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.00XX.0XXPR**

Profesijas kods 33 481 031 Programmēšanas nodaļa

PR-21. grupa

|  |  |  |
| --- | --- | --- |
| Izveidoja | paraksts | J.,Artjomenko  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
* 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. Ka arī sistēma atvieglo darba pārbaudi, jo ta jau pārbauda un izvada rezultātus, kas palīdz skolotājam novertēt darbu un izlikt atzīmi.

# Uzdevuma nostādne

## Sistēmas modelis

### Priekšmetiskās jomas informācijas modelis

Apskatamā 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.).

Datu bāze

Tests

Rezultāti

Administrātors/

Skolotājs

Lietotājs

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

Saišu nozīme vienkāršotā nodaļas modelī:

* Skolotājs ir testa veidotājs un pārbaudītajs, kuram ir pieeja pie atbildēm, testam un lietotāja rezultātiem;
* Datu Bāze satur failus ar jautājumiem un pareizām atbildēm;
* Lietotājs ir testa pildītājs, kas beigās var apskatīt savus rezultātus, nepieciešamības gadījumā viņš varēs vērsties pie skolotāja;
* Tests ir instruments, kas palīdzēs pārbaudīt lietotāja zināšanas un novertēt to ar atzīmi;
* Rezultātā 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.).

Lietotājs

Lietotāja

saskarne

Lietotāja dati un

rezultāti

Datu bāze

Sistēma

* 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.

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.](#_bookmark9) att.)

Testēšanas

Sistēma(V0)

Lietotājs(V11)

Datu bāze(V12)

Lietotāju reģistrs(V121)

Skolotājs

(V111)

Audzēknis

(V112)

Testa jautājumu reģistrs(V122)

Audzēkņa funkcijas(V132)

Skolotāja funkcijas(V131)

Testa izveide

(V1311)

Testa pildīšana

(V1321)

Testa manuāla novērtēšana

(V1312)

Testa rezultātu saņemšana

(V1322)

Tetst (V13)

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

### Datu modelēšāna

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.

Audzēkņi

Pilda

Tests

Saturs

Skolotājs

Izveido

Jautājums

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

### Datu vārdnīca

|  |  |  |
| --- | --- | --- |
| Datu tips | Nosaukums | Apraksts |
| String | **name** | Audzēknis ievada savu vārdu un uzvārdu |
| String | **questionText** | Neliels teksta apgabals, kas apraksta jautājumu |
| String[] | **answersText** | Atbilžu variantu masīvs. Tiek piedāvāti vairāki atbilžu varianti. |
| int | **correctAnswer** | Aile, kas nosaka, vai tika izvēlēts pareizs atbildes variants |
| float | **result** | Audzēkņa saņemtais punktu skaits (0-100) |
| int | **mark** | Audzēkņa atzīme (0-10) |

## Sistēmas evolūcija

Testēšanas sistēma ir neatkarīga un pašpietiekama programma. Produktam nav paradzētas

vairākas sastāvdaļas.

Programmai ir trīs ārējās saskarnes, kuras var apskatīt 1.4. nodaļā – viena paredzēta reģistrēšanas

sistēmā, otrā – ielagošanai sistēmā un treša testa pildīšanai. Grafisko lietotāja saskarni var pilnveidot, papildinot to ar dažādām krāsu un noformējuma shēmām.

Testēšanas sistēmā paredzēti divu lietotāju veidi: administratorsun 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 viņu lomas.

Programma satur vienu testu, kurā ir 10 jautajumi, tomēr nākotnē sistēmu var pilnveidot,

pievienojot jaunas testa tēmas, kļūdu uzrādīšanuun kļūdu skaidrojumu pēc testa izpildes, lietotāju reitingu, lietotāju datu izvadi lietotāja izvēlnē vai admistratora 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

1. **Funkcija „Reģistrēties sistēmā”**

***ID: T1***

**Ievads:** Ļauj ievadīt nepieciešāmos datus**,** lai izveidotu savu kontu. Visi ievadīti dati tiek saglabāti failā

**Ievade:**

1. Lietotāja vārds
2. Lietotāja uzvārds
3. Lietotāja lietotājvārds
4. Lietotāja parole
5. Ievadītas paroles apstiprināšāna

**Apstrāde:**

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

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

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

1. **Funkcija „Ielogoties sistēmā”**

***ID: T2***

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

**Ievade:**

1. Lietotāja Lietotājvārds
2. Lietotāja parole

**Apstrāde:**

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

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

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

1. **Funkcija „Sākt tetsu”**

***ID:T3***

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

**Ievade:** Lietotājs noklikšķina uz pogu „Sākt tetsu”.

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

1. **Funkcija „Atbildēt uz tetsta jautājumu”**

***ID:T4***

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

**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 "Turpināt", pēc kura tiek parādīts nākamā izvēlne ar testa jautājumiem.

1. **Funkcija „Beigt tetsu”**

***ID:T5***

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

**Ievade:** Lietotājs noklikšķina uz pogas "Beigt testu".

**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 jautājumu skaits.
2. Pareizo atbilžu skaits.
3. Pareizo atbilžu procentuālā daļa.
4. Atzīme.
5. Sistēmas komentārs par saņemto atzīmi.
6. **Funkcija „Palīdzība”**

***ID:T6***

## Ievads: Palīdz lietotājiem saprast kā izmantot programmu un strādāt tālāk ar viņu.

## Ievade: Lietotājs nospiež uz pogu „Palīdzība” augšājā labajā stūrī.

**Izvade:** Teik atvērts logs kur ir rakstīts kā , ko un kur vajag ievadīt nepiecišamos datus, vai/ka arī kuru pogu spiest, lai turpinātu darbību ar progrmmu.

1. **Funkcija „Mēģināt vēlreiz”**

***ID:T7***

**Ievads:** Ļauj lietotājiem veikt testu vēlreiz.

**Ievade:** Lietotājs nospiež pogu „**Mēģināt vēlreiz**” apakšā labajā stūrī.

**Izvade:** Tests būs pieejams atkārtotai pildīšānai, paradīsies logs „Sākt testu”.

* 1. **Funkcija „Beigt”**

***ID:T8***

**Ievads:** Ļauj lietotājam iziet no programmas.

**Ievade:** Lietotājs nospiež pogu „Beigt”.

**Izvade:** Programma aizvēras.

## 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: reģistrēšānai sistēmā, ieloģošanai 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 it 10’’ ar minimālo izšķirtspēju –800x600 punkti.
   3. Lietotāja saskarne „Lietotāja reģistrēšanas saskarne”. Saskarnes struktūra: skat. 1.6. att.

**1.6. att. Lietotāja reģistrēšanas saskarne**

Vārds

Uzvārds

Lietotājvārds

Parole

Parole (atkārtoti)

Reģistrēties

Ielogoties

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

Lietotājvārds:

Parole:

Reģistrēties

Ielogoties

1. **att. Lietotāja ielogošanas saskarne**

**Saskarnes elementi:**

1. Teksta lauks „Lietotājvards”
2. Teksta lauks „Parole”
3. Poga „Ielogoties”
4. Poga „Reģistrēties”
   1. Lietotāja saskarne “Testa izpilde”. Saskarnes struktūra: skat. [1.8. att.](#_bookmark18)

a)

Tests „Mans Tests”

Sākt testu

b)

1. Jautājums „Jautājuma Teksts” 10

1. atbilde

2. atbilde

3. atbilde

Tālāk

c)

Rezultāti

Procenti: X%

Atzīme: A

Beigt

Mēģināt vēlreiz

Neatbildēto jautājumu skaits:

Sistēmas ziņojums:

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

**Saskarnes elementi:**

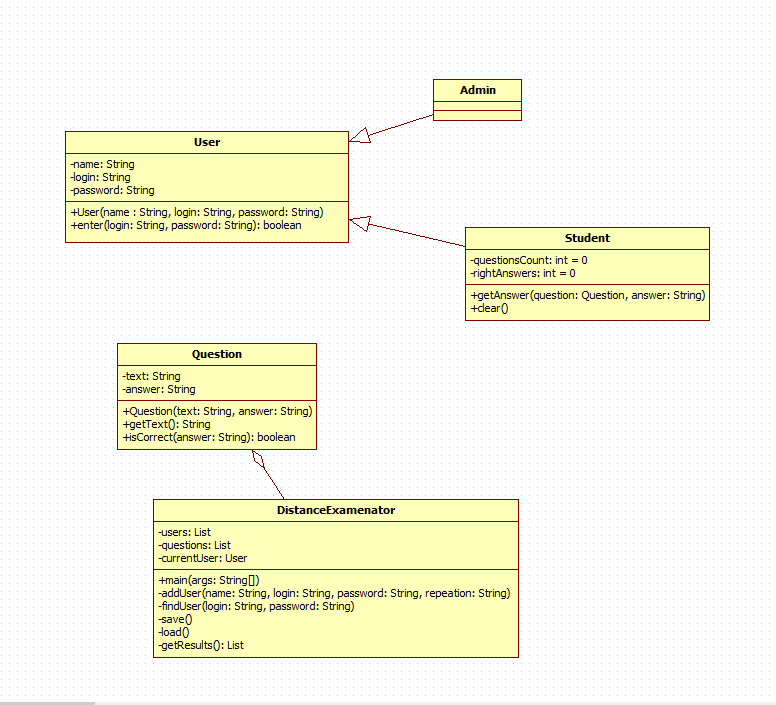
1. a)
   1. Poga „Sākt testu”
2. b)
   1. Atbildes logs „1. jautājums”
   2. Atbildes logs „2. jautājums”
   3. Atbildes logs „3. jautājums”
   4. Poga „Tālāk”
3. c)
   1. Poga „Beigt”
   2. Poga „Mēģināt vēlreiz”

# Priekšmetiskās jomas klašu sistēmas izstrādē

## 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.



* 1. **att. Testēšanas sistēmas klašu diagramma UML valodā**

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

* 1. Klase **User** – ir atbildīga par darbu ar lietotājiem. Tā satur pamata informāciju par lietotāju, piemēram, vārdu, lietotājvārdu un paroli.
* 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:

1. Klase **Admin** – nodrošina darbu ar lietotajiem un testiem, Ta paplašina klasi **User** un ir paredzēta sistēmas administrātoriem.
2. Klase **Student** – Ta arī paplašina klasi **User** un nodrošina audzēkņa testa pildīšanas darbības.

* Satur mainīgos: questionsCount (kopējais jautājumu skaits) un rightAnswers (pareizo atbilžu skaits);
* getAnswer(question: Question, answer: String) – metode kas apstrādā audzēkņa atbildi un nosaka tas pareizību;
* clear() – metode kas attīra testa rezultātus;

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

* Question(text: String, answer: String) – konstruktors kas inicializē jautājuma tekstu un pareizo atbildi;
* getText(): String – atgriež jautājuma tekstu;
* isCorrect(answer: String): boolean – pārbauda vai ievadītā atbilde ir pareiza;

1. Klase **DistanceExamenator** – programmas darbību koordinējošā klase. Tā nodrošina programmas galvenās metodes izsaukumu un grafiskās saskarnes izveidi. Tā satur:

* Lietotāju **User** un jautājumu **questions** sarakastus, ka arī pašreizējo litotāju **curentUser.**
* main(args: String[]) – programmas sākumpukts;
* addUser(name: String, login: String, password: String, repeation: String) – metode, kas pievieno jaunu lietotāju;
* findUser(login: String, password: String) – meklē lietotāju pēc pieslēgšanās datiem;
* save() un load() – metodes datu saglabāšanai un ielādei;
* getResults(): List – atgriež rezultātu sarakstu, iespējams, testēšanas statistiku

## Klašu realizēšāna Java valodā

Izstrādātās klašu diagrammas (2.1. att.) realizēšanai NetBeans vidē tika izveidots projekts ar

nosaukumu TestingSystem, kurā izstrādātās klases tika aprakstītas Java valodā. Šeit tiek piedāvāts klašu vispārīgā apraksta (klašu deklarācijas) programmas kods un metožu apraksti.

Pilns avota programmas kods ar klašu metožu definīcijām ir ievietots pielikumā “Pielikums A.

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

class User{

private String name;

private String login;

private String password;

private void User(String name, String login, String password){

}

private boolean enter(String login, String password){

return true;

}

}

class Student{

private int questionsCount = 0;

private int rightAnswers = 0;

private void getAnswer(String question, String answer){

}

}

class Question{

private String text;

private String answer;

private void Question (String text, String answer){

}

private void getText (){

}

}

class DistanceExamenator{

private String users;

private String questions;

private String currentUser;

private void addUser(String name, String login, String password, String repeation){

}

private void findUser(String login, String password){

}

private void save(){

}

private void load(){

}

private void getResults(){

}

}

# Testa programmas izstrāde

## Testēšanas metodikas

Pirms priekšmetiskās jomas klašu sistēmas testēšanas programmas izstrādes, ir jānosaka

Testēšānas metodika.

Ir jaātšķir 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. Testa piemēri 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. ). Ja lietotāja 15 dati sakrīt ar validācijas datiem, tad lietotājam tiek atvērts programmas galvenais logs
* Ielogošanas procesā ir iespējamas vairākas kļūdas – neaizpildīti lauki, neprecīzi ievadīti lietotāja dati.



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

1. Reģistrācijas procesa apraksts:

* Reģistrācjas procces ir paredzēts lietotāju pievienošanai sistēmā. Lai lietotājs varētu izmanot programmu un piekļūt testam, vispirms vajag reģistrēties. Reģistrācijas procesā lietotājs ievada savu vārdu, uzvārdu un izdomā lietotājvārdu un paroli, ka arī paroli ir jāievada otro reizi apstiprināšanai. Ja parole atbilst prasībām un parole apstiprināšanai sakrīt ar izdomāto paroli, tad lietotājam atvērsies logs „Ielogošanās sistēmā”. Reģistrācijas procesā ir iespējamas dažas kļūdas,

piemēram, neaizpildīti lauki vai paroles nesakrīt. 

**3.2. att. Testēšanas sistēmas reģistrācijas forma**

# Lietotāja rokasgrāmata

## Lietotāja instrukcija

1. Lietotājs palaiž programmu.
2. Parādās logs „Reģistrācija sistēmā”, lietotājam ir jāizveido savs konts.
   1. Reģistrācijas logā, lietotājas aizpilda visus nepieciešamos datus un nospied pogu „Reģistrēties”.
3. Pēc reģistrācijas lietotājam parādās logs „Ielogošanas sistēmā”, lietotājam ir jāieiet savā kontā.
   1. Ielogošānas logā lietotājs aizpilda visus nepieciešamos datus un nospied pogu „Ielogoties”.
4. Pēc ielogošanas parādās logs „Pildīt testu”, lietotājs nospiež pogu „Sākt testu” un sāk pildīt testu.
   1. Poga „Sākt testu” ļauj sākt pildīt testu;
   2. Poga „Beigt testu” testa beigās ļauj beigt testu un apskatīt savu rezultātus
   3. Poga „Beigt” ļauj iziet no programas.
   4. Poga „Mēģināt vēlreiz ” ļauj pildīt testu atkārtoti.

## Palīdzības sistēma (Help)

Lai sāktu strādāt ar programmu, lietotājam vispirms ir jāizveido savs konts. Rēģistrācijas logā

Lietotājam ir jāievada nepieciešamos datus (vārds, uzvārds, lietotājvārds, parole, un parole apstiprināšanai) un jānospiež pogu „Reģistrēties”.

Tālāk parādīsies logs ar Ielogošanos sistēmā. Lietotājam vajag ievadīt nepiecišamos datus

(lietotājvārds un parole), lai ielogoties sistēmā, tālāk lietotājam vajag nospiezt pogu „Ielogoties” un pēc tam parādīsies logs ar testu un lietotājam jānospiež poga „Sākt testu”.

Pēc pogas „Sākt testu” nospiešanas, parādās logs ar pirmo jajutājumu un katrā jautājuma logā ir

poga „Tālāk”, kura nosūta uz nākamo jautājumu. Lietotājam vajag atbildēt uz jautajumu un jānospied pogu „Tālāk”, lai turpinātu atbildēt uz jautājumiem.

Pedējā jautājumā ir poga „Beigt testu”, lietotājam ir jānospied to un tad parādas logs ar

rezultātiem, kur izvadās atzīme un procenti par izpildīto darbu. Šajā logā ir divas pogas „Beigt” un „Mēģināt vēlreiz”. Poga „Beigt” ļauj iziet no programmas, un poga „Mēģināt vēlreiz” ļauj pildīt testu atkārtoti, lietotājs nospiež šo pogu un viņu pārsūta uz Ielogošanas logu, lai lietotājs ielogotos un izpildītu testu vēlreiz.

# Secinājumi

Tika izstrādāts projekts «Testēšanas sistēma» NetBeans vidē. Tika izpildītas visas programmas

bāzes prasības. Projekta izstrādes laika radās daudz neskaidrību, piemēram, ar testa veidu, vai cik daudz testu ir jābūt, un arī cik daudz jautājumu jābūt. Bet protams projekta izstrādēj palīdzēja iepriekšējie darbi, kur mēs veidojām testu, tas atviegloja darba izpildi.

# Pielikums A. Klašu sistēma

class User{

private String name;

private String login;

private String password;

private void User(String name, String login, String password){

}

private boolean enter(String login, String password){

return true;

}

}

class Student{

private int questionsCount = 0;

private int rightAnswers = 0;

private void getAnswer(String question, String answer){

}

}

class Question{

private String text;

private String answer;

private void Question (String text, String answer){

}

private void getText (){

}

}

class DistanceExamenator{

private String users;

private String questions;

private String currentUser;

private void addUser(String name, String login, String password, String repeation){

}

private void findUser(String login, String password){

}

private void save(){

}

private void load(){

}

private void getResults(){

}

}

# Pielikums B. testa programmas kods

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

String enteredNickname = jTextField6.getText();

String enteredPassword = new String(jPasswordField3.getPassword());

boolean loginSuccess = false;

try (BufferedReader reader = new BufferedReader(new FileReader("dataArtjomenko.txt"))) {

String line;

String nickname = null;

String password = null;

while ((line = reader.readLine()) != null) {

if (line.startsWith("Lietotājvārds: ")) {

nickname = line.substring("Lietotājvārds: ".length()).trim();

} else if (line.startsWith("Parole: ")) {

password = line.substring("Parole: ".length()).trim();

// Pārbaudām, kad ir gan lietotājvārds, gan parole

if (enteredNickname.equals(nickname) && enteredPassword.equals(password)) {

loginSuccess = true;

break;

}

}

}

if (loginSuccess) {

JOptionPane.showMessageDialog(this, "Veiksmīga ielagošanās!", "Piekļuve atļauta", JOptionPane.INFORMATION\_MESSAGE);

// Atveram logu tikai tad, ja ielogošanās ir veiksmīga

TestIntroducion.setBounds(0, 0, 800, 600);

TestIntroducion.setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

TestIntroducion.setVisible(true);

TestIntroducion.setLocationRelativeTo(null); // Centrē logu ekrānā

LoginWindow.dispose();

jTextField6.setText("");

jPasswordField3.setText("");

} else {

JOptionPane.showMessageDialog(this, "Nepareizs lietotājvārds vai parole!", "Kļūda", JOptionPane.ERROR\_MESSAGE);

}

} catch (IOException e) {

JOptionPane.showMessageDialog(this, "Kļūda lasot failu!", "Kļūda", JOptionPane.ERROR\_MESSAGE);

e.printStackTrace();

}

}

private void HelpButton2MouseClicked(java.awt.event.MouseEvent evt) {

HelpFunctionWindow2.setBounds(0, 0, 400, 300);

HelpFunctionWindow2.setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

HelpFunctionWindow2.setLocationRelativeTo(null);

HelpFunctionWindow2.setVisible(true);

}

private void Quit1MouseClicked(java.awt.event.MouseEvent evt) {

HelpFunctionWindow1.dispose();

}

private void Quit2MouseClicked(java.awt.event.MouseEvent evt) {

HelpFunctionWindow2.dispose();

}

private void Quit3MouseClicked(java.awt.event.MouseEvent evt) {

HelpFunctionWindow3.dispose();

}

private void RegistrationButton1MouseClicked(java.awt.event.MouseEvent evt) {

this.setBounds(0, 0, 800, 600);

this.setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

this.setLocationRelativeTo(null); // Centrē logu ekrānā

this.setVisible(true);

LoginWindow.dispose();

jTextField6.setText("");

jPasswordField3.setText("");

jTextField1.setText("");

jTextField2.setText("");

jTextField3.setText("");

jPasswordField1.setText("");

jPasswordField2.setText("");

}

private void jButton1MouseClicked(java.awt.event.MouseEvent evt) {

HelpFunctionWindow3.setBounds(0, 0, 400, 300);

HelpFunctionWindow3.setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

HelpFunctionWindow3.setLocationRelativeTo(null);

HelpFunctionWindow3.setVisible(true);

}

private void jButton2MouseClicked(java.awt.event.MouseEvent evt) {

FirstQuestion.setBounds(0, 0, 800, 600);

FirstQuestion.setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

FirstQuestion.setLocationRelativeTo(null);

FirstQuestion.setVisible(true);

TestIntroducion.dispose();

}

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

String name = jTextField1.getText();

String lastname = jTextField2.getText();

String nickname = jTextField3.getText();

String password = new String(jPasswordField1.getPassword());

String confirmPassword = new String(jPasswordField2.getPassword()); // otrais paroles lauks

if (name.isEmpty() || lastname.isEmpty() || nickname.isEmpty() || password.isEmpty() || confirmPassword.isEmpty()) {

JOptionPane.showMessageDialog(this, "Lūdzu, aizpildiet visus laukus, ieskaitot abas paroles!", "Kļūda", JOptionPane.ERROR\_MESSAGE);

return;

}

// Pārbaude: vai parole satur gan burtus, gan ciparus

if (!password.matches("^(?=.\*[a-zA-Z])(?=.\*\\d).+$")) {

JOptionPane.showMessageDialog(this, "Parolei jāietver gan burti, gan cipari!", "Nepietiekama parole", JOptionPane.WARNING\_MESSAGE);

return;

}

// Pārbaude: vai paroles sakrīt

if (!password.equals(confirmPassword)) {

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

return;

}

// Saglabājam atbildes failā

try (FileWriter writer = new FileWriter("dataArtjomenko.txt", true)) {

writer.write("Vārds: " + name + "\n");

writer.write("Uzvārds: " + lastname + "\n");

writer.write("Lietotājvārds: " + nickname + "\n");

writer.write("Parole: " + password + "\n");

JOptionPane.showMessageDialog(this, "Reģistrēšana ir veiksmīgi pabeigta!");

} catch (IOException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, "Kļūda saglabājot failu!", "Kļūda", JOptionPane.ERROR\_MESSAGE);

}

// Atver logu tikai, ja viss kārtībā

LoginWindow.setBounds(0, 0, 800, 600);

LoginWindow.setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

LoginWindow.setLocationRelativeTo(null); // Centrē logu ekrānā

LoginWindow.setVisible(true);

this.dispose();

jTextField1.setText("");

jTextField2.setText("");

jTextField3.setText("");

jPasswordField1.setText("");

jPasswordField2.setText("");

}

private void LoginWindow1MouseClicked(java.awt.event.MouseEvent evt) {

LoginWindow.setBounds(0, 0, 800, 600);

LoginWindow.setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

LoginWindow.setLocationRelativeTo(null); // Centrē logu ekrānā

LoginWindow.setVisible(true);

this.dispose();

jTextField1.setText("");

jTextField2.setText("");

jTextField3.setText("");

jPasswordField1.setText("");

jPasswordField2.setText("");

}

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

HelpFunctionWindow1.setBounds(0, 0, 450, 350);

HelpFunctionWindow1.setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

HelpFunctionWindow1.setLocationRelativeTo(null); // Centrē logu ekrānā

HelpFunctionWindow1.setVisible(true);

}

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

SecondQuestion.setBounds(0, 0, 800, 610);

SecondQuestion.setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

SecondQuestion.setLocationRelativeTo(null); // Centrē logu ekrānā

SecondQuestion.setVisible(true);

FirstQuestion.dispose();

}

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

ThirdQuestion.setBounds(0, 0, 800, 600);

ThirdQuestion.setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

ThirdQuestion.setLocationRelativeTo(null); // Centrē logu ekrānā

ThirdQuestion.setVisible(true);

SecondQuestion.dispose();

}

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

FourthQuestion.setBounds(0, 0, 800, 600);

FourthQuestion.setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

FourthQuestion.setLocationRelativeTo(null); // Centrē logu ekrānā

FourthQuestion.setVisible(true);

ThirdQuestion.dispose();

}

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

FifthQuestion.setBounds(0, 0, 800, 600);

FifthQuestion.setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

FifthQuestion.setLocationRelativeTo(null); // Centrē logu ekrānā

FifthQuestion.setVisible(true);

FourthQuestion.dispose();

}

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

SixthQuestion.setBounds(0, 0, 800, 600);

SixthQuestion.setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

SixthQuestion.setLocationRelativeTo(null); // Centrē logu ekrānā

SixthQuestion.setVisible(true);

FifthQuestion.dispose();

}

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

SeventhQuestion.setBounds(0, 0, 800, 600);

SeventhQuestion.setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

SeventhQuestion.setLocationRelativeTo(null); // Centrē logu ekrānā

SeventhQuestion.setVisible(true);

SixthQuestion.dispose();

}

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

EighthQuestion.setBounds(0, 0, 800, 600);

EighthQuestion.setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

EighthQuestion.setLocationRelativeTo(null); // Centrē logu ekrānā

EighthQuestion.setVisible(true);

SeventhQuestion.dispose();

}

private void jButton3MouseClicked(java.awt.event.MouseEvent evt) {

NinethQuestion.setBounds(0, 0, 800, 600);

NinethQuestion.setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

NinethQuestion.setLocationRelativeTo(null); // Centrē logu ekrānā

NinethQuestion.setVisible(true);

EighthQuestion.dispose();

}

private void jButton4MouseClicked(java.awt.event.MouseEvent evt) {

TenthQuestion.setBounds(0, 0, 800, 600);

TenthQuestion.setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

TenthQuestion.setLocationRelativeTo(null); // Centrē logu ekrānā

TenthQuestion.setVisible(true);

NinethQuestion.dispose();

}

private void jButton5MouseClicked(java.awt.event.MouseEvent evt) {

EleventhQuestion.setBounds(0, 0, 800, 600);

EleventhQuestion.setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

EleventhQuestion.setLocationRelativeTo(null); // Centrē logu ekrānā

EleventhQuestion.setVisible(true);

TenthQuestion.dispose();

}

private void jButton6MouseClicked(java.awt.event.MouseEvent evt) {

TwelvethQuestion.setBounds(0, 0, 800, 600);

TwelvethQuestion.setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

TwelvethQuestion.setLocationRelativeTo(null); // Centrē logu ekrānā

TwelvethQuestion.setVisible(true);

EleventhQuestion.dispose();

}

private void jButton7MouseClicked(java.awt.event.MouseEvent evt) {

ThirteenthQuestion.setBounds(0, 0, 800, 600);

ThirteenthQuestion.setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

ThirteenthQuestion.setLocationRelativeTo(null); // Centrē logu ekrānā

ThirteenthQuestion.setVisible(true);

TwelvethQuestion.dispose();

}

private void jButton8MouseClicked(java.awt.event.MouseEvent evt) {

FourteenthQuestion.setBounds(0, 0, 800, 600);

FourteenthQuestion.setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

FourteenthQuestion.setLocationRelativeTo(null); // Centrē logu ekrānā

FourteenthQuestion.setVisible(true);

ThirteenthQuestion.dispose();

}

private void jButton9MouseClicked(java.awt.event.MouseEvent evt) {

FifteenthQuestion.setBounds(0, 0, 800, 600);

FifteenthQuestion.setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

FifteenthQuestion.setLocationRelativeTo(null); // Centrē logu ekrānā

FifteenthQuestion.setVisible(true);

FourteenthQuestion.dispose();

}

private void jButton10MouseClicked(java.awt.event.MouseEvent evt) {

TestEnd.setBounds(0, 0, 800, 600);

TestEnd.setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

TestEnd.setLocationRelativeTo(null); // Centrē logu ekrānā

TestEnd.setVisible(true);

FifteenthQuestion.dispose();

try {

int correct = 0;

String[] answers = new String[15];

// Jautājums 1

if (jRadioButton2.isSelected()) { correct++; answers[0] = "Cat"; }

else if (jRadioButton1.isSelected()) answers[0] = "Dog";

else if (jRadioButton3.isSelected()) answers[0] = "Cow";

else if (jRadioButton4.isSelected()) answers[0] = "Parrot";

// Jautājums 2

String a2 = jTextField4.getText().trim();

answers[1] = a2;

if (a2.equalsIgnoreCase("Dog")) correct++;

// Jautājums 3

String a3 = (String) jComboBox1.getSelectedItem();

answers[2] = a3;

if (a3.equalsIgnoreCase("Cow")) correct++;

// Jautājums 4

if (jRadioButton8.isSelected()) { correct++; answers[3] = "Parrot"; }

else if (jRadioButton5.isSelected()) answers[3] = "Cat";

else if (jRadioButton6.isSelected()) answers[3] = "Dog";

else if (jRadioButton7.isSelected()) answers[3] = "Cow";

// Jautājums 5

String a5 = jTextField5.getText().trim();

answers[4] = a5;

if (a5.equalsIgnoreCase("Horse")) correct++;

// Jautājums 6

String a6 = (String) jComboBox2.getSelectedItem();

answers[5] = a6;

if (a6.equalsIgnoreCase("Sheep")) correct++;

// Jautājums 7

if (jRadioButton10.isSelected()) { correct++; answers[6] = "Rabbit"; }

else if (jRadioButton9.isSelected()) answers[6] = "Rat";

else if (jRadioButton11.isSelected()) answers[6] = "Mouse";

else if (jRadioButton12.isSelected()) answers[6] = "Cat";

// Jautājums 8

String a8 = jTextField7.getText().trim();

answers[7] = a8;

if (a8.equalsIgnoreCase("Rat")) correct++;

// Jautājums 9

String a9 = (String) jComboBox3.getSelectedItem();

answers[8] = a9;

if (a9.equalsIgnoreCase("Fish")) correct++;

// Jautājums 10

if (jRadioButton16.isSelected()) { correct++; answers[9] = "Hamster"; }

else if (jRadioButton13.isSelected()) answers[9] = "Pig";

else if (jRadioButton14.isSelected()) answers[9] = "Chicken";

else if (jRadioButton15.isSelected()) answers[9] = "Duck";

// Jautājums 11

String a11 = jTextField8.getText().trim();

answers[10] = a11;

if (a11.equalsIgnoreCase("Pig")) correct++;

// Jautājums 12

String a12 = (String) jComboBox4.getSelectedItem();

answers[11] = a12;

if (a12.equalsIgnoreCase("Bear")) correct++;

// Jautājums 13

if (jRadioButton17.isSelected()) { correct++; answers[12] = "Chicken"; }

else if (jRadioButton18.isSelected()) answers[12] = "Deer";

else if (jRadioButton19.isSelected()) answers[12] = "Elk";

else if (jRadioButton20.isSelected()) answers[12] = "Owl";

// Jautājums 14

String a14 = jTextField9.getText().trim();

answers[13] = a14;

if (a14.equalsIgnoreCase("Owl")) correct++;

// Jautājums 15

String a15 = (String) jComboBox5.getSelectedItem();

answers[14] = a15;

if (a15.equalsIgnoreCase("Bee")) correct++;

// Saglabāt atbildes failā

try (BufferedWriter bw = new BufferedWriter(new FileWriter("answersArtjomenko.txt"))) {

for (int i = 0; i < answers.length; i++) {

bw.write((i + 1) + ". jautājums: " + answers[i]);

bw.newLine();

}

}

int percentage = (int) Math.round(correct \* 100.0 / 15);

String grade;

String message;

if (percentage >= 90) {

grade = "10";

message = "Ļoti labi paveikts darbs";

} else if (percentage >= 80) {

grade = "9";

message = "Ļoti labi paveikts darbs";

} else if (percentage >= 70) {

grade = "8";

message = "Labi, bet vajag pacensties";

} else if (percentage >= 60) {

grade = "7";

message = "Labi, bet vajag pacensties";

} else if (percentage >= 50) {

grade = "6";

message = "Vajag atkārtot tēmu";

} else if (percentage >= 40) {

grade = "5";

message = "Vajag atkārtot tēmu";

} else if (percentage >= 20) {

grade = "3–4";

message = "Slikti";

} else if (percentage >= 1) {

grade = "1–2";

message = "Ļoti slikti";

} else {

grade = "0";

message = "Nepietiekami";

}

// Aprēķināt neatbildēto jautājumu skaitu

int unanswered = 15 - correct;

try (BufferedWriter bw = new BufferedWriter(new FileWriter("resultsArtjomenko.txt"))) {

String name = jTextField1.getText().trim();

String lastname = jTextField2.getText().trim();

bw.write("Vārds: " + name);

bw.newLine();

bw.write("Uzvārds: " + lastname);

bw.newLine();

bw.write("Pareizās atbildes: " + correct + " no 15");

bw.newLine();

bw.write("Procentuāli: " + percentage + "%");

bw.newLine();

bw.write("Atzīme: " + grade);

bw.newLine();

bw.write("Vērtējums: " + message);

}

// Izvade TextFieldos

jTextField10.setText(grade); // atzīme

jTextField11.setText(percentage + "%"); // procenti

jTextField12.setText(String.valueOf(unanswered)); // neatbildētie

jTextField13.setText(message); // komentārs par darbu

} catch (Exception e) {

JOptionPane.showMessageDialog(this, "Kļūda pārbaudot atbildes!", "Kļūda", JOptionPane.ERROR\_MESSAGE);

e.printStackTrace();

}

}

private void jButton11MouseClicked(java.awt.event.MouseEvent evt) {

LoginWindow.setBounds(0, 0, 800, 600);

LoginWindow.setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

LoginWindow.setLocationRelativeTo(null); // Centrē logu ekrānā

LoginWindow.setVisible(true);

jTextField6.setText("");

jPasswordField3.setText("");

TestEnd.dispose();

}

private void jButton12MouseClicked(java.awt.event.MouseEvent evt) {

System.exit(0);

}

private void jButton13MouseClicked(java.awt.event.MouseEvent evt) {

FirstQuestion1.setBounds(0, 0, 800, 600);

FirstQuestion1.setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

FirstQuestion1.setLocationRelativeTo(null);

FirstQuestion1.setVisible(true);

TestIntroducion.dispose();

}

private void jButton14MouseClicked(java.awt.event.MouseEvent evt) {

SecondQuestion1.setBounds(0, 0, 800, 600);

SecondQuestion1.setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

SecondQuestion1.setLocationRelativeTo(null);

SecondQuestion1.setVisible(true);

FirstQuestion1.dispose();

}

private void jButton15MouseClicked(java.awt.event.MouseEvent evt) {

ThirdQuestion1.setBounds(0, 0, 800, 600);

ThirdQuestion1.setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

ThirdQuestion1.setLocationRelativeTo(null);

ThirdQuestion1.setVisible(true);

SecondQuestion1.dispose();

}

private void jButton16MouseClicked(java.awt.event.MouseEvent evt) {

FourthQuestion1.setBounds(0, 0, 800, 600);

FourthQuestion1.setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

FourthQuestion1.setLocationRelativeTo(null);

FourthQuestion1.setVisible(true);

ThirdQuestion1.dispose();

}

private void jButton17MouseClicked(java.awt.event.MouseEvent evt) {

FifthQuestion1.setBounds(0, 0, 800, 600);

FifthQuestion1.setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

FifthQuestion1.setLocationRelativeTo(null);

FifthQuestion1.setVisible(true);

FourthQuestion1.dispose();

}

private void jButton18MouseClicked(java.awt.event.MouseEvent evt) {

SixthQuestion1.setBounds(0, 0, 800, 600);

SixthQuestion1.setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

SixthQuestion1.setLocationRelativeTo(null);

SixthQuestion1.setVisible(true);

FifthQuestion1.dispose();

}

private void jButton19MouseClicked(java.awt.event.MouseEvent evt) {

SeventhQuestion1.setBounds(0, 0, 800, 600);

SeventhQuestion1.setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

SeventhQuestion1.setLocationRelativeTo(null);

SeventhQuestion1.setVisible(true);

SixthQuestion1.dispose();

}

private void jButton20MouseClicked(java.awt.event.MouseEvent evt) {

EighthQuestion1.setBounds(0, 0, 800, 600);

EighthQuestion1.setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

EighthQuestion1.setLocationRelativeTo(null);

EighthQuestion1.setVisible(true);

SeventhQuestion1.dispose();

}

private void jButton21MouseClicked(java.awt.event.MouseEvent evt) {

NinethQuestion1.setBounds(0, 0, 800, 600);

NinethQuestion1.setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

NinethQuestion1.setLocationRelativeTo(null);

NinethQuestion1.setVisible(true);

EighthQuestion1.dispose();

}

private void jButton22MouseClicked(java.awt.event.MouseEvent evt) {

TenthQuestion1.setBounds(0, 0, 800, 600);

TenthQuestion1.setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

TenthQuestion1.setLocationRelativeTo(null);

TenthQuestion1.setVisible(true);

NinethQuestion1.dispose();

}

private void jButton23MouseClicked(java.awt.event.MouseEvent evt) {

EleventhQuestion1.setBounds(0, 0, 800, 600);

EleventhQuestion1.setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

EleventhQuestion1.setLocationRelativeTo(null);

EleventhQuestion1.setVisible(true);

TenthQuestion1.dispose();

}

private void jButton24MouseClicked(java.awt.event.MouseEvent evt) {

TwelvethQuestion1.setBounds(0, 0, 800, 600);

TwelvethQuestion1.setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

TwelvethQuestion1.setLocationRelativeTo(null);

TwelvethQuestion1.setVisible(true);

EleventhQuestion1.dispose();

}

private void jButton25MouseClicked(java.awt.event.MouseEvent evt) {

ThirteenthQuestion1.setBounds(0, 0, 800, 600);

ThirteenthQuestion1.setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

ThirteenthQuestion1.setLocationRelativeTo(null);

ThirteenthQuestion1.setVisible(true);

TwelvethQuestion1.dispose();

}

private void jButton26MouseClicked(java.awt.event.MouseEvent evt) {

FourteenthQuestion1.setBounds(0, 0, 800, 600);

FourteenthQuestion1.setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

FourteenthQuestion1.setLocationRelativeTo(null);

FourteenthQuestion1.setVisible(true);

ThirteenthQuestion1.dispose();

}

private void jButton27MouseClicked(java.awt.event.MouseEvent evt) {

FifteenthQuestion1.setBounds(0, 0, 800, 600);

FifteenthQuestion1.setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

FifteenthQuestion1.setLocationRelativeTo(null);

FifteenthQuestion1.setVisible(true);

FourteenthQuestion1.dispose();

}

private void jButton28MouseClicked(java.awt.event.MouseEvent evt) {

TestEnd.setBounds(0, 0, 800, 600);

TestEnd.setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

TestEnd.setLocationRelativeTo(null); // Centrē logu ekrānā

TestEnd.setVisible(true);

FifteenthQuestion.dispose();

try {

int correct1 = 0;

String[] answers = new String[15];

// Jautājums 1

if (jRadioButton21.isSelected()) { correct1++; answers[0] = "Ball"; }

else if (jRadioButton22.isSelected()) answers[0] = "Table";

else if (jRadioButton23.isSelected()) answers[0] = "Pen";

else if (jRadioButton24.isSelected()) answers[0] = "Pencil";

// Jautājums 2

String a2 = jTextField14.getText().trim();

answers[1] = a2;

if (a2.equalsIgnoreCase("Bag")) correct1++;

// Jautājums 3

String a3 = (String) jComboBox6.getSelectedItem();

answers[2] = a3;

if (a3.equalsIgnoreCase("Chair")) correct1++;

// Jautājums 4

if (jRadioButton25.isSelected()) { correct1++; answers[3] = "Pencil"; }

else if (jRadioButton26.isSelected()) answers[3] = "Pencilcase";

else if (jRadioButton27.isSelected()) answers[3] = "Pen";

else if (jRadioButton28.isSelected()) answers[3] = "Phone";

// Jautājums 5

String a5 = jTextField15.getText().trim();

answers[4] = a5;

if (a5.equalsIgnoreCase("Computer")) correct1++;

// Jautājums 6

String a6 = (String) jComboBox7.getSelectedItem();

answers[5] = a6;

if (a6.equalsIgnoreCase("Backpack")) correct1++;

// Jautājums 7

if (jRadioButton32.isSelected()) { correct1++; answers[6] = "Soap"; }

else if (jRadioButton29.isSelected()) answers[6] = "Hat";

else if (jRadioButton30.isSelected()) answers[6] = "Spoon";

else if (jRadioButton31.isSelected()) answers[6] = "Lamp";

// Jautājums 8

String a8 = jTextField16.getText().trim();

answers[7] = a8;

if (a8.equalsIgnoreCase("Mirror")) correct1++;

// Jautājums 9

String a9 = (String) jComboBox8.getSelectedItem();

answers[8] = a9;

if (a9.equalsIgnoreCase("Door")) correct1++;

// Jautājums 10

if (jRadioButton34.isSelected()) { correct1++; answers[9] = "Book"; }

else if (jRadioButton33.isSelected()) answers[9] = "Paper";

else if (jRadioButton35.isSelected()) answers[9] = "Folder";

else if (jRadioButton36.isSelected()) answers[9] = "Notebook";

// Jautājums 11

String a11 = jTextField17.getText().trim();

answers[10] = a11;

if (a11.equalsIgnoreCase("Pillow")) correct1++;

// Jautājums 12

String a12 = (String) jComboBox9.getSelectedItem();

answers[11] = a12;

if (a12.equalsIgnoreCase("Phone")) correct1++;

// Jautājums 13

if (jRadioButton39.isSelected()) { correct1++; answers[12] = "Bed"; }

else if (jRadioButton37.isSelected()) answers[12] = "Chair";

else if (jRadioButton38.isSelected()) answers[12] = "Table";

else if (jRadioButton40.isSelected()) answers[12] = "Shelf";

// Jautājums 14

String a14 = jTextField18.getText().trim();

answers[13] = a14;

if (a14.equalsIgnoreCase("Ruler")) correct1++;

// Jautājums 15

String a15 = (String) jComboBox10.getSelectedItem();

answers[14] = a15;

if (a15.equalsIgnoreCase("Scissors")) correct1++;

// Saglabāt atbildes failā

try (BufferedWriter bw = new BufferedWriter(new FileWriter("answersArtjomenko.txt"))) {

for (int i = 0; i < answers.length; i++) {

bw.write((i + 1) + ". jautājums: " + answers[i]);

bw.newLine();

}

}

int percentage = (int) Math.round(correct1 \* 100.0 / 15);

String grade;

String message;

if (percentage >= 90) {

grade = "10";

message = "Ļoti labi paveikts darbs";

} else if (percentage >= 80) {

grade = "9";

message = "Ļoti labi paveikts darbs";

} else if (percentage >= 70) {

grade = "8";

message = "Labi, bet vajag pacensties";

} else if (percentage >= 60) {

grade = "7";

message = "Labi, bet vajag pacensties";

} else if (percentage >= 50) {

grade = "6";

message = "Vajag atkārtot tēmu";

} else if (percentage >= 40) {

grade = "5";

message = "Vajag atkārtot tēmu";

} else if (percentage >= 20) {

grade = "3–4";

message = "Slikti";

} else if (percentage >= 1) {

grade = "1–2";

message = "Ļoti slikti";

} else {

grade = "0";

message = "Nepietiekami";

}

// Aprēķināt neatbildēto jautājumu skaitu

int unanswered = 0;

try (BufferedWriter bw = new BufferedWriter(new FileWriter("resultsArtjomenko.txt"))) {

String name = jTextField1.getText().trim();

String lastname = jTextField2.getText().trim();

bw.write("Vārds: " + name);

bw.newLine();

bw.write("Uzvārds: " + lastname);

bw.newLine();

bw.write("Pareizās atbildes: " + correct1 + " no 15");

bw.newLine();

bw.write("Procentuāli: " + percentage + "%");

bw.newLine();

bw.write("Atzīme: " + grade);

bw.newLine();

bw.write("Vērtējums: " + message);

}

// Izvade TextFieldos

jTextField10.setText(grade); // atzīme

jTextField11.setText(percentage + "%"); // procenti

jTextField12.setText(String.valueOf(unanswered)); // neatbildētie

jTextField13.setText(message); // komentārs par darbu

} catch (Exception e) {

JOptionPane.showMessageDialog(this, "Kļūda pārbaudot atbildes!", "Kļūda", JOptionPane.ERROR\_MESSAGE);

e.printStackTrace();

}

}