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.0007.007PR**

Profesijas kods 33 481 031 Programmēšanas nodaļa

PR-21. grupa

|  |  |  |
| --- | --- | --- |
| Izveidoja | paraksts | Roberts Erts  V.,Uzvārds |
| Pārbaudīja | paraksts | I. Dortiņa  V.,Uzvārds |

2024./2025. 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.

Programma ļauj ievadīt nepieciešamus datus, lai izveidotu savu kontu. Visi ievadīti dati tiek saglabāti datu bāzē.

# 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

Rezultāti

Skolotājs

Tests

Lietotājs

* 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;
* 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 saskarne

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.](#_bookmark8) [att.](#_bookmark8)), tas faktiski ir viedokļu kopums, kur katra viedokļa nosaukums ir ierakstīts atsevišķa elipsē.

Nefunkcionalie viedokli

Funkcionalie viedokli

Viedokli par lietotaju

Datu viedokļi

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

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

Testēšanas sistēma (V0)

Datu bāze (V12)

Lietotājs (V11)

Tests (V13)

Lietotāju registrt (V121)

Skolotājs (V111)

Testa jautajumu registrs (V122)

Audzēknis (V112)

Audzekņa funkcijas (V132)

Skolotāja funkcijas (V131)

Testa pildīšana (V1321)

Testa rezultātu saņemšana (V1322)

Testa manuāla novērtēšana (V1311)

* 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

Pilda

Tests

Skolotājs

Izveido

Jautājums

Satur

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

# Datu vārdnīca

|  |  |  |
| --- | --- | --- |
| Datu tips | Nosaukums | Apraksts |
| String | **name** | Lietotājs ievada savu vārdu |
| String | **surname** | Lietotājs ievada savu uzvārdu |
| String | **login** | Lietotājs ievada savu loginu |
| String | **password** | Lietotājs ievada savu parole |
| String | **type** | Lietotājs izvēle savu tipu (Skolēns vai Skolotājs) |
| String | **text** | Neliels teksta apgabals, kas apraksta jautājumu |
| String[] | **answer** | Atbilžu variantu masīvs. Tiek piedāvāti vairāki atbilžu  varianti, kur ir jāizvēlas viens pareizs |
| int | **correctAnswer** | Aile, kas nosaka, vai tika izvēlēts pareizs atbildes variants |
| int | **percent** | Audzēkņa saņemto procentu skaits (0-100) |
| int | **grade** | Audzēkņa atzīme (0-10) |
| int | **rightAnswersCount** | Kopeja izveleto pareidzo atbīldes skaists |
| int | **totalQuestions** | Kopeju jautajumu skaists(20) |
| int | **avgPercent** | Videja saņemto procentu skaits (0-100) |
| int | average | Videja atzīme (0-10) |

# 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 20 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 testajautājumu”

### 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 “Rezultātu salīdzinošana ar vidējo klāse rezultātus”

### ID: T7

**Ievads:** Ļauj lietotājam salīdziniet savus rezultātus ar vidējo rezultātu. Visi rezultātu tiek saglabāti failā (datu bāzē).

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

**Apstrāde:** Sistēma saskaita vidējo rezultātus, tad nolasa vērtējumu par 10 balles skalu un salīdzina ar lietotaju rezultātus.

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

* 1. Vidējo klāse atzīme.
  2. Vidējo klāse punktu skaits.
  3. Vidējo klāse procenti par darbu.
  4. Lietotāja atzīme.
  5. Lietotāja punktu skaits.
  6. Lietotāja procenti par darbu.
  7. Lietotāja salīdzinošana ar vidējo klāse rezultātus.

## Funkcija “Atbildes apsaktīšana sistēma”

### ID: T8

**Ievads:** Ļauj lietotājam apskatīt savus atbildes un redzēt vai atbilde ir pareizo vai nē. Ja tā ir nepareiza ir iespejams nospiets/novietot kursoru uz atbilde un uzzināt pareizo atbilde.

**Ievade:** Lietotājs var noklikšķinat uz pogas "Apskatīt atbildes".

**Apstrāde:** Sistēma pārbauda atbildi uz uzdoto jautājumu un pēc tam sāka savu darbu apskatīšana.

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

* 1. Uzdevuma numurs.
  2. Jautājums.
  3. Lietotāja izveleto atbilde.
  4. Rezultats.

## Funkcija “Testajautājumu menu”

### ID: 9

**Ievads:** Ļauj lietotājam ne tikai parvietoties uz iepriekš, bet arī izvēlet konkretu jautajumu ar izvēlne cilne. Arī, ja lietotājs ieraksta atbildi uz jautājumu, tā tiks iekrāsota izvēlnes lodziņā.

**Ievade:** Lietotājs noklikšķina uz pogas "1" vai “2” vai “3” utt. vai uz pogu “Pabeigt darbs”

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

* 1. Visi uzdevumi.
  2. Lietotāja atbildes, kas iekrāsojas atkarībā no atbildes - ja atbilde ir pareiza, tā tiek atzīmēta zaļā krāsā, ja atbilde ir nepareiza, tā tiek atzīmēta sarkanā krāsā.

# 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)

Help

Reģistreties

Uzvards:

Lietotajvards:

Parole:

Parole (atkartoti):

Vards:

* 1. **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”
7. poga “Help”
   1. Lietotāja saskarne “Lietotāja ielogošanas saskarne”. Saskarnes struktūra: skat. [1.7. att.](#_bookmark17)

Reģistreties

Lietotājvārds:

Parole:

Ielogoties

Help

* 1. **att. Lietotāja ielogošanas saskarne**

## Saskarnes elementi:

1. teksta lauks “Lietotājvārds”
2. teksta lauks ‘Parole”
3. poga “Ielogoties”
4. poga “Reģistrēties”
5. poga “Help”
   1. Lietotāja saskarne “Testa izpilde”. Saskarnes struktūra: skat. [1.8. att.](#_bookmark18)

Sakt testu

Tests “Nosaukums”

Atteikties no konta

Talak

1. Jautajums “Jautajuma teksts”

1. atbilde

2. atbilde

3. atbilde

Jautajumi:

**1**

**2**

**4**

**3**

**5**

**6**

**8**

**7**

**9**

**10**

**12**

**11**

Pabeigt darbs

Beigt

**Jusu rezultati**

Procenti: X%

Atzime: A

Apskatīt atbildes

**Vidēji rezultati**

Procenti: X%

Atzime: A

c)

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

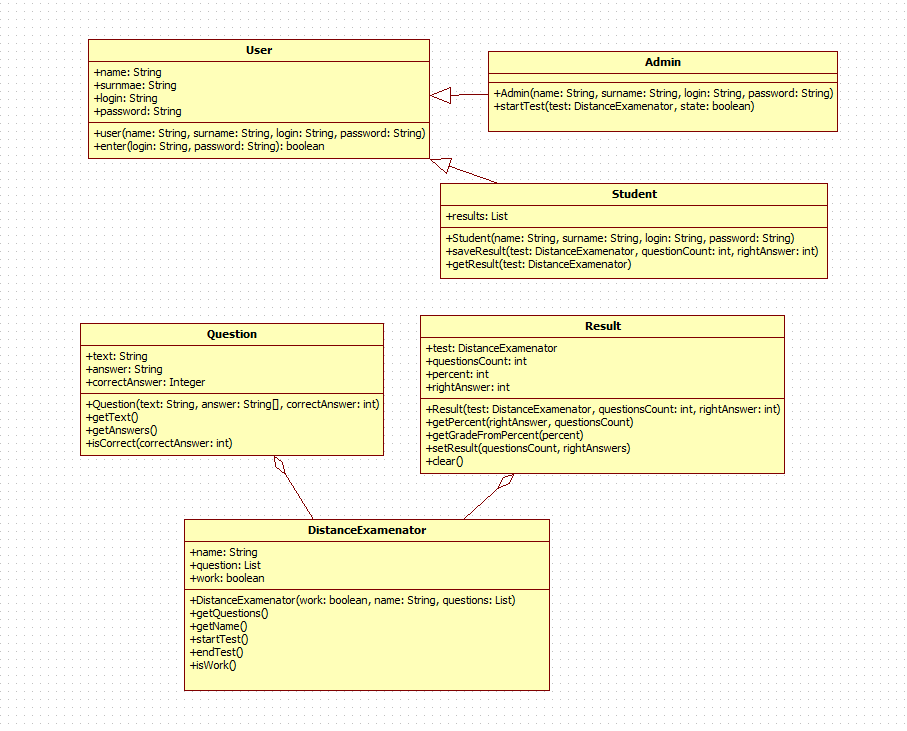
## Saskarnes elementi:

1. testa izvēle/sakums
2. poga “Sakt testu”
3. poga “Atteikties no konta”
4. atbilde uz jautājumu/-iem
5. atbilde izvēle cilne
6. poga “Talak”
7. Jautajumu izvēlne cilne
8. poga “Pabeigt darbs”
9. rezultāti
10. poga “Apskatīt atbildes”
11. poga “Beigt”

# 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 (name: String, login: String, password: String) – konstruktors, kurš izveido jaunu lietotāju ar norādītiem datiem;
   * enter (login: String, password: String) – metode, kura nodrošina lietotāju pieslēgšanu sistēmai;
2. Klase **Admin** – nodrošina darbu ar lietotajiem un testiem

* Admin(name: String, surname: String, login: String, password: String) – konstruktors, kas izveido administratoru.
* startTest(test: DistanceExamenator, state: boolean) – metode, kas ļauj administratoram sākt vai beigt testu, atkarībā no parametra state.

1. Klase **Student** – nodrošina audzēkņa testa pildīšanas darbus

* Student(name: String, surname: String, login: String, password: String) – konstruktors, kas izveido studentu.
* saveResult(test: DistanceExamenator, questionCount: int, rightAnswer: int) – metode, kas saglabā vai atjaunina testu rezultātus studentam.
* getResult(test: DistanceExamenator) – metode, kas atgriež rezultātu konkrētam testam.

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

* Question(text: String, answer: String[], correctAnswer: int) – konstruktors, kas izveido jautājumu ar atbilžu masīvu un pareizās atbildes indeksu.
* getText() – metode, kas atgriež jautājuma tekstu.
* getAnswers() – metode, kas atgriež atbilžu sarakstu.
* isCorrect(correctAnswer: int) – metode, kas atgriež pareizo atbildes tekstu pēc indeksa.

1. Klase **Result** – atbild par testa rezultātu glabāšanu un apstrādi

* Result(test: DistanceExamenator, questionsCount: int, rightAnswer: int) – konstruktors, kas saglabā datus par testu.
* getPercent(rightAnswer, questionsCount) – metode, kas aprēķina rezultātu procentos.
* getGradeFromPercent(percent) – metode, kas atgriež atzīmi skalā no 1 līdz 10 pēc procentiem.
* setResult(questionsCount, rightAnswers) – metode, kas atjaunina rezultātu.
* clear() – metode, kas nodzēš rezultātu (uzstāda 0 pareizo atbilžu un jautājumu skaitu).

1. Klase **DistanceExamenator** – programmas darbību koordinējošā klase. Tā nodrošina programmas galvenās metodes izsaukumu un grafiskās saskarnes izveidi, tā arī
   * DistanceExamenator(work: boolean, name: String, questions: ArrayList<Question>) – konstruktors, kas inicializē testu.
   * getQuestions() – metode, kas atgriež jautājumu sarakstu.
   * getName() – metode, kas atgriež testa nosaukumu.
   * startTest() / endTest() – metodes, kas sāk vai beidz testu.
   * isWork() – metode, kas parāda, vai tests ir aktīvs.

# 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](#_bookmark33)

[A. 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)”.

class TestingSystemErts{

public class User{

private String name;

private String surname;

private String login;

private String password;

public User(String name, String surname, String login, String password) {

this.name = name;

this.login = login;

this.password = password;

}

public boolean enter(String login, String password) {

return this.login.equals(login) && this.password.equals(password);

}

}

public class Student extends User{

private ArrayList<Result> results = new ArrayList();

public Student(String name, String surname, String login, String password) {

super(name, surname, login, password);

}

public Result getResult(DistanceExamenator test) {

for (Result result : results) {

if (result.getTest() == test) {

return result;

}

}

return null;

}

public void saveResult(DistanceExamenator test, int questionCount, int rightAnswer) {

Result result = getResult(test);

if (result == null) {

results.add(new Result(test, questionCount, rightAnswer));

} else {

result.setResult(questionCount, rightAnswer);

}

}

}

public class Admin extends User{

public Admin(String name, String surname, String login, String password) {

super(name, surname, login, password);

}

public void startTest(DistanceExamenator test, boolean state) {

if (state) {

test.startTest();

} else {

test.endTest();

}

}

}

public class Question{

private String text;

private String[] answer;

private int correctAnswer;

public Question(String text, String[] answer, int correctAnswer) {

this.text = text;

this.answer = answer;

this.correctAnswer = correctAnswer;

}

public String getText() {

return text;

}

public String[] getAnswers() {

return answer;

}

public int getCorrectAnswer(){

return correctAnswer;

}

public String isCorrect(int correctAnswer){

return answer[correctAnswer];

}

}

public class Result {

private final DistanceExamenator test;

private int questionsCount = 0;

private int rightAnswer = 0;

private int percent = 0;

public Result(DistanceExamenator test, int questionsCount, int rightAnswer){

this.test = test;

this.questionsCount = questionsCount;

this.rightAnswer = rightAnswer;

}

public int getPercent(int rightAnswer , int questionsCount){

this.percent = (int) ((rightAnswer / (double) questionsCount) \* 100);

return percent;

}

public int getGradeFromPercent(int percent) {

if (percent >= 95) return 10;

else if (percent >= 85) return 9;

else if (percent >= 75) return 8;

else if (percent >= 65) return 7;

else if (percent >= 55) return 6;

else if (percent >= 45) return 5;

else if (percent >= 35) return 4;

else if (percent >= 25) return 3;

else if (percent >= 15) return 2;

else return 1;

}

public DistanceExamenator getTest(){

return test;

}

public int getQuestionsCount(){

return questionsCount;

}

public int getRightAnswers(){

return rightAnswer;

}

public void setResult(int questionsCount, int rightAnswers){

this.questionsCount = questionsCount;

this.rightAnswer = rightAnswers;

}

public void clear(){

questionsCount = 0;

rightAnswer = 0;

}

}

public class DistanceExamenator{

private List<Question> questions = new ArrayList<>();

private String name;

private boolean work = false;

public DistanceExamenator(boolean work, String name, ArrayList questions) {

this.work = work;

this.name = name;

this.questions = questions;

}

public List<Question> getQuestions() {

return questions;

}

public String getName(){

return name;

}

public void startTest(){

work = true;

}

public void endTest(){

work = false;

}

public boolean isWork() {

return work;

}

}

}

# 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.

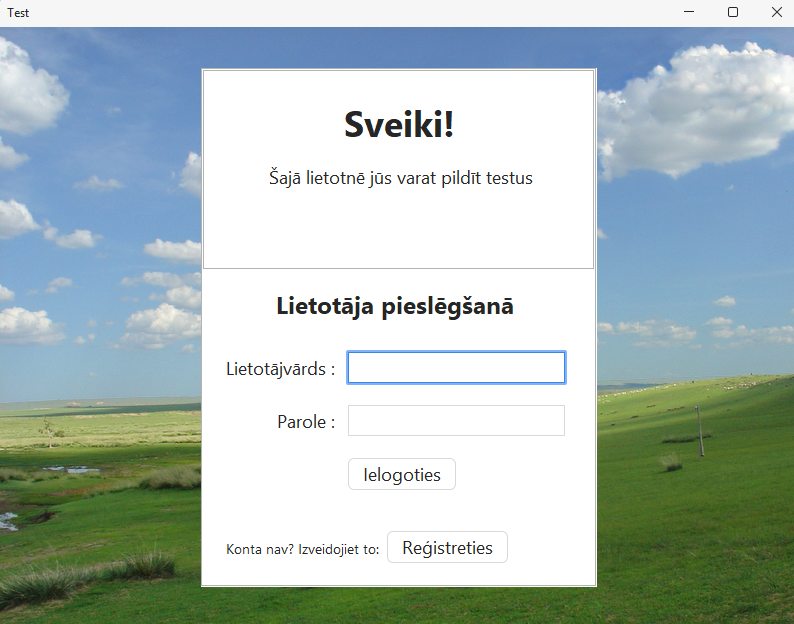
Jāatzīmē tie momenti un prasības, kas pastāv, izstrādājot testēšanas programmu.

* 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.x. att.).

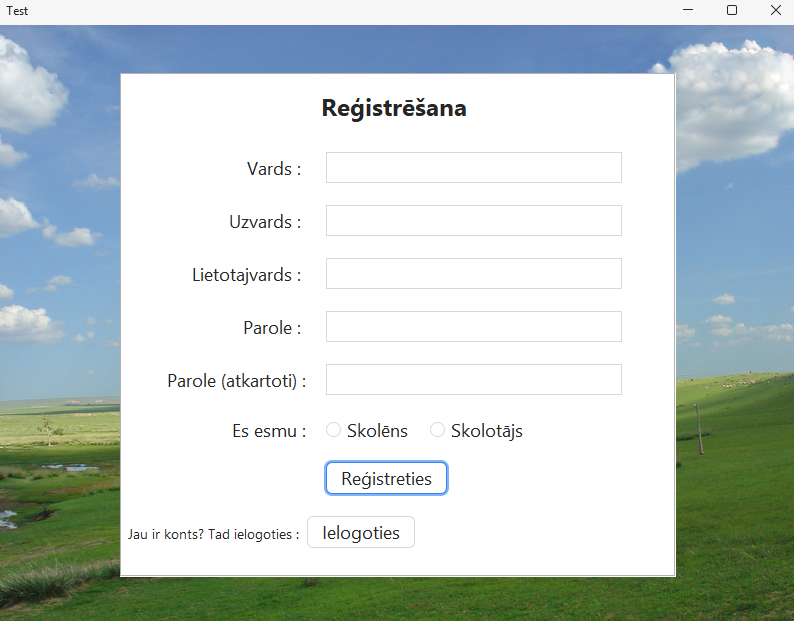
* + Ielogošanas procesā ir iespējamas vairākas kļūdas – neaizpildīti lauki, neprecīzi ievadīti lietotāja dati.



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

1. Reģistrācijas procesa apraksts:

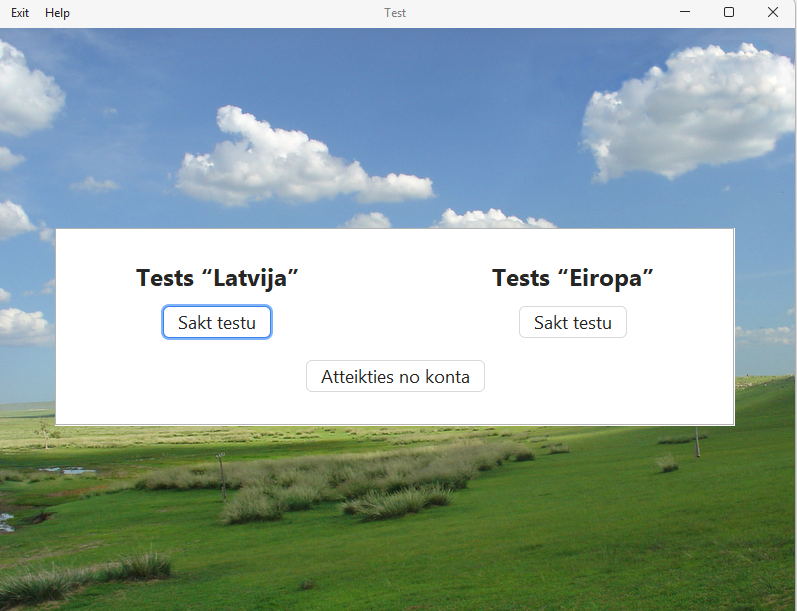
* Reģistrācijas forma paredzēta jaunu lietotāju pievienošanai sistēmai. Lietotājam ir jāaizpilda nepieciešamie lauki: vārds, uzvārds, lietotājvārds, parole, atkārtota parole, kā arī jānorāda, vai viņš ir skolēns vai skolotājs (3.2. att.).
* Sistēma pārbauda ievadīto datu korektumu – parolei jāsakrīt ar atkārtoto paroli, lietotājvārds nedrīkst būt jau reģistrēts u.c.
* Ja dati ir korekti, lietotājs tiek pievienots datubāzei, un viņam ir iespēja autorizēties sistēmā.



**3.2. att. Testēšanas sistēmas reģistrēšana forma**

1. Lietotāja interfeiss pēc ielogošanās:

* Pēc veiksmīgas autorizācijas lietotājam tiek piedāvāta iespēja uzsākt testu vai atteikties no konta (3.3. att.).
* Lietotāja skatā tiek parādīts testa nosaukums, un ir pieejama poga "Sākt testu".

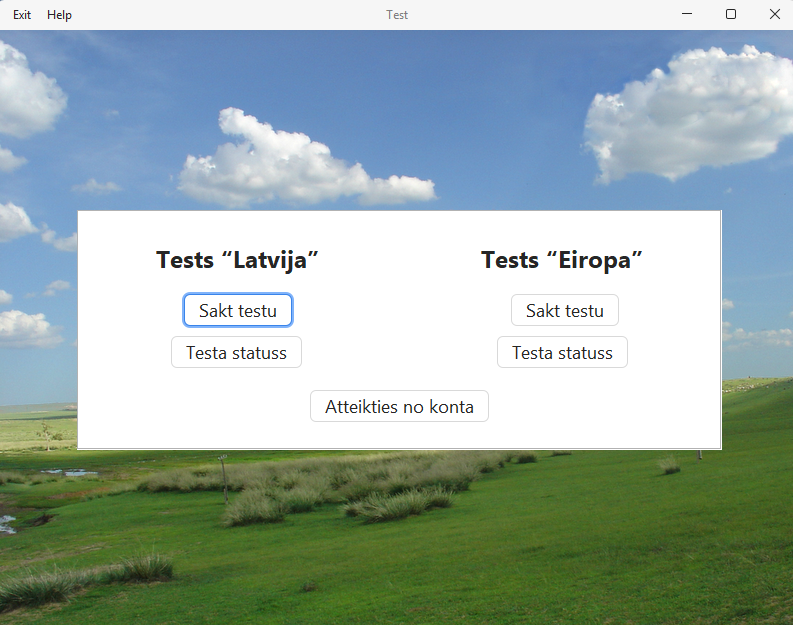


**3.3. att. Testēšanas sistēmas lietotaju testa izpilde forma**

1. Administratora skats:

• Administratoram pēc ielogošanās tiek piedāvātas papildiespējas, piemēram, skatīt testa statusu (3.4. att.).

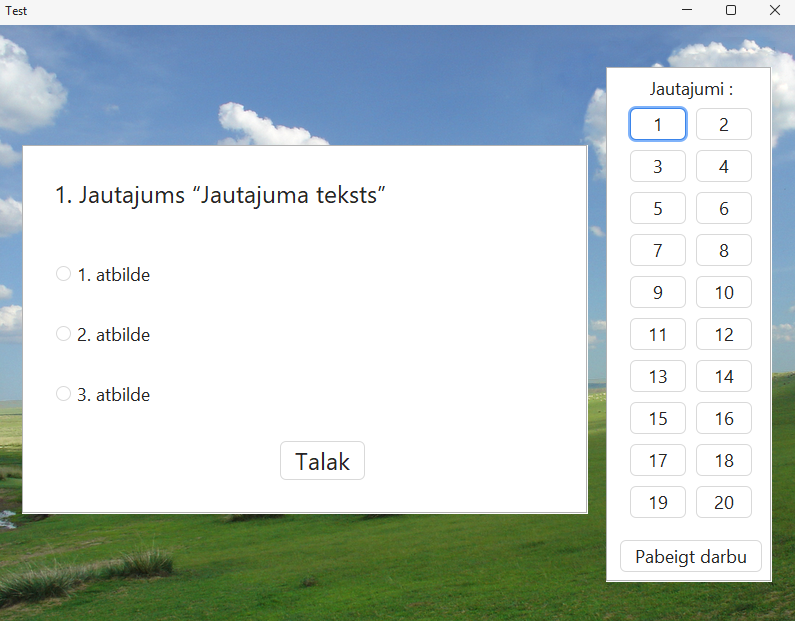
• Tas ļauj sekot līdzi testēšanas gaitai un analizēt rezultātus.



**3.4. att. Testēšanas sistēmas administartaru testa izpilde forma**

1. Testa veikšanas forma:

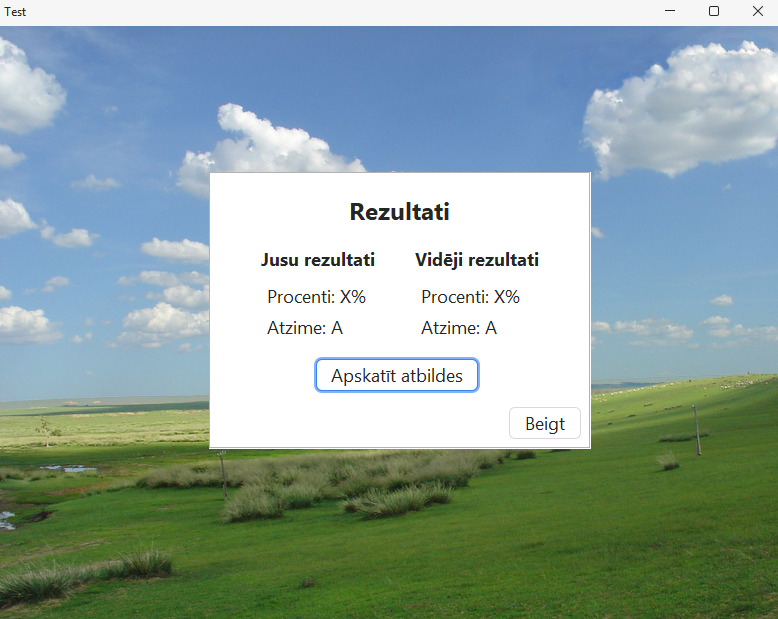
* Testa laikā lietotājam tiek uzdoti jautājumi ar atbilžu variantiem (3.5. att.).
* Lietotājs var pārvietoties starp jautājumiem, izmantojot numerētos pogas labajā pusē. Ir iespēja arī pabeigt darbu jebkurā brīdī.
* Sistēma saglabā atbildes un nodrošina to validāciju.



**3.5. att. Testēšanas sistēmas testa forma**

1. Rezultātu forma:

* Pēc testa pabeigšanas tiek parādīti rezultāti – iegūtie procenti, vērtējums, kā arī salīdzinājums ar vidējiem rezultātiem (3.6. att.).
* Lietotājs var apskatīt savas atbildes vai beigt sesiju.

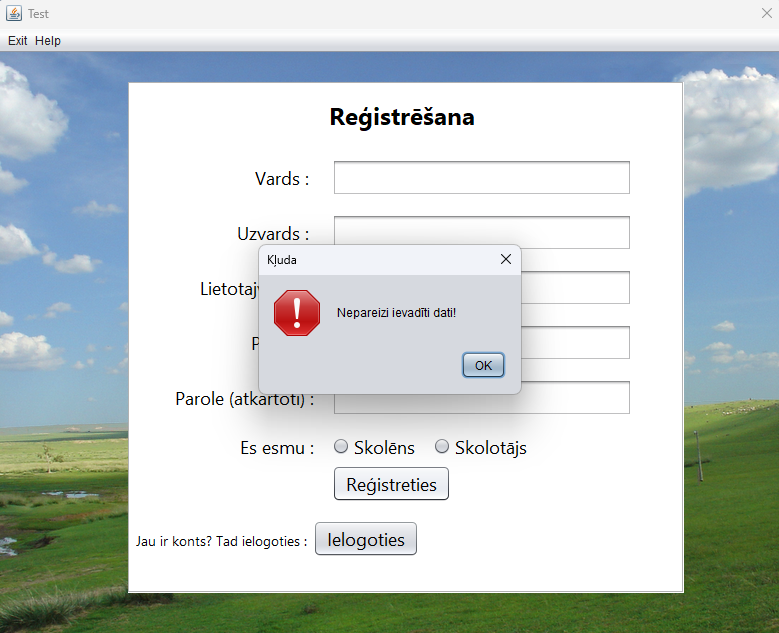


**3.6. att. Testēšanas sistēmas rezultatu forma**

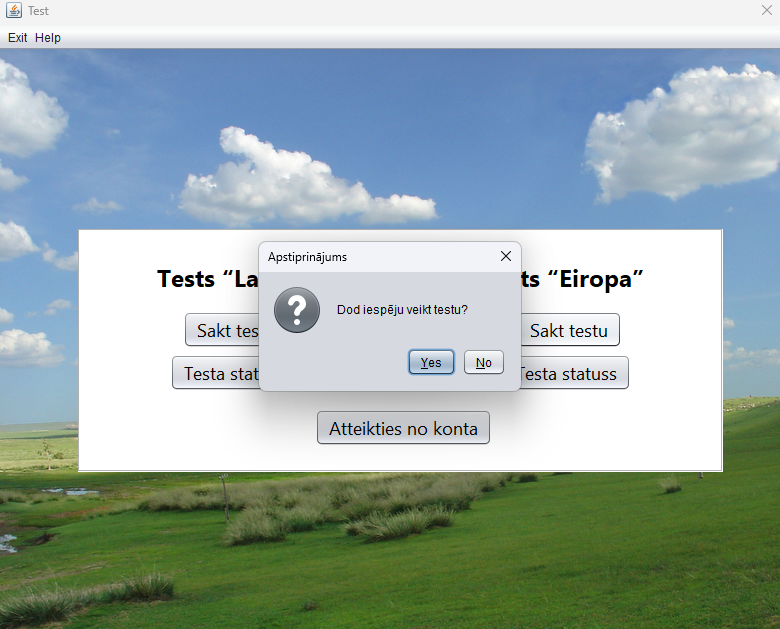
# Testēšanas rezultāti



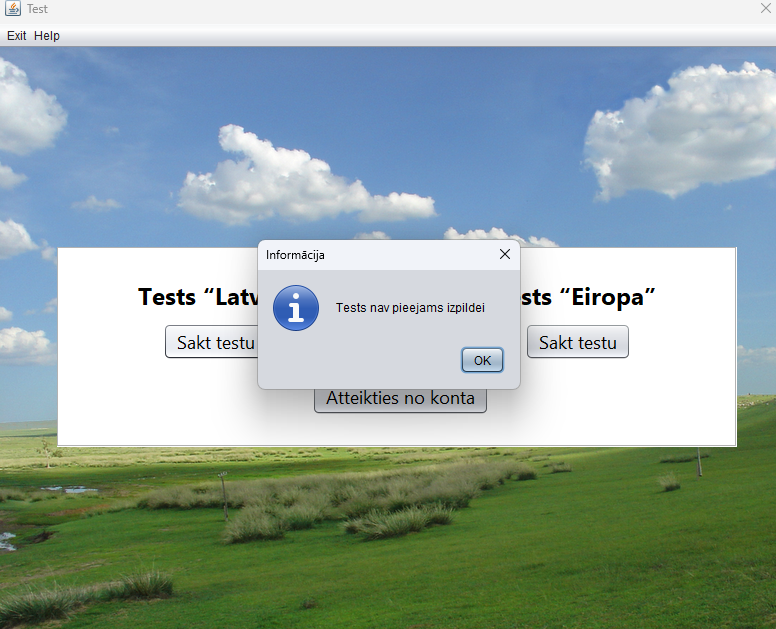
**3.7. att. Ielagošāna pārbaude**



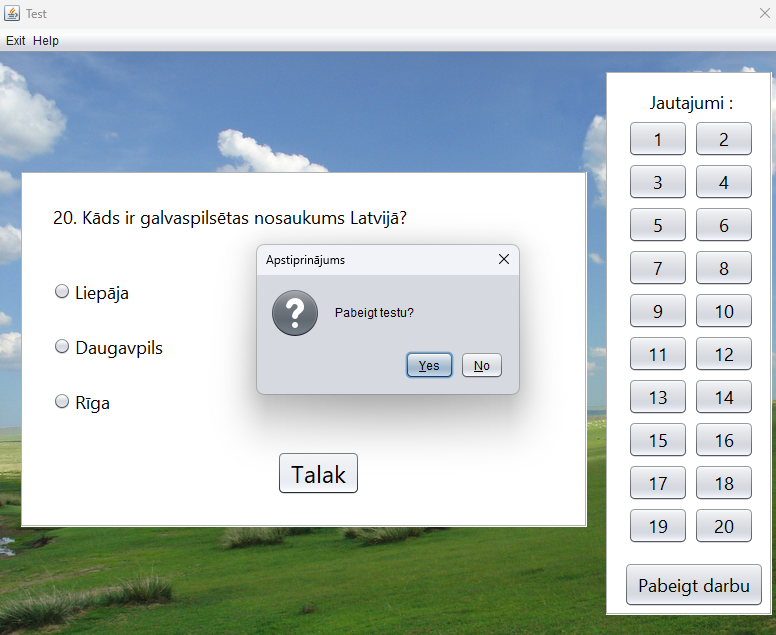
**3.8. att. Reģistrācijas** **pārbaude**



**3.9. att. Skolotāja apstiprinājuma** **pārbaude**



**3.10. att. Skolotāja apstiprinājuma** **pārbaude**



**3.10. att. Testa beigšana pārbaude**

# 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 «Pildīt testu» ļauj sākt pildīt testu.
      1. Pēc testa izpildīšanas tiek paradīts testa izpildes rezultāti.
   2. Poga «Atteikties no konta» ļauj iziet no sava konta.
4. Administratora izvēlnē ir 3 pogas.
   1. Poga «Sākt testu» ļauj atļaut lietotājam piekļuvi testam.
   2. Poga «Testa statuss» ļauj paradīt testa statusu – «Tests ir atļauts» vai «Tests nav atļauts».
   3. Poga «Atteikties no konta» ļauj iziet no administratora konta.

# Palīdzības sistēma (Help)

Lai sāktu pildīt testu, ir jāpiesakās savā kontā, izmantojot lietotājvārdu un paroli. Ja jums nav konta programmā, jums jāreģistrējas, norādot vārdu, uzvārdu, lietotājvārdu un paroli. Pēc tam varat pieteikties savā kontā. Pēc tam jums tiks dota iespēja izvēlēties testu un to izpildīt. Varat izvēlēties no 2 testiem: 1. “Latvija” un 2. “Eiropa”. Jūs izvēlaties vienu no testiem, un jūs varat sākt pildīt testu, kura augšpusē tiek parādīts jautājums, bet apakšā - 3 atbilžu varianti. Jūsu uzdevums ir izvēlēties vienu no atbilžu variantiem un nospiest pogu “Tālāk” vai izmantot izvēlne cilne un izvēlēties konkreto jautajums. Pabeidzot pildīt testu, jūs saņemsiet savu rezultātu, kas tiks saglabāts datu bāzē.

Ja rodas problēmas ar reģistrāciju vai ielagošānu, tad lietotājs var noklikšķināt uz pogas “Help”. Atvērsies logs, kurā būs aprakstīti reģistrācijas un ielagošānas noteikumi (4.1. att. un (4.2. att).



4.1. att. Ielagošāna noteikums



4.2. att. Reģistrācijas noteikums

# Secinājumi

Tika izstrādāts projekts «Testēšanas sistēma» NetBeans vidē. Tika izpildītas visas programmas bāzes prasības. Lai gan izstrādes gaitā radās dažādi sarežģījumi, kas neļāva pilnībā sasniegt sākotnēji iecerēto rezultātu, process bija vērtīgs – tika apgūtas jaunas zināšanas, kas būs noderīgas nākotnē. Tomēr svarīgākais ir tas, ka projekta pamatfunkcionalitāte ir ieviesta un darbojas korekti. Šāds liels un sarežģīts projekts tika īstenots pirmo reizi, taču tas bija ļoti interesants un lietderīgs.

# Izmantotās literatūras (informācijas avotu) saraksts

1. **2005-2023.** Latvijas Nacionālais terminoloģijas portāls. . *termini.gov.lv.* [Tiešsaiste] 2005-2023. gada.

[Citēts: 2025. gada 20. 2.] termini.gov.lv.

1. **Dortiņa, I. un Dortiņš, A. 2024.** 1.1.1. Ievads Java. Java pamati. [Tiešsaiste] 2024. gada 13. 2. [Citēts:

2025. gada 20. 2.] https://skolo.lv/mod/resource/view.php?id=82126301.

1. **—.** 1.1.2. Java sazarotas struktыras operвtori. [Tiešsaiste] 2024. gada 14. 02. [Citēts: 2025. gada 20.

02.] https://skolo.lv/mod/resource/view.php?id=82257905.

1. **—.** 1.1.3. Java cikliskās struktūras operatori. [Tiešsaiste] 2022. gada 22. 2. [Citēts: 2025. gada 20. 2.]

https://skolo.lv/mod/resource/view.php?id=82383944.

1. **—.** 1.2.1.1. Java masīvi. [Tiešsaiste] 2024. gada 29. 2. [Citēts: 2025. gada 20. 2.]

https://skolo.lv/mod/resource/view.php?id=82503734.

1. **—.** 1.2.1.2. Java rakstzīmju masīvi un rakstzīmju virknes. [Tiešsaiste] 2024. gada 7. 3. [Citēts: 2025.

gada 20. 2.] https://skolo.lv/mod/resource/view.php?id=82608403.

1. **—.** 1.2.1.3. Java dinamiskie masīvi. [Tiešsaiste] 2024. gada 13. 3. [Citēts: 2025. gada 20. 2.]

https://skolo.lv/mod/resource/view.php?id=82703677.

1. **—.** 1.2.2. Java ieraksti. [Tiešsaiste] 2024. gada 19. 3. [Citēts: 2025. gada 20. 2.]

https://skolo.lv/mod/resource/view.php?id=82785148.

1. **—.** 1.2.3. Java Ievades/izvades sistēma. [Tiešsaiste] 2024. gada 3. 4. [Citēts: 2025. gada 20. 2.]

https://skolo.lv/mod/resource/view.php?id=82944472&redirect=1.

1. **Dortiņa, I. un Stašanova, L. 2024.** 2.1.1.2. Java. Metodes un konstruktori. [Tiešsaiste] 2024. gada 2.

9. [Citēts: 2025. gada 20. 2.] https://skolo.lv/mod/resource/view.php?id=84426472.

1. **—.**  2.1.2. Java. Mantošana un metožu pārdefinēšana. [Tiešsaiste] 2024. gada 30. 9. [Citēts: 2025. gada

20. 2.] https://skolo.lv/mod/resource/view.php?id=85417326.

1. **—.**  2.1.3. Java pakotnes un interfeisi. [Tiešsaiste] 2024. gada 14. 10. [Citēts: 2025. gada 20. 2.]

https://skolo.lv/mod/resource/view.php?id=86185154.

1. **—.**  2.2. Java izņēmumi un to apstrāde. . [Tiešsaiste] 2022. gada 23. 10. [Citēts: 2025. gada 20. 2.]

https://skolo.lv/mod/resource/view.php?id=86272136&redirect=1.

1. **—.**  4.1.1. Java. Grafiskais lietotāja interfeiss. . [Tiešsaiste] 2025. gada 23. 1. [Citēts: 2025. gada 20. 2.]

https://skolo.lv/mod/resource/view.php?id=87282004.

1. **—.**  4.1.2. Java GUI izveidošana NetBeans vidē. [Tiešsaiste] 2025. gada 5. 2. [Citēts: 2025. gada 20.

2.] https://skolo.lv/mod/resource/view.php?id=87334441.

1. **—.**  Dortiņa, I. un Dortiņš, A. [Tiešsaiste] 2024. gada 16. 4. [Citēts: 2025. gada 20. 2.]

https://skolo.lv/mod/resource/view.php?id=83119822.

1. **Kodors, Sergejs. 2019.** *IEVADS PRASĪBU INŽENIERIJĀ. Mācību līdzeklis.* Rēzekne : Rēzeknes

Tehnoloģiju akadēmija, 2019. 978-9984-44-226-6.

1. **Taylor Johnson, Dung X. Nguyen. 2007.** UML and more JAVA Syntax. *COMP 212 LAB 1.5.*

[Tiešsaiste] 2007. gada 16. 1. [Citēts: 2025. gada 20. 2.] https://www.clear.rice.edu/comp212/07-spring/labs/01.5/.

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

# Pielikums A. Klašu sistēma

public class User{

private String name;

private String surname;

private String login;

private String password;

public User(String name, String surname, String login, String password) {

this.name = name;

this.login = login;

this.password = password;

}

public boolean enter(String login, String password) {

return this.login.equals(login) && this.password.equals(password);

}

}

public class Student extends User{

private ArrayList<Result> results = new ArrayList();

public Student(String name, String surname, String login, String password) {

super(name, surname, login, password);

}

public Result getResult(DistanceExamenator test) {

for (Result result : results) {

if (result.getTest() == test) {

return result;

}

}

return null;

}

public void saveResult(DistanceExamenator test, int questionCount, int rightAnswer) {

Result result = getResult(test);

if (result == null) {

results.add(new Result(test, questionCount, rightAnswer));

} else {

result.setResult(questionCount, rightAnswer);

}

}

}

public class Admin extends User{

public Admin(String name, String surname, String login, String password) {

super(name, surname, login, password);

}

public void startTest(DistanceExamenator test, boolean state) {

if (state) {

test.startTest();

} else {

test.endTest();

}

}

}

public class Question{

private String text;

private String[] answer;

private int correctAnswer;

public Question(String text, String[] answer, int correctAnswer) {

this.text = text;

this.answer = answer;

this.correctAnswer = correctAnswer;

}

public String getText() {

return text;

}

public String[] getAnswers() {

return answer;

}

public int getCorrectAnswer(){

return correctAnswer;

}

public String isCorrect(int correctAnswer){

return answer[correctAnswer];

}

}

public class Result {

private final DistanceExamenator test;

private int questionsCount = 0;

private int rightAnswer = 0;

private int percent = 0;

public Result(DistanceExamenator test, int questionsCount, int rightAnswer){

this.test = test;

this.questionsCount = questionsCount;

this.rightAnswer = rightAnswer;

}

public int getPercent(int rightAnswer , int questionsCount){

this.percent = (int) ((rightAnswer / (double) questionsCount) \* 100);

return percent;

}

public int getGradeFromPercent(int percent) {

if (percent >= 95) return 10;

else if (percent >= 85) return 9;

else if (percent >= 75) return 8;

else if (percent >= 65) return 7;

else if (percent >= 55) return 6;

else if (percent >= 45) return 5;

else if (percent >= 35) return 4;

else if (percent >= 25) return 3;

else if (percent >= 15) return 2;

else return 1;

}

public DistanceExamenator getTest(){

return test;

}

public int getQuestionsCount(){

return questionsCount;

}

public int getRightAnswers(){

return rightAnswer;

}

public void setResult(int questionsCount, int rightAnswers){

this.questionsCount = questionsCount;

this.rightAnswer = rightAnswers;

}

public void clear(){

questionsCount = 0;

rightAnswer = 0;

}

}

public class DistanceExamenator{

private List<Question> questions = new ArrayList<>();

private String name;

private boolean work = false;

public DistanceExamenator(boolean work, String name, ArrayList questions) {

this.work = work;

this.name = name;

this.questions = questions;

}

public List<Question> getQuestions() {

return questions;

}

public String getName(){

return name;

}

public void startTest(){

work = true;

}

public void endTest(){

work = false;

}

public boolean isWork() {

return work;

}

}

# Pielikums B. testa programmas kods

import org.junit.Test;

import static org.junit.Assert.\*;

import java.util.ArrayList;

import java.util.List;

/\*\*

\*

\* @author Roberts Erts

\*/

public class UserTest {

@Test

public void testUserEnter() {

User user = new User("Anna", "Liepa", "annie", "pass123");

assertTrue(user.enter("annie", "pass123"));

assertFalse(user.enter("annie", "wrongpass"));

}

}

public class StudentTest {

@Test

public void testSaveAndGetResult() {

DistanceExamenator test = new DistanceExamenator(false, "Test1", new ArrayList<>());

Student student = new Student("Roberts", "Erts", "roberterts", "qwerty123");

student.saveResult(test, 10, 8);

Result result = student.getResult(test);

assertNotNull(result);

assertEquals(8, result.getRightAnswers());

assertEquals(10, result.getQuestionsCount());

}

}

public class AdminTest {

@Test

public void testAdminStartTest() {

DistanceExamenator test = new DistanceExamenator(false, "Test1", new ArrayList<>());

Admin admin = new Admin("Jānis", "Prozers", "jan3", "123qwerty");

admin.startTest(test, true);

assertTrue(test.isWork());

admin.startTest(test, false);

assertFalse(test.isWork());

}

}

public class QuestionTest {

@Test

public void testQuestionFields() {

String[] answers = {"Jā", "Nē", "Nezinu"};

Question q = new Question("Vai tas darbojas?", answers, 0);

assertEquals("Vai tas darbojas?", q.getText());

assertArrayEquals(answers, q.getAnswers());

assertEquals(0, q.getCorrectAnswer());

assertEquals("Jā", q.isCorrect(0));

}

}

public class ResultTest {

@Test

public void testPercentAndGrade() {

DistanceExamenator test = new DistanceExamenator(false, "Test1", new ArrayList<>());

Result result = new Result(test, 20, 18);

int percent = result.getPercent(18, 20);

assertEquals(90, percent);

assertEquals(9, result.getGradeFromPercent(percent));

}

@Test

public void testClearAndSetResult() {

DistanceExamenator test = new DistanceExamenator(false, "Test1", new ArrayList<>());

Result result = new Result(test, 10, 5);

result.setResult(20, 15);

assertEquals(15, result.getRightAnswers());

assertEquals(20, result.getQuestionsCount());

result.clear();

assertEquals(0, result.getRightAnswers());

assertEquals(0, result.getQuestionsCount());

}

}

public class DistanceExamenatorTest {

@Test

public void testStartAndEndTest() {

DistanceExamenator test = new DistanceExamenator(false, "Test1", new ArrayList<>());

test.startTest();

assertTrue(test.isWork());

test.endTest();

assertFalse(test.isWork());

}

@Test

public void testGetNameAndQuestions() {

List<Question> questions = new ArrayList<>();

questions.add(new Question("Jautajuns?", new String[]{"1", "2", "3"}, 0));

DistanceExamenator test = new DistanceExamenator(true, "Test", new ArrayList<>(questions));

assertEquals("Test", test.getName());

assertEquals(1, test.getQuestions().size());

}

}