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.0012.012PR**

Profesijas kods 33 481 031 Programmēšanas nodaļa

PR-21. grupa

|  |  |  |
| --- | --- | --- |
| Izveidoja | paraksts | D.Kižlo  V.,Uzvārds |
| Pārbaudīja | paraksts | V. Uzvārds  V.,Uzvārds |

2022./2023. m.g.

# Saturs

[IEVADS 3](#_bookmark0)

1. [UZDEVUMA NOSTĀDNE 4](#_bookmark1)
   1. [SISTĒMAS MODELIS 4](#_bookmark2)
      1. [Priekšmetiskās jomas informācijas modelis 4](#_bookmark3)
      2. [Sistēmas apkārtnes shēma 4](#_bookmark5)
      3. [Viedokļu analīze 5](#_bookmark7)
      4. [Datu modelēšana 6](#_bookmark10)
      5. [Datu vārdnīca 6](#_bookmark12)
   2. [SISTĒMAS EVOLŪCIJA 7](#_bookmark13)
   3. [SISTĒMAS FUNKCIONĀLĀS PRASĪBAS 7](#_bookmark14)
   4. [SISTĒMAS NEFUNKCIONĀLĀS PRASĪBAS 9](#_bookmark15)
2. [PRIEKŠMETISKĀS JOMAS KLAŠU SISTĒMAS IZSTRĀDE 12](#_bookmark19)
   1. [KLAŠU DIAGRAMMA 12](#_bookmark20)
   2. [KLAŠU REALIZĒŠANA JAVA VALODĀ 13](#_bookmark22)
3. [TESTA PROGRAMMAS IZSTRĀDE 14](#_bookmark23)
   1. [TESTĒŠANAS METODIKAS 14](#_bookmark24)
   2. [TESTĒŠANAS PROGRAMMAS STRUKTŪRA 14](#_bookmark25)
   3. [TESTĒŠANAS REZULTĀTI 15](#_bookmark27)
4. [LIETOTĀJA ROKASGRĀMATA 16](#_bookmark28)
   1. [LIETOTĀJA INSTRUKCIJA 16](#_bookmark29)
   2. [PALĪDZĪBAS SISTĒMA (HELP) 16](#_bookmark30)

[SECINĀJUMI 17](#_bookmark31)

[IZMANTOTĀS LITERATŪRAS (INFORMĀCIJAS AVOTU) SARAKSTS 18](#_bookmark32)

[PIELIKUMS A. KLAŠU SISTĒMA 20](#_bookmark33)

[PIELIKUMS B. TESTA PROGRAMMAS KODS 21](#_bookmark34)

# Ievads

Projekta darbā ir aprakstīta testēšanas programma, kura izpilda sekojošas funkcijas:

* lietotāju reģistrēšana
* datu ievade
* datu saglabāšana
* datu rediģēšana
* testa pildīšana
* rezultātu aprēķināšana

Sistēma nodrošina audzēkņa testēšanu par noteiktu tēmu/vielu, tas ir domāts, lai skolotājs varētu viegli izlikt atzīmi skolēnam digitālā veidā attālināti vai klātienē. Projekta darbības princips būs šāds: klients (students) ievada savus datus pēc veida (Vārds, Uzvārds, Parole), pēc tam ar šiem datiem viņš varēs ievadīt testu. Pēc visu soļu veikšanas klients varēs veikt testu, un rezultāti tiks parādīti datu bāzē punktu veidā. Būs pieejamas arī tādas funkcijas kā administratora tiesības, viņš varēs gan pievienot datubāzes sarakstu, gan rediģēt pašu testu. testa beigās skolēns (klients) varēs apskatīt savu rezultātu un punktu skaitu, un turpmāk tas tiks saglabāts datubāzē, lai skolotājs varētu apskatīt viņa rezultātus.

# 

# 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.](#_bookmark4)).

Datu bāze

Administrātors/Skolotājs

Rezultāti

Lietotājs

Tests

* 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.](#_bookmark6)).

Lietotājs

Lietotāja

saskane

Sistēma

Datu bāze

Lietotāja dati un

rezultāti

* 1. **att. Sistēmas apkārtnes modeļa piemērs**

# Viedokļu analīze

Savāktie un identificētie viedokļi par sistēmu ir attēloti ar burbuļdiagrammas palīdzību ([1.3. att.](#_bookmark8)), tas faktiski ir viedokļu kopums, kur katra viedokļa nosaukums ir ierakstīts atsevišķa elipsē.

Datu viedokļi

Funkcionālie viedokļi

Nefunkcionālie viedokļi

Viedokļi par lietotāju

* 1. **att. Viedokļu burbuļu diagramma**

Otrais etaps viedokļu analīzē ir viedokļu klasifikācija, kad viedokļi ir sagrupēti grupās ([1.4. att.](#_bookmark9)).

* 1. **att. Viedokļu hierarhiskā struktūra**

# Datu modelēšana

Datu modelēšanai ir tiek izmantots ER modelis. Testēšanas sistēmas ER modelis ([1.5. att.](#_bookmark11)) ietver datu entītiju kopumu un relāciju kopumu starp datu entītijām, pie kam ar šo modeli var attēlot dažāda tipa relācijas starp datu elementiem.

Audzēknis

Satur

Tests

Pilda

Skolotājs

Izveido

Jautājums

* 1. **att. Vienkārsots testēšanas sistēmas ER modelis**

# Datu vārdnīca

|  |  |  |
| --- | --- | --- |
| **Datu tips** | **Nosaukums** | **Apraksts** |
| **String** | **USERNAME** | Audzēknis ievada savu lietotāju vārdu. |
| **String** | **PASSWORD** | Audzēknis ievada savu paroli. |
| **String[]** | **OPT1-15** | Atbilžu variantu masīvs. Tiek piedāvāti vairāki atbilžu  varianti, kur ir jāizvēlas viens pareizs. |
| **String** | **ANS1-5** | Pareizas atbildes. |
| **String** | **BIRTH** | Audzēkņa dzimšanas dienas datums. |
| **int** | **MARK** | Audzēkņa atzīme (0-5) |
| **double** | **PUNKTI** | Audzēkņa punkti (0-5) |
| **double** | **PROCENTI** | Audzēkņa procenti (0-100%) |
| **String** | **TITLE** | Testa nosaukums |
| **String** | **QUESTION1-5** | Testa jautajumi |
| **String** | **Gender** | Audzēkņis izvelas savu dzimumu |
| **String** | **GRUPA** | Audzēkņis izvelas savu grupu(Ssolotajs/Students) |

# Sistēmas evolūcija

Testēšanas sistēma ir neatkarīga un pašpietiekama programma. Produktam nav paredzētas vairākas sastāvdaļas.

Programmai ir trīs ārējās saskarnes, kuras var apskatīt 1.4. nodaļā – viena paredzēta reģistrēšanai sistēmā, otrā – ielogošanai sistēmā un trešā – testa pildīšanai. Grafisko lietotāja saskarni var pilnveidot, papildinot to ar dažādām krasu un noformējuma shēmām.

Testēšanas sistēmā paredzēti divu lietotāju veidi: administrators un lietotājs, kurš pilda testu. Nākotnē plānots pievienot vēl vienu lietotāju grupu ar funkciju veidot jaunus testus un pārbaudīt izpildītos testus, līdz ar to lietotājiem būs divas lomas: audzēknis, kurš pilda testu un skolotājs, kurš veido un pārbauda testus. Administratoram plānots realizēt funkciju mainīt lietotāja vārdu, uzvārdu, lietotājvārdu vai paroli, ka arī pievienot un dzēst lietotājus un mainīt vīnu lomas.

Programma satur vienu testu, kurā ir 10 jautājumi, tomēr nākotnē sistēmu var pilnveidot, pievienojot jaunas testa tēmas, kļūdu uzrādīšanu un kļūdu skaidrojumu pēc testa izpildes, lietotāju reitingu, lietotāju datu izvadi lietotāja izvēlnē vai administratora izvēlnē.

Sistēmas dažiem datu tipiem tiks izmantoti dati, kuri tiek saglabāti teksta failā, nākotnē programmu var pilnveidot darbam ar datu bāzi (datu bāzes savienojums, testa jautājumi un atbildes uz tiem u.c.)

# Sistēmas funkcionālās prasības

## 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!”.

## 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!”.

## 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!"

## 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!"

## Funkcija “Atbildēt uz testa jautājumu”

### ID: T5

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

**Ievade:** Lietotājs izvēlas 1 no 4 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.

## 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. Neatbildēto jautājuma skaits.
  2. Pareizo atbilžu skaits.
  3. Nepareizo atbilžu skaits.
  4. Pareizo atbilžu procentuālā daļa.
  5. Atzīme.
  6. Sistēmas komentārs par saņemto atzīmi.

## Funkcija “Atkārtoti pildīt testu”

***ID: T7***

**Ievads:** Ļauj atkārtoti pildīt testu.

**Ievade:** Lietotājs noklikšķina uz pogas " Mēģinat vēlreiz ".

**Apstrāde:** Noklikšķinot uz pogas, lietotājs tiks pārcelts uz to pašu testu,bet visiem lietotajiem ir tikai viens meiginajums.

**Izvade:** Tiek atvērts jauna testa dialoglodziņš.

## Funkcija “Slēgt programmu”

### ID: T8

**Ievads:** Ļauj aizvērt programmu

**Ievade:** Lietotājs noklikšķina mein loga uz pogu " Iziet " vai “Exit”.

**Apstrāde:** Noklikšķinot uz pogas, lietotājs aizver visu programu vai tieši to logu kura viņam ir atverta.

**Izvade:** Tiek aizslēgts logs.

## 

## Funkcija “Paroles maiņa”

### ID: T9

**Ievads:** Ļauj pamaiņit paroļi

**Ievade:** Lietotājs noklikšķina uz jLeble “aizmirsāt savu paroli?".

**Apstrāde:** Noklikšķinot uz jLeble, lietotājs tiek pārcelts uz citu jDialog(PassM), kurā var mainīt paroli. Tas ir nepieciešams situācijās, kad students vai pasniedzējs ir aizmirsis savu iepriekšējo paroli..

**Izvade:** Tiek atvērts dialoglodziņš ar paroles maiņu.

## Funkcija “Rādīt visus lietotājus”

### ID: T10

**Ievads:** Ļauj redzēt, kuri lietotāji jau ir reģistrēti(Tas ir piejams tikaj Skolotajam!!).

**Ievade:** Lietotājs noklikšķina uz jMenu5 “View Users".

**Apstrāde:** Noklikšķinot uz jMenu5, lietotājs tiek pārcelts uz jDialog(ViewUser), kurā viņš var redzet tabulu ar visiem lietotājiem.

**Izvade:** Tiek atvērts dialoglodziņš ar visiem lietotajiem.

## Funkcija “Meklet lietotājus”

### ID: T11

**Ievads:** Ļauj meklet jeb kuru lietotāju kuri ir reģistrēti(Tas ir piejams tikaj Skolotajam!!).

**Ievade:** Skolotajs noklikšķina uz jMenu5 “View Users" pēc tam jTextFild raksta Skolena username.

**Apstrāde:** Noklikšķinot uz jMenu5, lietotājs tiek pārcelts uz jDialog(ViewUser), kurā viņš var redzet tabulu ar visiem lietotājiem un meklet tos pēc varda.

**Izvade:** Tiek atvērts dialoglodziņš ar visiem lietotajiem.

## Funkcija “Dzēst lietotājus”

### ID: T12

**Ievads:** Ļauj dzēst jeb kuru lietotāju kuri ir reģistrēti bet dzēst skolotaju ir slēgts(Tas ir piejams tikaj Skolotajam!!).

**Ievade:** Skolotajs noklikšķina uz jMenu7 “Delete Users" pēc tam jTextFild raksta Skolena username un var izvelet tabula skolenu kuru grib dzēst.

**Apstrāde:** Noklikšķinot uz jMenu5, lietotājs tiek pārcelts uz jDialog(DeleteUser), kurā viņš var redzet tabulu ar visiem lietotājiem un meklet tos pēc varda un pēc izveles dzēst to.

**Izvade:** Tiek atvērts dialoglodziņš ar visiem lietotajiem un pec izveles skolena kuru grib dzēst atveras JOptionPane.showMessageDialog ar Jautajumu “Vai tiešām vēlaties dzēst lietotāju?” ar atbildes variantiem Jā vai nē.

## Funkcija “Paradit paroli”

### ID: T13

**Ievads:** Ļauj lietotajam paskatit ka viņš ievada paroli.

**Ievade:** Lietotajs noklikšķina uz jCheckBox “ParaditPass" pēc tam radas ievadita parole.

**Apstrāde:** Noklikšķinot uz jCheckBox “ParaditPass" mainas jPasswordFild no “.”uz paroli kuru ievada lietotajs

**Izvade:** Tiek mainits jPasswordFild.

## Funkcija “Pievienot testu”

### ID: T14

**Ievads:** Ļauj izveidot jaunu testu(Tas ir piejams tikaj Skolotajam!!).

**Ievade:** Skolotajs noklikšķina uz jMenu1 “Add New Test" pēc tama atveras jDailog”TestTools”.

**Apstrāde:** Noklikšķinot uz jMenu1, Skolotajs tiek pārcelts uz jDialog(TestTools), kurā viņš var uzraksti jauna testa (Title, piecus Question un katram Question pa tris Option un piecas atbildes uz katru Question) .

**Izvade:** Tiek atvērts dialoglodziņš ”TestTools”.

## Funkcija “Help”

### ID: T15

**Ievads:** Ļauj paskatit loga instrukciju.

C:\Users\user\AppData\Local\Packages\Microsoft.Windows.Photos_8wekyb3d8bbwe\TempState\ShareServiceTempFolder\hiclipart.com.jpegC:\Users\user\AppData\Local\Packages\Microsoft.Windows.Photos_8wekyb3d8bbwe\TempState\ShareServiceTempFolder\hiclipart.com.jpeg**Ievade:** Audzēkņis noklikšķina uz atveras jDailog”Help”.

**Apstrāde:** Noklikšķinot uz , Audzēkņis tiek pārcelts uz jDialog(Help), kurā viņš var paskatit visu informāciju par logu kur viņš atrodas

**Izvade:** Tiek atvērts dialoglodziņš ” Help”.

# Sistēmas nefunkcionālās prasības

1. Prasības produktam:
   1. Lietotāju saskarne ar sistēmu notiek latviešu valodā.
   2. Produktam nav paredzētas vairākas sastāvdaļas. Produktam ir trīs ārējās saskarnes: reģistrēšanai sistēmā, ielogošanai sistēmā, testa pildīšanai.
   3. Programma ir paredzēta vienam lietotājam, kurš saprot latviešu valodu. Lietotāja vecuma vai izglītības ierobežojumi nav paredzēti.
2. Ārējās saskarnes prasības
   1. Lietotāja saskarne: produkts paredz vienotu lietotāja saskarni.
   2. Ekrāna formāti: minimālais ekrāna izmērs ir 10” ar minimālo izšķirtspēju –800x600 punkti.
   3. Lietotāja saskarne “Lietotāja reģistrēšanas saskarne”. Saskarnes struktūra: skat. [1.6. att.](#_bookmark16)

Attels

**Registration**

Date of birth(YYYY/MM/DD):

Group:

<izvēlēties dzimumu>

Username:

Gender:

Password:

Ielogoties

Exit

**1.6. att. Lietotāja reģistrēšanas saskarne**

Saskarnes elementi:

1. teksta lauks “Username”
2. teksta lauks “Date of birth(YYYY/MM/DD)”
3. JComboBox “Gender(Izvēlēties dzimumu)”
4. teksta lauks ‘Parole”
5. JComboBox “Group(izvēlēties grupu(Skolotajs/Student))”
6. poga “Ielogoties”
7. poga “Exit”
   1. Lietotāja saskarne “Lietotāja ielogošanas saskarne”. Saskarnes struktūra: skat. [1.7. att.](#_bookmark17)

Attels

**Login**

Username:

Password:

Ienakt

Exit

aizmirsāt savu paroli?

Parādīt paroli.

izveidot kontu.

**1.7. att. Lietotāja ielogošanas saskarne**

Saskarnes elementi:

1. teksta lauks “Lietotājvārds”
2. teksta lauks ‘Parole”
3. JLable “aimirsāt savu parole?”
4. JLable “izveidot kontu.”
5. JCheckBox “Parādīt parole”
6. poga “Ienakt”
7. poga “Exit”
   1. Lietotāja saskarne “Testa izpilde”. Saskarnes struktūra: skat. [1.8. att.](#_bookmark18)

**Testa Izvele**

Sakt Testu

Exit

Test1

Test2

Test3

Mēģiniet vēlreiz

Saglabat un iziet

**Rezultati**

User name:

Testa nosa:

Procenti: X/100.0%

Mark: X/5

Punkti: X/10p

3

1

2

Izveles logs

Rezultati

1

2

TITLE

2)Jautajums?

1)Jautajums?

3)Jautajums?

4)Jautajums?

5)Jautajums?

Save

Pabeigt

Testa logs

1

1

1

1

2

3

1

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

Saskarnes elementi:

Testa Izvele

1. Poga “Sākt testu”.
2. Poga “Beigt”
3. jList1(Testu atrašanas vieta)

Rezultati

1. Poga “Savglabat un iziet”
2. Poga “Mēģiniet vēlreiz”

Testu logs

1. jRadioButton:Atbilžu varianti
2. Poga “Save”
3. Poga “Pabeigt”
   1. Lietotāja saskarne “Testa izpilde”. Saskarnes struktūra: skat. [1.9. att.](#_bookmark18)

TITLE

2)Jautajums?

1)Jautajums?

3)Jautajums?

4)Jautajums?

5)Jautajums?

Save

Pabeigt

Testa logs

1

1

1

1

2

3

1

Mēģiniet vēlreiz

Saglabat un iziet

**Rezultati**

User name:

Testa nosa:

Procenti: X/100.0%

Mark: X/5

Punkti: X/10p

Rezultati

1

2

1.9 . att. Lietotāja saskarne " Mēģinat vēlreiz"

Saskarnes elementi:

Testa Logs:

1. jRadioButton:Atbilžu varianti
2. Poga “Save”
3. Poga “Pabeigt”

Rezultati:

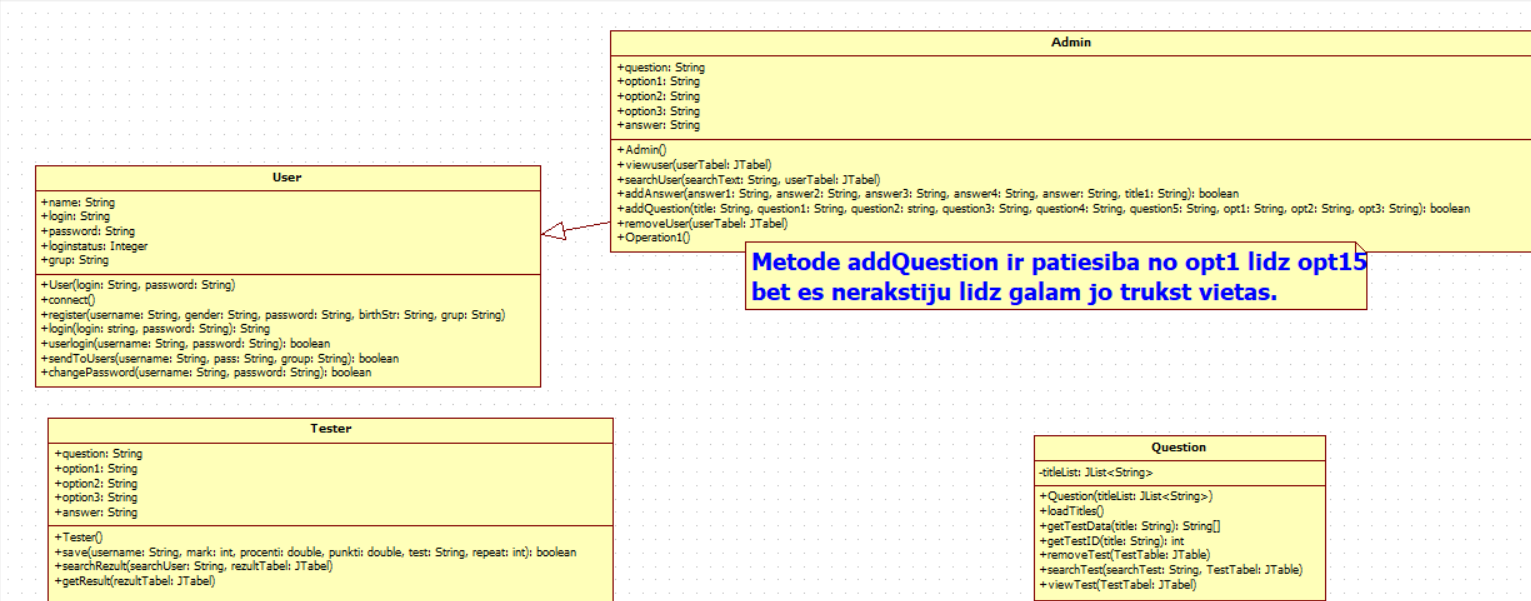
1. Poga “Savglabat un iziet”
2. Poga “Mēģiniet vēlreiz”

# 

# 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.](#_bookmark21)):

1. Klase **User** – ir atbildīga par darbu ar lietotājiem.
   * User (login: String, password: String) – konstruktors, kurš izveido jaunu lietotāju ar norādītiem datiem;
   * Connect()-metode , kura dot iespeju ienakt;
   * Register()-metode,kura dot iespeju lietotājam ielogoties;
   * Login()-metode,kura dot lietotājam ienakt portala;
   * sendToUser()-metode pievieno lietotaju datu baze.
   * changePassword()-metode kura dot lietotajam iespeju mainit paroli
2. Klase **Admin** – nodrošina darbu ar lietotajiem un testa viedošanu un redigēšanu.

* Viewuser()-metode,dot adminam iespeju paskatit visus lietotajus;
* searchUser()-metode,dot adminam iespeju meklet lietotajus tabula;
* addAnswer()-metode, kura dot veidot un parnest uzrakstitas atbildes uz datu bazi;
* addQuestion()-metode, kura dot veidot un parnest uzrakstitus jautajumus uz datu bazi;
* removeUser()-dot adminam izdzest jeb kuru lietotaju bet viņš nevar izdzet skolotajus;

1. Klase **Question** – ir atbildīga par darbu ar testa jautājumiem, to izvadi un ievadītas atbildes pārbaudi , meklet testus un t.t.

* loadTitles()-metode, pievieno testa nosaukumus JList;
* getTestData()-metode, kura saņiem visus datus par testu;
* removeTest()-metode, kura ļauj dzest testu uspiežot uz nosaukumu;
* searchTest()-metode, kura ļauj meklet testu pēc nosaukuma;
* viewTest()-metode, kura ļauj paskatit kadi testi ir izveidoti

1. Klase **Tester** – ir atbildīga par darbu ar testa pareizam atbildem un tas saglabšanu.

* Save()-metode , ļauj saglabat visus pareizus atbildes
* searchRezult()-metode, ļauj meklset lietotaju rezultatus;
* getResult()-metode, parnes visus rezultatus datu baze;

# Klašu realizēšana Java valodā

Izstrādātās klašu diagrammas ([2.1. att.](#_bookmark21)) 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.](#_bookmark33) [Klašu sistēma](#_bookmark33)”, bet programmas kods, kas realizē šīs klašu sistēmas testēšanu (klase ar grafisko lietotāja saskarni) ir ievietots pielikumā “[Pielikums B. testa programmas kods](#_bookmark34)”.

public class User {

String login;

String name;

String password;

static final String DB\_URL = "jdbc:derby://localhost:1527/connect";

static final String DB\_USER = "admin44";

static final String DB\_PASSWD = "Qwerty1234";

public boolean User(String login, String password) {

return false;

}

public void connect() throws SQLException {

}

public int loginstatus = 0;

public String grup;

public String login(String login, String password) {

return grup;

}

public boolean register(String username, String gender, String password, String birthStr, String grup) throws SQLException {

}

public boolean sendToUsers(String username, String pass, String group) {

return false;

}

}

class Admin

String question;

String option1;

String option2;

String option3;

String option4;

String answer;

static final String DB\_URL = "jdbc:derby://localhost:1527/connect";

static final String DB\_USER = "admin44";

static final String DB\_PASSWD = "Qwerty1234";

Admin() {

}

public void viewuser(JTable userTable) {

}

public void searchUser(String searchText, JTable userTable) {

}

public boolean addAnswer(String answer1,String answer2,String answer3,String answer4,String answer5,String title1) {

return false;

}

public boolean addQuestion(String title, String quest1, String quest2, String quest3, String quest4, String quest5,

return false;

}

public void removeUser(JTable userTable) {

}

}

public class Question {

static final String DB\_URL = "jdbc:derby://localhost:1527/connect";

static final String DB\_USER = "admin44";

static final String DB\_PASSWD = "Qwerty1234";

private JList<String> titleList;

public Question(JList<String> titleList) {

}

public void loadTitles() {

}

titleList.setModel(listModel);

}

public String[][] getTestData(String title) {

return data;

}

public int getTestID(String title) {

return 0;

}

public void removeTest(JTable TestTable){

}

public void searchTest(String searchTest, JTable TestTable){

}

public void viewTest(JTable TestTable) {

}

}

public class Tester {

String password;

String question;

String option1;

String option2;

String option3;

String option4;

String answer;

static final String DB\_URL = "jdbc:derby://localhost:1527/connect";

static final String DB\_USER = "admin44";

static final String DB\_PASSWD = "Qwerty1234";

public Tester(){

}

public boolean save(String username,int mark, double procenti,double punkti,String test,int repeat ){

return false;

}

public void load(){

}

public void searchRezult(String searchUser, JTable rezultTable) {

}

public void getResults(JTable rezultTable){

}

}

# 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:
   * Ielogošanas process paredzēts reģistrēto lietotāju pievienošanai sistēmā. Lai lietotājs varētu piekļūt testam viņam pēc ir nepieciešams autorizēties. Ielogošanas procesā lietotājs ievada iepriekš reģistrētos datus – lietotājvārdu un paroli ([3.1. att.](#_bookmark26) ). Ja lietotāja dati sakrīt ar validācijas datiem, tad lietotājam tiek atvērts programmas galvenais logs (3.2. att.).
   * Ielogošanas procesā ir iespējamas vairākas kļūdas – neaizpildīti lauki, neprecīzi ievadīti lietotāja dati.

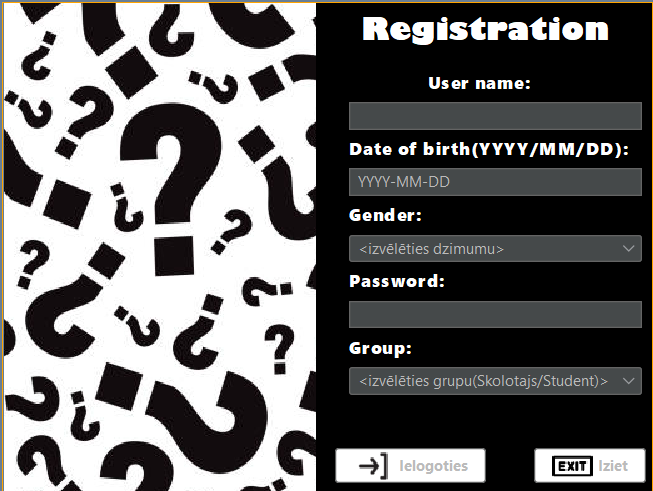


**3.1. att. Testēšanas sistēmas ielogošanas forma**

****

3.2. att. Programmas galvenais logs

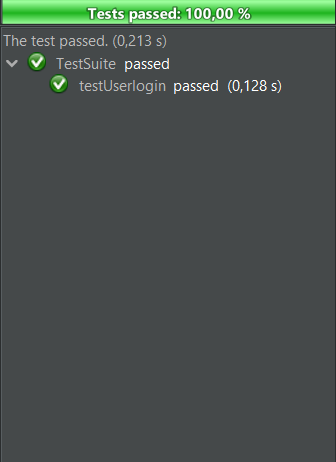
1. Reģistrācijas procesa apraksts:
   * Reģistrācijas process reģistrēto lietotāju pievienošanai sistēmā. Lai lietotājs varētu izveidot tkontu viņam ir nepieciešams uzrakstit nepieciešamus datus. Reģistrācijas procesā lietotājs ievada reģistracijas datus – lietotājvārdu,dzimšanas datums,gender, paroli un grupu(skolotajs/skolens) (3.3. att. ). Ja lietotāja dati sakrīt ar validācijas datiem, tad lietotājam tiek atvērts programmas ielogošanas logs (3.4. att.).
   * Reģistrācijas procesā ir iespējamas vairākas kļūdas – neaizpildīti lauki.



3.3. att. Testēšanas sistēmas reģistrācijas forma

3.4. att. Testēšanas sistēmas ielogošanas forma

# Testēšanas rezultāti



Pārbauda savienojumu ar datubāzi un autorizāciju.

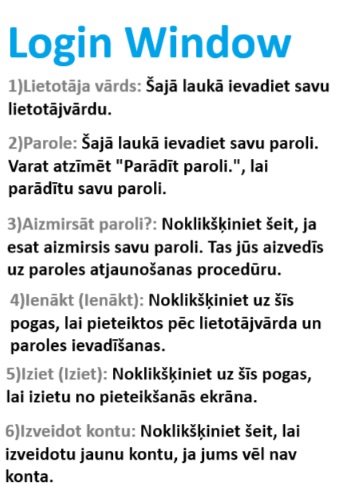
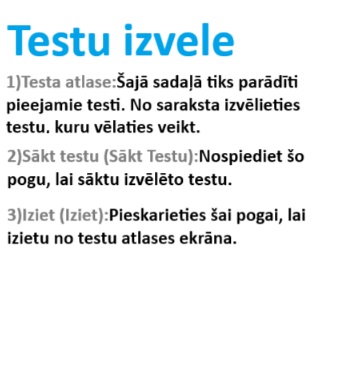
izmantojot automatizēto JUnit4 testu sistēmu.

# 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. Lietotāja izvēlnē ir 2 pogas.
   1. Poga « Sākt testu» ļauj sākt pildīt testu.
      1. Pēc testa izpildīšanas tiek paradīts testa izpildes rezultāti.
   2. Poga «Iziet» ļauj iziet no sava konta.
4. Administratora izvēlnē ir 3 pogas.
   1. Poga «Sākt testu» ļauj atļaut lietotājam piekļuvi testam.
   2. Poga «Add New Test» ļauj izveidot jaunu testu skolēņiem.
   3. Poga «Delete Test» ļauj dzest jeb kuru testu kas ir izveidots.
   4. Poga «All Student Rezultat» ļauj skatit kadi rezultati ir studentam.
   5. Poga «View User» ļauj skatit visus studentu registracijas datus.
   6. Poga «Delete User» ļauj dzest jeb kuru studentu , ir svarigs ka skolotajs nevar dzest skolotajus.
   7. Poga «Iziet» ļauj iziet no administratora konta.

# Palīdzības sistēma (Help)



# 

# Secinājumi

Tika izstrādāts projekts «Testēšanas sistēma» NetBeans vidē. Tika izpildītas visas programmas bāzes prasības. Projekta izpildīšanas laikā radās dažādas problēmas, kas neļāva veikt sākotnējo vēlamu rezultātu, bet tajā pašā laikā bija interesanti uzzināt jaunu informāciju, kas noderēs turpmāk. Piemēram, darbs ar JDBC (the Java Database Connectivity) un jaunas darba izpildes metodes. Tā bija mana pirmā pieredze, izstrādājot projektu. Diemžēl ir palikušas nerealizētas idejas, kas varētu izveidot manu projektu labāk, bet galvenais, ka ir realizētas galvenās prasības un viss strādā. Ceru, jo tālāk, jo labāk!

# Izmantotās literatūras (informācijas avotu) saraksts

1. **Zaiceva, L.** *Programmatūras izstrādes tehnoloģija.* Rīga : RTU, 2002.
2. **Taylor Johnson, Dung X. Nguyen.** UML and more JAVA Syntax. [Tiešsaiste] COMP 212 LAB 1.5, 2007. gada 16. 1. [Citēts: 2020. gada 16. 10.] https:/[/www.clear.ric](http://www.clear.rice.edu/comp212/07-)e[.edu/comp212/07-](http://www.clear.rice.edu/comp212/07-) spring/labs/01.5/.
3. **Minkyu Lee, Hyunsoo Kim, Jeongil Kim, Jangwoo Lee, Deukkyu Gum.** StarUml 5.0 User Guide. [Tiešsaiste] 2005. gada. [Citēts: 2020. gada 12. 10.] <http://staruml.sourceforge.net/docs/user-> guide(en)/toc.html.
4. **Dortiņa, I. un Dauģerts, A.** 1.1.1. Ievads Java. Java pamati. [Tiešsaiste] 2022. gada 15. 2. [Citēts: 2023. gada 03. 30.] [http://e.daugvt.lv/mod/resource/view.php?id=3321.](http://e.daugvt.lv/mod/resource/view.php?id=3321)
5. —. 2.1.1.1. Java. OOP koncepcija. [Tiešsaiste] 2022. gada 01. 06. [Citēts: 2023. gada 30. 03.] [http://e.daugvt.lv/mod/resource/view.php?id=3432.](http://e.daugvt.lv/mod/resource/view.php?id=3432)
6. —. 2.1.2. Java. Mantošana un metožu pārdefinēšana. [Tiešsaiste] 2022. gada 07. 20. [Citēts: 2023. gada 30. 03.] [http://e.daugvt.lv/mod/resource/view.php?id=3611.](http://e.daugvt.lv/mod/resource/view.php?id=3611)
7. —. 2.1.3. Java pakotnes un interfeisi. [Tiešsaiste] 2022. gada 04. 11. [Citēts: 2023. gada 30. 03.] [http://e.daugvt.lv/mod/resource/view.php?id=3676.](http://e.daugvt.lv/mod/resource/view.php?id=3676)
8. —. 2.2. Java izņēmumi un to apstrāde. [Tiešsaiste] 2022. gada 24. 11. [Citēts: 2023. gada 30. 03.] [http://e.daugvt.lv/mod/resource/view.php?id=3742.](http://e.daugvt.lv/mod/resource/view.php?id=3742)
9. —. 4.1.1. Java. Grafiskais lietotāja interfeiss. [Tiešsaiste] 2023. gada 23. 02. [Citēts: 2023. gada
10. 03.] [http://e.daugvt.lv/mod/resource/view.php?id=3852.](http://e.daugvt.lv/mod/resource/view.php?id=3852)
    1. —. 4.1.2. Java GUI izveidošana NetBeans vidē. [Tiešsaiste] 2023. gada 03. 03. [Citēts: 2023. gada 30. 03.] [http://e.daugvt.lv/mod/resource/view.php?id=3870.](http://e.daugvt.lv/mod/resource/view.php?id=3870)
    2. —. 2.1.1.2. Java. Metodes un konstruktori. [Tiešsaiste] 2022. gada 16. 09. [Citēts: 2023. gada
11. 03.] [http://e.daugvt.lv/mod/resource/view.php?id=3567.](http://e.daugvt.lv/mod/resource/view.php?id=3567)
    1. —. 1.1.2. Java sazarotas struktūras operātori. [Tiešsaiste] 2022. gada 21. 02. [Citēts: 2023. gada 30. 03.] [http://e.daugvt.lv/mod/resource/view.php?id=3330.](http://e.daugvt.lv/mod/resource/view.php?id=3330)
    2. —. 1.1.3. Java cikliskās struktūras operatori. [Tiešsaiste] 2022. gada 01. 03. [Citēts: 2023. gada 30. 03.] [http://e.daugvt.lv/mod/resource/view.php?id=3340.](http://e.daugvt.lv/mod/resource/view.php?id=3340)

14. —. 1.2.1.1. Java masīvi. [Tiešsaiste] 2022. gada 25. 03. [Citēts: 2023. gada 30. 03.] [http://e.daugvt.lv/mod/resource/view.php?id=3347.](http://e.daugvt.lv/mod/resource/view.php?id=3347)

1. —. 1.2.3. Java Ievades/izvades sistēma. [Tiešsaiste] 2022. gada 11. 05. [Citēts: 2023. gada 30. 03.] [http://e.daugvt.lv/mod/resource/view.php?id=3419.](http://e.daugvt.lv/mod/resource/view.php?id=3419)
2. —. 1.2.1.2. Java rakstzīmju masīvi un rakstzīmju virknes. [Tiešsaiste] 2022. gada 06. 04. [Citēts: 2023. gada 30. 03.] [http://e.daugvt.lv/mod/resource/view.php?id=3383.](http://e.daugvt.lv/mod/resource/view.php?id=3383)
3. —. 1.2.1.3. Java dinamiskie masīvi. [Tiešsaiste] 2022. gada 13. 04. [Citēts: 2023. gada 30. 03.] [http://e.daugvt.lv/mod/resource/view.php?id=3398.](http://e.daugvt.lv/mod/resource/view.php?id=3398)
4. —. 1.2.2. Java ieraksti. [Tiešsaiste] 2022. gada 27. 04. [Citēts: 2023. gada 30. 03.] [http://e.daugvt.lv/mod/resource/view.php?id=3408.](http://e.daugvt.lv/mod/resource/view.php?id=3408)
5. **Kodors, Sergejs.** *IEVADS PRASĪBU INŽENIERIJĀ. Mācību līdzeklis.* Rēzekne : Rēzeknes Tehnoloģiju akadēmija, 2019. 978-9984-44-226-6.
6. Latvijas Nacionālais terminoloģijas portāls. [Tiešsaiste] termini.gov.lv. Izstrādātājs: Tilde., 2005-2023. gada. [Citēts: 2023. gada 30. 03.] https://termini.gov.lv/.

# Pielikums A. Klašu sistēma

**ADMIN CLASS:**

import java.sql.\*;

import java.util.Vector;

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

public class Admin extends User {

String question;

String option1;

String option2;

String option3;

String answer;

static final String DB\_URL = "jdbc:derby://localhost:1527/connect";

static final String DB\_USER = "admin44";

static final String DB\_PASSWD = "Qwerty1234";

public Admin() {

}

public void viewuser(JTable userTable) {

String query = "SELECT \* FROM APP.REGISTRATION";

try {

Connection connection = DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASSWD);

Statement statement = connection.createStatement();

ResultSet resultSet = statement.executeQuery(query);

ResultSetMetaData metaData = resultSet.getMetaData();

int columnCount = metaData.getColumnCount();

Vector<String> columnNames = new Vector<>();

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

columnNames.add(metaData.getColumnName(i));

}

Vector<Vector<Object>> data = new Vector<>();

while (resultSet.next()) {

Vector<Object> row = new Vector<>();

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

row.add(resultSet.getObject(i));

}

data.add(row);

}

DefaultTableModel model = new DefaultTableModel(data, columnNames);

userTable.setModel(model);

connection.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

public void searchUser(String searchText, JTable userTable) {

String query = "SELECT \* FROM APP.REGISTRATION WHERE USERNAME LIKE ?";

try {

Connection connection = DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASSWD);

PreparedStatement preparedStatement = connection.prepareStatement(query);

preparedStatement.setString(1, "%" + searchText + "%");

ResultSet resultSet = preparedStatement.executeQuery();

ResultSetMetaData metaData = resultSet.getMetaData();

int columnCount = metaData.getColumnCount();

Vector<String> columnNames = new Vector<>();

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

columnNames.add(metaData.getColumnName(i));

}

Vector<Vector<Object>> data = new Vector<>();

while (resultSet.next()) {

Vector<Object> row = new Vector<>();

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

row.add(resultSet.getObject(i));

}

data.add(row);

}

DefaultTableModel model = new DefaultTableModel(data, columnNames);

userTable.setModel(model);

connection.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

public boolean addAnswer(String answer1,String answer2,String answer3,String answer4,String answer5,String title1) {

String getMaxIdQuery = "SELECT MAX(ID) FROM APP.ANSWER";

String insertQuery = "INSERT INTO APP.ANSWER (ID,AN1,AN2,AN3,AN4,AN5,TITLE ) VALUES (?, ?, ?, ?, ?, ?, ?)";

try (Connection connection = DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASSWD);

PreparedStatement getMaxIdStmt = connection.prepareStatement(getMaxIdQuery);

PreparedStatement insertStmt = connection.prepareStatement(insertQuery)) {

ResultSet rs = getMaxIdStmt.executeQuery();

int maxId = 0;

if (rs.next()) {

maxId = rs.getInt(1);

}

int newId = maxId + 1;

insertStmt.setInt(1, newId);

insertStmt.setString(2, answer1);

insertStmt.setString(3, answer2);

insertStmt.setString(4, answer3);

insertStmt.setString(5, answer4);

insertStmt.setString(6, answer5);

insertStmt.setString(7, title1);

int rowsAff = insertStmt.executeUpdate();

return rowsAff > 0;

} catch (SQLException e) {

e.printStackTrace();

return false;

}

}

public boolean addQuestion(String title, String quest1, String quest2, String quest3, String quest4, String quest5, String opt1, String opt2, String opt3, String opt4, String opt5, String opt6, String opt7, String opt8, String opt9, String opt10, String opt11, String opt12, String opt13, String opt14, String opt15) {

String getMaxIdQuery = "SELECT MAX(ID) FROM APP.QUESTION";

String insertQuery = "INSERT INTO APP.QUESTION (ID, TITLE, QUEST1, QUEST2, QUEST3, QUEST4, QUEST5, OPT1, OPT2, OPT3, OPT4, OPT5, OPT6, OPT7, OPT8, OPT9, OPT10, OPT11, OPT12, OPT13, OPT14, OPT15) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";

try (Connection connection = DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASSWD);

PreparedStatement getMaxIdStmt = connection.prepareStatement(getMaxIdQuery);

PreparedStatement insertStmt = connection.prepareStatement(insertQuery)) {

ResultSet rs = getMaxIdStmt.executeQuery();

int maxId = 0;

if (rs.next()) {

maxId = rs.getInt(1);

}

int newId = maxId + 1;

insertStmt.setInt(1, newId);

insertStmt.setString(2, title);

insertStmt.setString(3, quest1);

insertStmt.setString(4, quest2);

insertStmt.setString(5, quest3);

insertStmt.setString(6, quest4);

insertStmt.setString(7, quest5);

insertStmt.setString(8, opt1);

insertStmt.setString(9, opt2);

insertStmt.setString(10, opt3);

insertStmt.setString(11, opt4);

insertStmt.setString(12, opt5);

insertStmt.setString(13, opt6);

insertStmt.setString(14, opt7);

insertStmt.setString(15, opt8);

insertStmt.setString(16, opt9);

insertStmt.setString(17, opt10);

insertStmt.setString(18, opt11);

insertStmt.setString(19, opt12);

insertStmt.setString(20, opt13);

insertStmt.setString(21, opt14);

insertStmt.setString(22, opt15);

int rowsAff = insertStmt.executeUpdate();

return rowsAff > 0;

} catch (SQLException e) {

e.printStackTrace();

return false;

}

}

public void removeUser(JTable userTable) {

int selectedRow = userTable.getSelectedRow();

if (selectedRow >= 0) {

String username = (String) userTable.getValueAt(selectedRow, 0);

int columnCount = userTable.getColumnCount();

String group = "";

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

if ("GRUP".equals(userTable.getColumnName(i))) {

group = (String) userTable.getValueAt(selectedRow, i);

break;

}

}

if ("Skolotajs".equals(group)) {

JOptionPane.showMessageDialog(null, "Jūs nevarat dzēst šo personu, jo viņš ir Skolotajs.", "Kļūda", JOptionPane.ERROR\_MESSAGE);

return;

}

String query1 = "DELETE FROM APP.REGISTRATION WHERE USERNAME = ?";

String query2 = "DELETE FROM APP.USERS WHERE USERNAME = ?";

try (Connection connection = DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASSWD); PreparedStatement stmt1 = connection.prepareStatement(query1); PreparedStatement stmt2 = connection.prepareStatement(query2)) {

stmt1.setString(1, username);

int rowsAff1 = stmt1.executeUpdate();

stmt2.setString(1, username);

int rowsAff2 = stmt2.executeUpdate();

if (rowsAff1 > 0 && rowsAff2 > 0) {

System.out.println("User " + username + " deleted successfully.");

((DefaultTableModel) userTable.getModel()).removeRow(selectedRow);

} else {

System.out.println("User " + username + " not found.");

}

} catch (SQLException e) {

e.printStackTrace();

}

} else {

System.out.println("No user selected.");

}

}

}

**USER NAME:**

import java.sql.\*;

import javax.swing.JOptionPane;

public class User {

String login;

String name;

String password;

static final String DB\_URL = "jdbc:derby://localhost:1527/connect";

static final String DB\_USER = "admin44";

static final String DB\_PASSWD = "Qwerty1234";

public boolean User(String login, String password) {

String query = "SELECT \* FROM APP.USERS WHERE USERNAME = ?";

try ( Connection connection = DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASSWD); PreparedStatement preparedStatement = connection.prepareStatement(query)) {

preparedStatement.setString(1, login);

try ( ResultSet resultSet = preparedStatement.executeQuery()) {

if (resultSet.next()) {

String dbPassword = resultSet.getString("PASSWORD");

return password.equals(dbPassword);

} else {

return false;

}

} catch (SQLException e) {

e.printStackTrace();

return false;

}

} catch (SQLException e) {

e.printStackTrace();

return false;

}

}

public void connect() throws SQLException {

try ( Connection connection = DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASSWD); Statement statement = connection.createStatement();) {

try ( ResultSet resultSet

= statement.executeQuery("SELECT \* FROM APP.USERS");) {

while (resultSet.next()) {

System.out.print(resultSet.getInt("ID") + ". ");

System.out.print(resultSet.getString("USERNAME") + " ");

System.out.print(resultSet.getString("PASSWORD") + " \n");

}

} catch (SQLException e) {

e.printStackTrace();

}

} catch (SQLException e) {

e.printStackTrace();

}

}

public int loginstatus = 0;

public String grup;

public String login(String login, String password) {

String query = "SELECT \* FROM APP.USERS WHERE USERNAME = ? AND PASSWORD = ?";

try ( Connection connection = DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASSWD); PreparedStatement preparedStatement = connection.prepareStatement(query)) {

preparedStatement.setString(1, login);

preparedStatement.setString(2, password);

try ( ResultSet resultSet = preparedStatement.executeQuery()) {

if (resultSet.next()) {

grup = resultSet.getString("GRUP");

loginstatus = 1;

} else {

loginstatus = 0;

}

} catch (SQLException e) {

e.printStackTrace();

loginstatus = 0;

}

} catch (SQLException e) {

e.printStackTrace();

loginstatus = 0;

}

return grup;

}

public boolean register(String username, String gender, String password, String birthStr, String grup) throws SQLException {

String query = "INSERT INTO APP.REGISTRATION (USERNAME, GENDER, PASSWORD, BIRTH, GRUP) VALUES (?, ?, ?, ?, ?)";

try ( Connection connection = DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASSWD); PreparedStatement stmt = connection.prepareStatement(query)) {

stmt.setString(1, username);

stmt.setString(2, gender);

stmt.setString(3, password);

stmt.setString(4, birthStr);

stmt.setString(5, grup);

int rowsAff = stmt.executeUpdate();

return rowsAff > 0;

} catch (SQLException e) {

e.printStackTrace();

return false;

}

}

public boolean userlogin(String username, String password) throws SQLException {

String query = "INSERT INTO APP.REGISTRATION (USERNAME, PASSWORD) VALUES (?, ?)";

try ( Connection connection = DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASSWD); PreparedStatement preparedStatement = connection.prepareStatement(query)) {

preparedStatement.setString(1, username);

preparedStatement.setString(2, password);

int rowsAffected = preparedStatement.executeUpdate();

return rowsAffected > 0;

} catch (SQLException e) {

e.printStackTrace();

return false;

}

}

public boolean sendToUsers(String username, String pass, String group) {

String query = "INSERT INTO APP.USERS ( USERNAME, PASSWORD, GRUP) VALUES ( ?, ?, ?)";

try ( Connection connection = DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASSWD); PreparedStatement stmt = connection.prepareStatement(query)) {

stmt.setString(1, username);

stmt.setString(2, pass);

stmt.setString(3, group);

int rowsAff = stmt.executeUpdate();

return rowsAff > 0;

} catch (SQLException e) {

e.printStackTrace();

return false;

}

}

public boolean changePassword(String username, String newPassword) {

String query = "UPDATE APP.USERS SET PASSWORD = ? WHERE USERNAME = ?";

try ( Connection connection = DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASSWD); PreparedStatement stmt = connection.prepareStatement(query)) {

stmt.setString(1, newPassword);

stmt.setString(2, username);

int rowsAff = stmt.executeUpdate();

return rowsAff > 0;

} catch (SQLException e) {

e.printStackTrace();

return false;

}

}

}

**QUESTION CLASS:**

import javax.swing.\*;

import java.sql.\*;

import java.util.Vector;

import javax.swing.table.DefaultTableModel;

public class Question {

static final String DB\_URL = "jdbc:derby://localhost:1527/connect";

static final String DB\_USER = "admin44";

static final String DB\_PASSWD = "Qwerty1234";

private JList<String> titleList;

public Question(JList<String> titleList) {

this.titleList = titleList;

}

public void loadTitles() {

DefaultListModel<String> listModel = new DefaultListModel<>();

try ( Connection connection = DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASSWD); Statement stmt = connection.createStatement(); ResultSet rs = stmt.executeQuery("SELECT TITLE FROM APP.QUESTION")) {

while (rs.next()) {

String title = rs.getString("TITLE");

listModel.addElement(title);

}

} catch (SQLException e) {

e.printStackTrace();

}

titleList.setModel(listModel);

}

public String[][] getTestData(String title) {

String[][] data = new String[5][4]; // 5 questions, each with the question text and 3 options

Connection conn = null;

PreparedStatement pstmt = null;

ResultSet rs = null;

try {

conn = DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASSWD);

String sql = "SELECT QUEST1, QUEST2, QUEST3, QUEST4, QUEST5, OPT1, OPT2, OPT3, OPT4, OPT5, OPT6, OPT7, OPT8, OPT9, OPT10, OPT11, OPT12, OPT13, OPT14, OPT15 FROM APP.QUESTION WHERE TITLE = ?";

pstmt = conn.prepareStatement(sql);

pstmt.setString(1, title);

rs = pstmt.executeQuery();

if (rs.next()) {

data[0][0] = rs.getString("QUEST1");

data[0][1] = rs.getString("OPT1");

data[0][2] = rs.getString("OPT2");

data[0][3] = rs.getString("OPT3");

data[1][0] = rs.getString("QUEST2");

data[1][1] = rs.getString("OPT4");

data[1][2] = rs.getString("OPT5");

data[1][3] = rs.getString("OPT6");

data[2][0] = rs.getString("QUEST3");

data[2][1] = rs.getString("OPT7");

data[2][2] = rs.getString("OPT8");

data[2][3] = rs.getString("OPT9");

data[3][0] = rs.getString("QUEST4");

data[3][1] = rs.getString("OPT10");

data[3][2] = rs.getString("OPT11");

data[3][3] = rs.getString("OPT12");

data[4][0] = rs.getString("QUEST5");

data[4][1] = rs.getString("OPT13");

data[4][2] = rs.getString("OPT14");

data[4][3] = rs.getString("OPT15");

}

} catch (SQLException e) {

e.printStackTrace();

} finally {

try {

if (rs != null) {

rs.close();

}

if (pstmt != null) {

pstmt.close();

}

if (conn != null) {

conn.close();

}

} catch (SQLException e) {

e.printStackTrace();

}

}

return data;

}

public int getTestID(String title) {

Connection conn = null;

PreparedStatement pstmt = null;

ResultSet rs = null;

try {

conn = DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASSWD);

String sql = "SELECT QUEST1, QUEST2, QUEST3, QUEST4, QUEST5, OPT1, OPT2, OPT3, OPT4, OPT5, OPT6, OPT7, OPT8, OPT9, OPT10, OPT11, OPT12, OPT13, OPT14, OPT15 FROM APP.QUESTION WHERE TITLE = ?";

pstmt = conn.prepareStatement(sql);

pstmt.setString(1, title);

rs = pstmt.executeQuery();

if (rs.next()) {

return rs.getInt("ID");

}

} catch (SQLException e) {

e.printStackTrace();

} finally {

try {

if (rs != null) {

rs.close();

}

if (pstmt != null) {

pstmt.close();

}

if (conn != null) {

conn.close();

}

} catch (SQLException e) {

e.printStackTrace();

}

}

return 0;

}

public void removeTest(JTable TestTable) {

int selectedRow = TestTable.getSelectedRow();

if (selectedRow >= 0) {

String title = (String) TestTable.getValueAt(selectedRow, 1);

int columnCount = TestTable.getColumnCount();

String query1 = "DELETE FROM APP.QUESTION WHERE TITLE = ?";

String query2 = "DELETE FROM APP.ANSWER WHERE TITLE = ?";

try ( Connection connection = DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASSWD); PreparedStatement stmt1 = connection.prepareStatement(query1); PreparedStatement stmt2 = connection.prepareStatement(query2)) {

stmt1.setString(1, title);

int rowsAff1 = stmt1.executeUpdate();

stmt2.setString(1, title);

int rowsAff2 = stmt2.executeUpdate();

if (rowsAff1 > 0 && rowsAff2 > 0) {

System.out.println("Quiz " + title + " deleted successfully.");

((DefaultTableModel) TestTable.getModel()).removeRow(selectedRow);

} else {

System.out.println("Quiz " + title + " not found.");

}

} catch (SQLException e) {

e.printStackTrace();

}

} else {

System.out.println("No Quiz selected.");

}

}

public void searchTest(String searchTest, JTable TestTable) {

String query = "SELECT \* FROM APP.QUESTION WHERE TITLE LIKE ?";

try {

Connection connection = DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASSWD);

PreparedStatement preparedStatement = connection.prepareStatement(query);

preparedStatement.setString(1, "%" + searchTest + "%");

ResultSet resultSet = preparedStatement.executeQuery();

ResultSetMetaData metaData = resultSet.getMetaData();

int columnCount = metaData.getColumnCount();

Vector<String> columnNames = new Vector<>();

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

columnNames.add(metaData.getColumnName(i));

}

Vector<Vector<Object>> data = new Vector<>();

while (resultSet.next()) {

Vector<Object> row = new Vector<>();

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

row.add(resultSet.getObject(i));

}

data.add(row);

}

DefaultTableModel model = new DefaultTableModel(data, columnNames);

TestTable.setModel(model);

connection.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

public void viewTest(JTable TestTable) {

String query = "SELECT \* FROM APP.QUESTION";

try {

Connection connection = DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASSWD);

Statement statement = connection.createStatement();

ResultSet resultSet = statement.executeQuery(query);

ResultSetMetaData metaData = resultSet.getMetaData();

int columnCount = metaData.getColumnCount();

Vector<String> columnNames = new Vector<>();

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

columnNames.add(metaData.getColumnName(i));

}

Vector<Vector<Object>> data = new Vector<>();

while (resultSet.next()) {

Vector<Object> row = new Vector<>();

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

row.add(resultSet.getObject(i));

}

data.add(row);

}

DefaultTableModel model = new DefaultTableModel(data, columnNames);

TestTable.setModel(model);

connection.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

}

**TESTER CLASS:**

import java.sql.\*;

import java.util.\*;

import javax.swing.JTable;

import javax.swing.table.DefaultTableModel;

public class Tester {

String question;

String option1;

String option2;

String option3;

String answer;

static final String DB\_URL = "jdbc:derby://localhost:1527/connect";

static final String DB\_USER = "admin44";

static final String DB\_PASSWD = "Qwerty1234";

public Tester() {

}

public boolean save(String username, int mark, double procenti, double punkti, String test, int repeat) {

String insertQuery = "INSERT INTO APP.REZULTATI (USERNAME,MARK,PROCENTI,PUNKTI,TEST,REPEAT) VALUES (?,?,?,?,?,?)";

try ( Connection connection = DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASSWD); PreparedStatement insertStmt = connection.prepareStatement(insertQuery)) {

insertStmt.setString(1, username);

insertStmt.setInt(2, mark);

insertStmt.setDouble(3, procenti);

insertStmt.setDouble(4, punkti);

insertStmt.setString(5, test);

insertStmt.setInt(6, repeat);

int rowsAff = insertStmt.executeUpdate();

return rowsAff > 0;

} catch (SQLException e) {

e.printStackTrace();

return false;

}

}

public void searchRezult(String searchUser, JTable rezultTable) {

String query = "SELECT \* FROM APP.REZULTATI WHERE USERNAME LIKE ?";

try {

Connection connection = DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASSWD);

PreparedStatement preparedStatement = connection.prepareStatement(query);

preparedStatement.setString(1, "%" + searchUser + "%");

ResultSet resultSet = preparedStatement.executeQuery();

ResultSetMetaData metaData = resultSet.getMetaData();

int columnCount = metaData.getColumnCount();

Vector<String> columnNames = new Vector<>();

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

columnNames.add(metaData.getColumnName(i));

}

Vector<Vector<Object>> data = new Vector<>();

while (resultSet.next()) {

Vector<Object> row = new Vector<>();

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

row.add(resultSet.getObject(i));

}

data.add(row);

}

DefaultTableModel model = new DefaultTableModel(data, columnNames);

rezultTable.setModel(model);

connection.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

public void getResults(JTable rezultTable) {

String query = "SELECT \* FROM APP.REZULTATI";

try {

Connection connection = DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASSWD);

Statement statement = connection.createStatement();

ResultSet resultSet = statement.executeQuery(query);

ResultSetMetaData metaData = resultSet.getMetaData();

int columnCount = metaData.getColumnCount();

Vector<String> columnNames = new Vector<>();

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

columnNames.add(metaData.getColumnName(i));

}

Vector<Vector<Object>> data = new Vector<>();

while (resultSet.next()) {

Vector<Object> row = new Vector<>();

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

row.add(resultSet.getObject(i));

}

data.add(row);

}

DefaultTableModel model = new DefaultTableModel(data, columnNames);

rezultTable.setModel(model);

connection.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

}

# Pielikums B. testa programmas kods

import java.awt.Color;

import java.sql.\*;

import javax.swing.\*;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.swing.JOptionPane;

/\*

\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

\* Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JFrame.java to edit this template

\*/

/\*\*

\*

\* @author user

\*/

public class GUI extends javax.swing.JFrame {

private String title;

private String[][] testData;

private int punkt = 0;

private int megsk = 0;

public GUI() {

initComponents();

this.setLocationRelativeTo(null);

}

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

Question test = new Question(VisiTesti);

User db = new User();

String login = Login.getText();

String password = Password.getText();

String grup = db.login(login, password);

if (db.loginstatus == 1) {

this.setVisible(false);

if ("Skolotajs".equals(grup)) {

test.loadTitles();

AdminPan.setBounds(0,0,592, 342);

AdminPan.setLocationRelativeTo(null);

AdminPan.setVisible(true);

jMenuBar1.setVisible(true);

} else {

test.loadTitles();

AdminPan.pack();

AdminPan.setLocationRelativeTo(null);

AdminPan.setVisible(true);

jMenuBar1.setVisible(false);

}

} else {

logerror.setVisible(true);

logerror.pack();

logerror.setLocationRelativeTo(null);

errorL.setText("Invalid login or password");

}

}

private void RegMouseClicked(java.awt.event.MouseEvent evt) {

this.setVisible(false);

Registr.pack();

Registr.setLocationRelativeTo(null);

Registr.setVisible(true);

}

private void RemPassMouseClicked(java.awt.event.MouseEvent evt) {

PassM.pack();

PassM.setLocationRelativeTo(null);

PassM.setVisible(true);

}

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

User db = new User();

String username = LoginR.getText();

String birth = Date.getText();

String gender = Gender.getSelectedItem().toString();

String password = PasswordR.getText();

String grup = Grup.getSelectedItem().toString();

try {

db.register(username, gender, password, birth, grup);

} catch (SQLException ex) {

Logger.getLogger(GUI.class.getName()).log(Level.SEVERE, null, ex);

}

db.sendToUsers(username, password, grup);

PasswordR.setText("");

LoginR.setText("");

Date.setText("");

Registr.dispose();

setVisible(true);

}

private void iziet1ActionPerformed(java.awt.event.ActionEvent evt) {

System.exit(0);

}

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

System.exit(0);

}

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

Question db = new Question(VisiTesti);

title = VisiTesti.getSelectedValue();

if (title != null) {

testData = db.getTestData(title);

jLabel29.setText(title);

Quest1.setText(testData[0][0]);

JRB1.setText(testData[0][1]);

JRB2.setText(testData[0][2]);

JRB3.setText(testData[0][3]);

Quest2.setText(testData[1][0]);

JRB4.setText(testData[1][1]);

JRB5.setText(testData[1][2]);

JRB6.setText(testData[1][3]);

Quest3.setText(testData[2][0]);

JRB7.setText(testData[2][1]);

JRB8.setText(testData[2][2]);

JRB9.setText(testData[2][3]);

Quest4.setText(testData[3][0]);

JRB10.setText(testData[3][1]);

JRB11.setText(testData[3][2]);

JRB12.setText(testData[3][3]);

Quest5.setText(testData[4][0]);

JRB13.setText(testData[4][1]);

JRB14.setText(testData[4][2]);

JRB15.setText(testData[4][3]);

JRB1.setActionCommand(JRB1.getText());

JRB2.setActionCommand(JRB2.getText());

JRB3.setActionCommand(JRB3.getText());

JRB4.setActionCommand(JRB4.getText());

JRB5.setActionCommand(JRB5.getText());

JRB6.setActionCommand(JRB6.getText());

JRB7.setActionCommand(JRB7.getText());

JRB8.setActionCommand(JRB8.getText());

JRB9.setActionCommand(JRB9.getText());

JRB10.setActionCommand(JRB10.getText());

JRB11.setActionCommand(JRB11.getText());

JRB12.setActionCommand(JRB12.getText());

JRB13.setActionCommand(JRB13.getText());

JRB14.setActionCommand(JRB14.getText());

JRB15.setActionCommand(JRB15.getText());

ANS1.setText("");

ANS2.setText("");

ANS3.setText("");

ANS4.setText("");

ANS5.setText("");

BG1.clearSelection();

BG2.clearSelection();

BG3.clearSelection();

BG4.clearSelection();

BG5.clearSelection();

JRB1.setEnabled(true);

JRB2.setEnabled(true);

JRB3.setEnabled(true);

JRB4.setEnabled(true);

JRB5.setEnabled(true);

JRB6.setEnabled(true);

JRB7.setEnabled(true);

JRB8.setEnabled(true);

JRB9.setEnabled(true);

JRB10.setEnabled(true);

JRB11.setEnabled(true);

JRB12.setEnabled(true);

JRB13.setEnabled(true);

JRB14.setEnabled(true);

JRB15.setEnabled(true);

megsk++;

Test.pack();

Test.setLocationRelativeTo(null);

Test.setVisible(true);

} else {

logerror.setVisible(true);

logerror.pack();

logerror.setLocationRelativeTo(null);

errorL.setText("Please select a test.");

}

}

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

Question db = new Question(VisiTesti);

punkt = 0;

String checkAnswer1 = "SELECT \* FROM APP.ANSWER WHERE TITLE=? AND AN1 =? ";

String checkAnswer2 = "SELECT \* FROM APP.ANSWER WHERE TITLE=? AND AN2 =? ";

String checkAnswer3 = "SELECT \* FROM APP.ANSWER WHERE TITLE=? AND AN3 =? ";

String checkAnswer4 = "SELECT \* FROM APP.ANSWER WHERE TITLE=? AND AN4 =? ";

String checkAnswer5 = "SELECT \* FROM APP.ANSWER WHERE TITLE=? AND AN5 =? ";

title = VisiTesti.getSelectedValue();

try (Connection connection = DriverManager.getConnection(Question.DB\_URL, Question.DB\_USER, Question.DB\_PASSWD);) {

PreparedStatement pst;

ResultSet rs;

pst = connection.prepareStatement(checkAnswer1);

pst.setString(1, title);

pst.setString(2, BG1.getSelection().getActionCommand());

rs = pst.executeQuery();

if (rs.next()) {

punkt++;

ANS1.setText("Pareiza atbilde");

ANS1.setForeground(Color.GREEN);

} else {

ANS1.setText("Atbilde nav pareiza");

ANS1.setForeground(Color.RED);

}

JRB1.setEnabled(false);

JRB2.setEnabled(false);

JRB3.setEnabled(false);

pst = connection.prepareStatement(checkAnswer2);

pst.setString(1, title);

pst.setString(2, BG2.getSelection().getActionCommand());

rs = pst.executeQuery();

if (rs.next()) {

punkt++;

ANS2.setText("Pareiza atbilde");

ANS2.setForeground(Color.GREEN);

} else {

ANS2.setText("Atbilde nav pareiza");

ANS2.setForeground(Color.RED);

}

JRB4.setEnabled(false);

JRB5.setEnabled(false);

JRB6.setEnabled(false);

pst = connection.prepareStatement(checkAnswer3);

pst.setString(1, title);

pst.setString(2, BG3.getSelection().getActionCommand());

rs = pst.executeQuery();

if (rs.next()) {

punkt++;

ANS3.setText("Pareiza atbilde");

ANS3.setForeground(Color.GREEN);

} else {

ANS3.setText("Atbilde nav pareiza");

ANS3.setForeground(Color.RED);

}

JRB7.setEnabled(false);

JRB8.setEnabled(false);

JRB9.setEnabled(false);

pst = connection.prepareStatement(checkAnswer4);

pst.setString(1, title);

pst.setString(2, BG4.getSelection().getActionCommand());

rs = pst.executeQuery();

if (rs.next()) {

punkt++;

ANS4.setText("Pareiza atbilde");

ANS4.setForeground(Color.GREEN);

} else {

ANS4.setText("Atbilde nav pareiza");

ANS4.setForeground(Color.RED);

}

JRB10.setEnabled(false);

JRB11.setEnabled(false);

JRB12.setEnabled(false);

pst = connection.prepareStatement(checkAnswer5);

pst.setString(1, title);

pst.setString(2, BG5.getSelection().getActionCommand());

rs = pst.executeQuery();

if (rs.next()) {

punkt++;

ANS5.setText("Pareiza atbilde");

ANS5.setForeground(Color.GREEN);

} else {

ANS5.setText("Atbilde nav pareiza");

ANS5.setForeground(Color.RED);

}

JRB13.setEnabled(false);

JRB14.setEnabled(false);

JRB15.setEnabled(false);

} catch (SQLException e) {

logerror.setVisible(true);

logerror.pack();

logerror.setLocationRelativeTo(null);

errorL.setText("Please select a test answer.");

}

}

private void AddNewTestMouseClicked(java.awt.event.MouseEvent evt) {

this.setVisible(false);

TestTools.pack();

TestTools.setLocationRelativeTo(null);

TestTools.setVisible(true);

}

private void ViewUsMouseClicked(java.awt.event.MouseEvent evt) {

this.setVisible(false);

ViewUser.pack();

ViewUser.setLocationRelativeTo(null);

ViewUser.setVisible(true);

Admin db = new Admin();

db.viewuser(jTable1);

}

private void SaerchKeyReleased(java.awt.event.KeyEvent evt) {

Admin db = new Admin();

String searchUser = Saerch.getText();

db.searchUser(searchUser, jTable1);

}

private void Iziet3ActionPerformed(java.awt.event.ActionEvent evt) {

ViewUser.dispose();

AdminPan.setVisible(true);

}

private void DeleteUsMouseClicked(java.awt.event.MouseEvent evt) {

this.setVisible(false);

DeleteUser.pack();

DeleteUser.setLocationRelativeTo(null);

DeleteUser.setVisible(true);

Admin db = new Admin();

db.viewuser(jTable2);

}

private void Search2KeyReleased(java.awt.event.KeyEvent evt) {

Admin db = new Admin();

String searchUser = Search2.getText();

db.searchUser(searchUser, jTable2);

}

private void jTable2MouseClicked(java.awt.event.MouseEvent evt) {

int selectedRow = jTable2.getSelectedRow();

if (selectedRow >= 0) {

String username = (String) jTable2.getValueAt(selectedRow, 0);

int response = JOptionPane.showConfirmDialog(null, "Vai tiešām vēlaties dzēst lietotāju " + username + "?", "Dzēšanas apstiprināšana", JOptionPane.YES\_NO\_OPTION);

if (response == JOptionPane.YES\_OPTION) {

Admin db = new Admin();

db.removeUser(jTable2);

}

} else {

JOptionPane.showMessageDialog(null, "Lūdzu, atlasiet dzēšamo lietotāju.");

}

}

private void Iziet4ActionPerformed(java.awt.event.ActionEvent evt) {

DeleteUser.dispose();

}

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

User db = new User();

String username = UserM.getText();

String newPassword = PassMa.getText();

String confirmPassword = PassC.getText();

if (username.isEmpty() || newPassword.isEmpty() || confirmPassword.isEmpty()) {

JOptionPane.showMessageDialog(this, "Aizpildiet visus laukus.", "Kļūda", JOptionPane.ERROR\_MESSAGE);

return;

}

if (!newPassword.equals(confirmPassword)) {

JOptionPane.showMessageDialog(this, "Paroles nesakrīt.", "Kļūda", JOptionPane.ERROR\_MESSAGE);

return;

}

if (db.changePassword(username, newPassword)) {

JOptionPane.showMessageDialog(this, "Parole ir veiksmīgi atjaunināta.");

PassM.dispose();

} else {

JOptionPane.showMessageDialog(this, "Kļūda, atjauninot paroli.", "Kļūda", JOptionPane.ERROR\_MESSAGE);

}

}

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

PassM.dispose();

}

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

if (Paradit.isSelected()) {

PassMa.setEchoChar((char) 0);

PassC.setEchoChar((char) 0);

} else {

PassMa.setEchoChar('•');

PassC.setEchoChar('•');

}

}

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

if (ParaditPass.isSelected()) {

Password.setEchoChar((char) 0);

} else {

Password.setEchoChar('•');

}

}

private void Iziet2ActionPerformed(java.awt.event.ActionEvent evt) {

AdminPan.dispose();

setVisible(true);

}

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

Question db = new Question(VisiTesti);

title = VisiTesti.getSelectedValue();

if (title != null) {

testData = db.getTestData(title);

jLabel29.setText(title);

Quest1.setText(testData[0][0]);

JRB1.setText(testData[0][1]);

JRB2.setText(testData[0][2]);

JRB3.setText(testData[0][3]);

Quest2.setText(testData[1][0]);

JRB4.setText(testData[1][1]);

JRB5.setText(testData[1][2]);

JRB6.setText(testData[1][3]);

Quest3.setText(testData[2][0]);

JRB7.setText(testData[2][1]);

JRB8.setText(testData[2][2]);

JRB9.setText(testData[2][3]);

Quest4.setText(testData[3][0]);

JRB10.setText(testData[3][1]);

JRB11.setText(testData[3][2]);

JRB12.setText(testData[3][3]);

Quest5.setText(testData[4][0]);

JRB13.setText(testData[4][1]);

JRB14.setText(testData[4][2]);

JRB15.setText(testData[4][3]);

JRB1.setActionCommand(JRB1.getText());

JRB2.setActionCommand(JRB2.getText());

JRB3.setActionCommand(JRB3.getText());

JRB4.setActionCommand(JRB4.getText());

JRB5.setActionCommand(JRB5.getText());

JRB6.setActionCommand(JRB6.getText());

JRB7.setActionCommand(JRB7.getText());

JRB8.setActionCommand(JRB8.getText());

JRB9.setActionCommand(JRB9.getText());

JRB10.setActionCommand(JRB10.getText());

JRB11.setActionCommand(JRB11.getText());

JRB12.setActionCommand(JRB12.getText());

JRB13.setActionCommand(JRB13.getText());

JRB14.setActionCommand(JRB14.getText());

JRB15.setActionCommand(JRB15.getText());

Test.pack();

Test.setLocationRelativeTo(null);

Test.setVisible(true);

} else {

logerror.setVisible(true);

logerror.pack();

logerror.setLocationRelativeTo(null);

errorL.setText("Please select a test.");

}

}

private void jButton12ActionPerformed(java.awt.event.ActionEvent evt) {

TestTools.dispose();

}

private void jButton11ActionPerformed(java.awt.event.ActionEvent evt) {

Admin db = new Admin();

Question test = new Question(VisiTesti);

String title1 = Title.getText();

String question1 = qu1.getText(), question2 = qu2.getText(), question3 = qu3.getText(), question4 = qu4.getText(), question5 = qu5.getText();

String option1\_1 = op1\_1.getText(), option1\_2 = op1\_2.getText(), option1\_3 = op1\_3.getText(), option1\_4 = op1\_4.getText(), option1\_5 = op1\_5.getText();

String option2\_1 = op2\_1.getText(), option2\_2 = op2\_2.getText(), option2\_3 = op2\_3.getText(), option2\_4 = op2\_4.getText(), option2\_5 = op2\_5.getText();

String option3\_1 = op3\_1.getText(), option3\_2 = op3\_2.getText(), option3\_3 = op3\_3.getText(), option3\_4 = op3\_4.getText(), option3\_5 = op3\_5.getText();

String answer1 = an1.getText(), answer2 = an2.getText(), answer3 = an3.getText(), answer4 = an4.getText(), answer5 = an5.getText();

db.addQuestion(title1, question1, question2, question3, question4, question5, option1\_1, option2\_1, option3\_1, option1\_2, option2\_2, option3\_2, option1\_3, option2\_3, option3\_3, option1\_4, option2\_4, option3\_4, option1\_5, option2\_5, option3\_5);

db.addAnswer(answer1, answer2, answer3, answer4, answer5, title1);

test.loadTitles();

Title.setText("");

op1\_1.setText("Option1");

op1\_2.setText("Option1");

op1\_3.setText("Option1");

op1\_4.setText("Option1");

op1\_5.setText("Option1");

op2\_1.setText("Option2");

op2\_2.setText("Option2");

op2\_3.setText("Option2");

op2\_4.setText("Option2");

op2\_5.setText("Option2");

qu1.setText("Question1");

qu2.setText("Question2");

qu3.setText("Question3");

qu4.setText("Question4");

qu5.setText("Question5");

op3\_1.setText("Option3");

op3\_2.setText("Option3");

op3\_3.setText("Option3");

op3\_4.setText("Option3");

op3\_5.setText("Option3");

}

private void jTable3MouseClicked(java.awt.event.MouseEvent evt) {

int selectedRow = jTable3.getSelectedRow();

if (selectedRow >= 0) {

String title = String.valueOf(jTable3.getValueAt(selectedRow, 1));

int response = JOptionPane.showConfirmDialog(null, "Vai tiešām vēlaties dzēst Testu " + title + "?", "Dzēšanas apstiprināšana", JOptionPane.YES\_NO\_OPTION);

if (response == JOptionPane.YES\_OPTION) {

Question db = new Question(VisiTesti);

db.removeTest(jTable3);

}

} else {

JOptionPane.showMessageDialog(null, "Lūdzu, atlasiet dzēšamo Testu.");

}

}

private void SearchTestKeyReleased(java.awt.event.KeyEvent evt) {

Question db = new Question(VisiTesti);

String searchTest = SearchTest.getText();

db.searchTest(searchTest, jTable3);

}

private void Iziet5ActionPerformed(java.awt.event.ActionEvent evt) {

Question db = new Question(VisiTesti);

db.loadTitles();

DeleteTest.dispose();

}

private void DeleteTestiMouseClicked(java.awt.event.MouseEvent evt) {

this.setVisible(false);

DeleteTest.pack();

DeleteTest.setLocationRelativeTo(null);

DeleteTest.setVisible(true);

Question db = new Question(VisiTesti);

db.viewTest(jTable3);

}

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

Tester db = new Tester();

Test.setVisible(false);

Rezult.pack();

Rezult.setLocationRelativeTo(null);

Rezult.setVisible(true);

double punkti = (punkt \* 2);

Punkt.setText("Punkti:" + punkti + "/10.0");

double procenti = (punkti / 10) \* 100;

Proc.setText("Procenti:" + procenti + "%/100.0%");

int ball = 0;

switch (punkt \* 2) {

case 2:

ball = 1;

Atz.setText("Mark:" + ball + "/5");

Proc.setForeground(Color.RED);

Atz.setForeground(Color.RED);

Punkt.setForeground(Color.RED);

break;

case 4:

ball = 2;

Atz.setText("Mark:" + ball + "/5");

Proc.setForeground(Color.RED);

Atz.setForeground(Color.RED);

Punkt.setForeground(Color.RED);

break;

case 6:

ball = 3;

Atz.setText("Mark:" + ball + "/5");

Proc.setForeground(Color.YELLOW);

Atz.setForeground(Color.YELLOW);

Punkt.setForeground(Color.YELLOW);

break;

case 8:

ball = 4;

Atz.setText("Mark:" + ball + "/5");

Proc.setForeground(Color.GREEN);

Atz.setForeground(Color.GREEN);

Punkt.setForeground(Color.GREEN);

break;

case 10:

ball = 5;

Atz.setText("Mark:" + ball + "/5");

Proc.setForeground(Color.GREEN);

Atz.setForeground(Color.GREEN);

Punkt.setForeground(Color.GREEN);

break;

}

if (megsk == 1) {

Megsk.setText("Mēģinajuma skaits: 1/1");

Meiginat.setEnabled(false);

} else {

Megsk.setText("Mēģinajuma skaits: 0/1");

Meiginat.setEnabled(true);

}

String test = VisiTesti.getSelectedValue();

String username = Login.getText();

Nosaukums.setText("Testa nosaukums:"+"\n"+test);

UserR.setText("User name:"+"\n"+username);

db.save(username, ball, procenti, punkti, test, megsk);

}

private void SearchRKeyReleased(java.awt.event.KeyEvent evt) {

Tester dats = new Tester();

String searchTest = SearchR.getText();

dats.searchRezult(searchTest, jTable4);

}

private void Iziet6ActionPerformed(java.awt.event.ActionEvent evt) {

ViewRezult.dispose();

AdminPan.setVisible(true);

}

private void AllSudRezMouseClicked(java.awt.event.MouseEvent evt) {

ViewRezult.pack();

ViewRezult.setLocationRelativeTo(null);

ViewRezult.setVisible(true);

Tester dats = new Tester();

dats.getResults(jTable4);

}

private void Sag\_IzActionPerformed(java.awt.event.ActionEvent evt) {

megsk = 0;

ANS1.setText("");

ANS2.setText("");

ANS3.setText("");

ANS4.setText("");

ANS5.setText("");

BG1.clearSelection();

BG2.clearSelection();

BG3.clearSelection();

BG4.clearSelection();

BG5.clearSelection();

JRB1.setEnabled(true);

JRB2.setEnabled(true);

JRB3.setEnabled(true);

JRB4.setEnabled(true);

JRB5.setEnabled(true);

JRB6.setEnabled(true);

JRB7.setEnabled(true);

JRB8.setEnabled(true);

JRB9.setEnabled(true);

JRB10.setEnabled(true);

JRB11.setEnabled(true);

JRB12.setEnabled(true);

JRB13.setEnabled(true);

JRB14.setEnabled(true);

JRB15.setEnabled(true);

Rezult.setVisible(false);

Tester dats = new Tester();

dats.getResults(jTable4);

}

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

logerror.dispose();

}

private void jLabel27MouseClicked(java.awt.event.MouseEvent evt) {

ImageIcon imageIcon = new ImageIcon(getClass().getResource("/icone/HELP1.jpg"));

image.setIcon(imageIcon);

HELPS.setViewportView(image);

Help.setVisible(true);

Help.pack();

Help.setLocationRelativeTo(null);

}

private void jLabel28MouseClicked(java.awt.event.MouseEvent evt) {

ImageIcon imageIcon = new ImageIcon(getClass().getResource("/icone/HELP2.jpg"));

image.setIcon(imageIcon);

HELPS.setViewportView(image);

Help.setVisible(true);

Help.pack();

Help.setLocationRelativeTo(null);

}

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

Help.dispose();

}

private void jLabel41MouseClicked(java.awt.event.MouseEvent evt) {

User db = new User();

String login = Login.getText();

String password = Password.getText();

String grup = db.login(login, password);

if("Skolotajs".equals(grup)){

ImageIcon imageIcon1 = new ImageIcon(getClass().getResource("/icone/HELP8.jpg"));

image.setIcon(imageIcon1);

HELPS.setViewportView(image);

Help.setVisible(true);

Help.pack();

Help.setLocationRelativeTo(null);

}else{

ImageIcon imageIcon = new ImageIcon(getClass().getResource("/icone/HELP3.jpg"));

image.setIcon(imageIcon);

HELPS.setViewportView(image);

Help.setVisible(true);

Help.pack();

Help.setLocationRelativeTo(null);

}

}

private void jLabel35MouseClicked(java.awt.event.MouseEvent evt) {

ImageIcon imageIcon = new ImageIcon(getClass().getResource("/icone/HELP4.jpg"));

image.setIcon(imageIcon);

HELPS.setViewportView(image);

Help.setVisible(true);

Help.pack();

Help.setLocationRelativeTo(null);

}

private void jLabel42MouseClicked(java.awt.event.MouseEvent evt) {

ImageIcon imageIcon = new ImageIcon(getClass().getResource("/icone/HELP5.jpg"));

image.setIcon(imageIcon);

HELPS.setViewportView(image);

Help.setVisible(true);

Help.pack();

Help.setLocationRelativeTo(null);

}

private void jLabel43MouseClicked(java.awt.event.MouseEvent evt) {

ImageIcon imageIcon = new ImageIcon(getClass().getResource("/icone/HELP6.jpg"));

image.setIcon(imageIcon);

HELPS.setViewportView(image);

Help.setVisible(true);

Help.pack();

Help.setLocationRelativeTo(null);

}

private void jLabel44MouseClicked(java.awt.event.MouseEvent evt) {

ImageIcon imageIcon = new ImageIcon(getClass().getResource("/icone/HELP7.jpg"));

image.setIcon(imageIcon);

HELPS.setViewportView(image);

Help.setVisible(true);

Help.pack();

Help.setLocationRelativeTo(null);

}

private void jLabel45MouseClicked(java.awt.event.MouseEvent evt) {

ImageIcon imageIcon = new ImageIcon(getClass().getResource("/icone/HELP9.jpg"));

image.setIcon(imageIcon);

HELPS.setViewportView(image);

Help.setVisible(true);

Help.pack();

Help.setLocationRelativeTo(null);

}

private void jLabel48MouseClicked(java.awt.event.MouseEvent evt) {

ImageIcon imageIcon = new ImageIcon(getClass().getResource("/icone/HELP10.jpg"));

image.setIcon(imageIcon);

HELPS.setViewportView(image);

Help.setVisible(true);

Help.pack();

Help.setLocationRelativeTo(null);

}

private void jLabel26MouseClicked(java.awt.event.MouseEvent evt) {

ImageIcon imageIcon = new ImageIcon(getClass().getResource("/icone/HELP11.jpg"));

image.setIcon(imageIcon);

HELPS.setViewportView(image);

Help.setVisible(true);

Help.pack();

Help.setLocationRelativeTo(null);

}