame.pack();

HelpFrame.setLocationRelativeTo(null);

}

private void jButton9ActionPerformed(java.awt.event.ActionEvent evt) {

MainFrameTeacher.dispose();

HelpFrame.setVisible(true);

HelpFrame.pack();

HelpFrame.setLocationRelativeTo(null);

}

private void jButton17ActionPerformed(java.awt.event.ActionEvent evt) {

HelpFrame.dispose();

if ("teacher".equals(currentUser.usertype)) {

MainFrameTeacher.setVisible(true);

MainFrameTeacher.pack();

MainFrameTeacher.setLocationRelativeTo(null);

} else {

MainFrameStudent.setVisible(true);

MainFrameStudent.pack();

MainFrameStudent.setLocationRelativeTo(null);

}

}

// Jauna studenta pievienošana

private void jButton35ActionPerformed(java.awt.event.ActionEvent evt) {

DefaultTableModel model = (DefaultTableModel)jTable5.getModel();

model.addRow(new Object[]{"", "", "", ""});

}

// Studenta izdzēšanas funkcija

private void jButton36ActionPerformed(java.awt.event.ActionEvent evt) {

if (currentUser instanceof Teacher) {

((Teacher)currentUser).deleteStudent(jTable5);

}

}

// Studenta tabulas aizpildīšanas funkcija

private void jButton31ActionPerformed(java.awt.event.ActionEvent evt) {

jTabbedPane2.setSelectedIndex(2);

if (currentUser instanceof Teacher) {

((Teacher)currentUser).populateStudentTable(jTable5);

}

}

// Studentu saraksta saglabāšanas funkcija

private void jButton37ActionPerformed(java.awt.event.ActionEvent evt) {

if (jTable5.isEditing()) {

jTable5.getCellEditor().stopCellEditing();

}

if (currentUser instanceof Teacher) {

((Teacher)currentUser).saveNewStudent(jTable5);

}

}

/\*\*

\* @param args the command line arguments

\*/

public static void main(String args[]) {

LoginFrame.setVisible(true);

LoginFrame.pack();

LoginFrame.setLocationRelativeTo(null);

}

public void showError(String message) {

ErrorFrame.setVisible(true);

ErrorFrame.pack();

ErrorFrame.setLocationRelativeTo(null);

ErrorLabel.setText(message);

}

public void showInfo(String message) {

InfoFrame.setVisible(true);

InfoFrame.pack();

InfoFrame.setLocationRelativeTo(null);

InfoLabel.setText(message);

}

public void showResult(float percent, int mark) {

ResultFrame.setVisible(true);

ResultFrame.pack();

ResultFrame.setLocationRelativeTo(null);

NameLabel.setText("Apsveicām, " + currentUser.name);

PercentLabel.setText("Par testu \"" + jList1.getSelectedValue() + "\" jūs ieguvāt " + String.format("%.2f", percent) + "% ");

MarkLabel.setText("Jūsu atzīme ir " + mark + "!");

}

public void clearRegistrationFields() {

RegNameField.setText("");

RegSurnameField.setText("");

RegUsernameField.setText("");

RegPasswordField.setText("");

RegPasswordRepeatField.setText("");

buttonGroup7.clearSelection();

}

public String getSelectedButtonText(ButtonGroup buttonGroup) {

for (Enumeration<AbstractButton> buttons = buttonGroup.getElements(); buttons.hasMoreElements();) {

AbstractButton button = buttons.nextElement();

if (button.isSelected()) {

return button.getText();

}

}

return null;

}

String getUserName(Connection con, String firstName, String lastName) throws SQLException {

String sql = "SELECT UserName FROM users WHERE FirstName=? AND LastName=?";

PreparedStatement pst = con.prepareStatement(sql);

pst.setString(1, firstName);

pst.setString(2, lastName);

ResultSet rs = pst.executeQuery();

String userName = null;

if (rs.next()) {

userName = rs.getString("UserName");

}

rs.close();

pst.close();

return userName;

}