VIDZEMES AUGSTSKOLA

INŽENIERZINĀTŅU FAKULTĀTE

**Ziņu rakstu ātrrakstīšanas programma**

Gada projekts

Darba autors: Roberts Ronins Puķītis

Stud. apl. nr.: IT20042

Darba vadītājs: Miķelis Baltruks

Valmiera 2022

# **KOPSAVILKUMS**

Autors : Roberts Ronins Puķītis, Stud.apl.nr: IT20042

Darbs vadītājs: Mg.sc.comp. Miķelis Baltruks

**Ziņu rakstu ātrrakstīšanas programma** – Gada darbs, Valmiera: Vidzemes augstskola 2022 – 17 lapaspuses.

Pētījumi rāda, ka pēdējo gadu laikā cilvēku spēja noturēt uzmanību ir samazinājusies (Hayes, 2022), kas traucē ikdienas produktivitātei, piemēram, jaunāko ziņu rakstu lasīšanai.

Mūsdienās ievērojama daļa no ikdienas tiek pavadīta pie datora ekrāna, it īpaši veicot darbus attālinātajā režīmā, tādēļ ir nācies iemainīt rakstīšanu ar pildspalvu pret rakstīšanu ar klaviatūru.

Darba mērķis ir izveidot programmu ar ātrrakstīšanas testu, kurā lietotājs var izvēlēties kādu no piedāvātajiem ziņu portāliem, no kā vēlas lasīt ziņas, un tad ātrrakstīšanas testa gaitā lasīt un rakstīt uz ekrāna krītošos vārdus, kas ņemti no izvēlētā ziņu raksta, tādējādi uzlabojot savu lasītprasmi, rakstītprasmi uz klaviatūras un izlasot jaunākos ziņu rakstus.

Darba laikā tika izstrādāta programma ar ātrrakstīšanas testu, ko autors vēlas uzlabot, lai paaugstinātu produkta kvalitāti.

# **SUMMARY**

Author: Roberts Ronins Puķītis, Stud.apl.nr: IT20042

Supervisor: Mg.sc.comp. Miķelis Baltruks

**News article speed typing program** – Thesis, Valmiera: Vidzeme University of Applied Sciences 2022 – 17 pages.

Research shows that in recent years, the average time that people can keep attention has reduced (Hayes, 2022), which can reduce daily productivity, for instance reading recent news articles.

These days, a significant portion of our daily lives is spent at the computer screen, especially while doing work from home. While working remotely, pen and paper has been traded in for using the keyboard.

The aim of this work is to develop a program with a speed typing game, where the user can choose one of the provided news websites, from which data shall be sent to the programs speed typing test. During the playthrough of the test, words from the chosen news article will fall from the top of the screen and the user will have to type them out, resulting in an improvement in reading ability, improving keyboard writing skills and gaining knowledge about the most recent news articles.

During the work, the author developed a program with a speed typing game, which the author wishes to improve to develop a more qualitative product.

# **SAĪSINĀJUMI UN ATSLĒGAS VĀRDI**

OOP - objektu orientēta programmēšana,

atpakaļsaderīgs – funkcionē ar aplikācijas vecākām versijām,

API - Application Programming Interface (Aplikācijas Programmēšanas Interfeiss),

IDE - integrated development environment (integrēta izstrādes vide),

Spraudņi - plugins (datorprogramma, kas sadarbojas ar citu programmu un papildina tās funkcionalitāti),

Mākonis - serveru tīkls, kas sūta, saņem un uzglabā datus,

RSS - Really Simple Syndication (Tiešām vienkārša sindikācija),

XML - Extensible Markup Language (Paplašināmā iezīmēšanas valoda),

Webscrape - datu nolasīšana no tīmekļa pārlūkprogrammas,

Classpath - parametrs java kompilētājam, kas nosaka lokāciju lietotāja pievienotām klasēm, bibliotēkām vai pakotnēm,

Maven - rīks programmatūru projekta menedžmentam, kas atvieglo projekta uzbūves veidošanu,

HTML - HyperText Markup Language (hiperteksta iezīmēšanas valoda) : iezīmēšanas valoda, kas tīmekļa lapās glabā attēlojamo informāciju,

Diakritiskās zīmes - papildu rakstu zīmes, kas tiek rakstītas virs vai zem [burta](https://lv.wikipedia.org/wiki/Burts), blakus vai pāri tam, lai norādītu citādu burta [izrunu](https://lv.wikipedia.org/w/index.php?title=Izruna&action=edit&redlink=1), nekā tas parasti tiktu izrunāts (garumzīmes un mīkstinājuma zīmes).

# **SATURS**

[KOPSAVILKUMS 2](#_Toc104507494)

[SUMMARY 3](#_Toc104507495)

[SAĪSINĀJUMI UN ATSLĒGAS VĀRDI 4](#_Toc104507496)

[SATURS 5](#_Toc104507497)

[IEVADS 6](#_Toc104507498)

[1. PROJEKTA IZPĒTE 7](#_Toc104507499)

[1.1. **Vārdu uztvere** 7](#_Toc104507500)

[1.2. **Ātrrakstīšanas testi** 7](#_Toc104507501)

[1.3. **Rakstīšanas ātrums** 7](#_Toc104507502)

[1.4. **Java programmēšanas valoda** 8](#_Toc104507503)

[1.5. **Eclipse IDE** 9](#_Toc104507504)

[1.6. **Projekta pamats** 9](#_Toc104507505)

[1.7. **RSS plūsmu parsētājs** 9](#_Toc104507506)

[1.8. **Lietotāja rokasgrāmata** 10](#_Toc104507507)

[1.9. **Programmatūras prasību specifikācija (PPS)** 11](#_Toc104507508)

[1.10. **Programmatūras prasību specifikācija (PPS)** 14](#_Toc104507509)

[2. Programmas struktūra 16](#_Toc104507510)

[3. Programmas funkciju demonstrēšana 17](#_Toc104507511)

[3.1. **Navigācija.** 17](#_Toc104507512)

[3.2. **Grūtības pakāpes izvēle.** 19](#_Toc104507513)

[3.3. **Ziņu portāla izvēle.** 20](#_Toc104507514)

[3.4. **Ziņu raksta izvēle.** 21](#_Toc104507515)

[3.5. **Testa logs.** 21](#_Toc104507516)

[3.6. **Rezultātu logs.** 22](#_Toc104507517)

[4. DARBA EKONOMISKAIS PAMATOJUMS 24](#_Toc104507518)

[4.1. **Izmaksas** 24](#_Toc104507519)

[4.2. **Potenciālie ienākumi** 24](#_Toc104507520)

[3. SECINĀJUMI 25](#_Toc104507521)

[4. LITERATŪRA 26](#_Toc104507522)

[5. PIELIKUMI 27](#_Toc104507523)

# **IEVADS**

Pētījumi rāda, ka pēdējo gadu laikā, cilvēku spēja uzturēt uzmanību ir samazinājusies, kas traucē ikdienas produktivitātei, piemēram jaunāko ziņu rakstu lasīšanai. Ir izteikts viedoklis, ka viens no galvenajiem iemesliem šai parādībai ir palielināts datora lietošanas laiks. (Hayes, 2022)

Mūsdienās proporcionāla daļa no ikdienas tiek pavadīta pie datora ekrāna, it īpaši veicot darbus attālinātajā režīmā. Attālināti strādājot ir nācies iemainīt rakstīšanu ar pildspalvu pret rakstīšanu ar klaviatūru.

Problēma: Jaunāko ziņu rakstu lasīšana ir apgrūtināta, samazinātās uzmanības uzturēšanas dēļ.

Ir nepieciešamība uzlabot lasītprasmi pie datora un rakstītprasmi ar klaviatūru.

**Darba uzdevumi :**

* Izveidot programmu, kurā lietotājs var
  + izvēlēties kādu no piedāvātajiem ziņu portāliem,
  + izvēlēties sev piemērotu lasīšanas ātrumu,
  + veikt ātrrakstīšanas testu ar ziņu rakstiem
  + saņemt vērtējumu par savām ātrrakstīšanas spējām
  + salīdzināt savas ātrrakstīšanas spējas ar citiem lietotājiem
* Saglabāt ziņu rakstus un lietotāju ātrrakstīšanas vērtējumus teksta failā.
* Izmantot kvalitatīva pētījuma metodes………………..

**Darba mērķis:**

Izveidot programmu ar ātrrakstīšanas testu, kurā lietotājs var izvēlēties kādu no piedāvātajiem ziņu portāliem, no kā tiks sūtīti raksti uz ātrrakstīšanas testu, kuras gaitā vajadzēs rakstīt uz ekrāna krītošos vārdus no ziņu rakstiem, tādējādi uzlabojot savu lasītprasmi, rakstītprasmi uz klaviatūras un izlasot jaunākos ziņu rakstus.

# **PROJEKTA IZPĒTE**

## **Vārdu uztvere**

Aplūkojot vārdus, tiek stimulēta cilvēka objektu atpazīšanas spēja, atkarībā no tā, attiecīgā vārda konteksta. Lasot tiek ne tikai tiek apstrādāta informācija par paša vārda būtību, bet arī cilvēka domās tiek izstrādāta vizuāla reprezentācija vārdiem, jo tie virza vizuālo perspektīvu, aktivizējot informācijas uztveri par objektu formām. Īpašības kā krāsa vai garša tiek iztēlota, vārdu lasīšanas laikā (Noorman, 2018).

## **Ātrrakstīšanas testi**

Ātrrakstīšanas tests ir ne tikai veids kā pārbaudīt lietotāja rakstīšanas ātrumu, bet arī ir veids kā paātrināt rakstīšanas ātrumu un mazināt pieļauto rakstības kļūdu apjomu (Sikowski, 2019). Ātrs rakstīšanas ātrums pie datora ir svarīga prasme mūsdienās, jo rakstīšana ar klaviatūru ir iesaistīta lielākā daļā no darbiem, it īpaši ņemot vērā attālinātās strādāšanas situācijas. Tas ļauj ātrāk un efektīvāk veikt darbu un komunicēt ar kolēģiem vai darba vadītajiem (ABM College, 2015).

Internetā ir atrodamas vairākas saites ar ātrrakstīšanas testiem, starp kuriem lielākā daļa izmanto nejauši ģenerētus vārdus, taču ir arī tādas, kas lietotājam ļauj pašam ievietot savu tekstu, lai to izmantotu testam, saglabātu un tad dalītos ar to (Key Hero, 2022). Visbiežāk testā lietotājam ir dots laika limits 60 sekundes, lai uzrakstītu pēc iespējams vairāk vārdus bez kļūdu pieļaušanas, taču ir arī saites, kas lietotājam ļauj pašam izvēlēties gan laika limitu gan arī vārdu limitu (Typing, 2022).

## **Rakstīšanas ātrums**

Ātrākais rakstīšanas ātrums uz klaviatūras pasaulē ir 216 vārdi minūtē, taču vidējais ātrums ir ap 40 vārdi minūtē, starp kuriem 8% saturēs kļūdu (Ratatype, 2022). Produktīvam darbam pie datora ir piemērots rakstīšanas ātrums

Rakstīšanas ātruma uzlabošanai, ir jācenšas ievērot attiecīgus noteikumus (Herzing University, 2019):

* + 1. **Pareizs plaukstu novietojums:**

Labā prakse ir turēt kreiso plaukstu tā, lai pirksti būtu virs “A”, “S”, “D” un “F” pogām, kamēr labā roka tā, lai pirksti ir virs “J”, “K”, “L” un “;” pogām. Abu roku īkšķiem ir jābūt uz atstarpēšanas taustiņa(spacebar).

* + 1. **Skatīšanās tikai uz ekrānu:**

Rakstot ar klaviatūru nav vajadzība skatīties uz klaviatūru, jo taustiņu pozīcijām jau jābūt iegaumētām. Uzreiz redzot, kas uz ekrāna tiek rakstīts, ir daudz vieglāk pamanīt kļūdas rakstītajā tekstā. Iesācējiem rakstīšana skatoties tikai uz ekrānu ir nedaudz grūts uzdevums, jo vēl nav pieraduši pie klaviatūras izkārtojuma, taču ar laiku un trenēšanos var atcerēties, katra taustiņa pozīciju.

* + 1. **Pareizas ķermeņa pozas noturēšana:**

Sēdēšana ar iztaisnotu muguru padarīs rakstīšanas procesu vieglāku, un uzlabos rakstīšanas ātrumu. Pareiza sēdēšanas poza arī veicinās labāku elpošanu un asins cirkulāciju, mazinās muguras sāpes un dos enerģijas sajūtu (Gordon, 2019).

* + 1. **Ērta roku pozīcija:**

Nepareizas roku pozīcijas noturēšana uz ilgu laiku izraisīs neērtības rakstīšanai ar klaviatūru. Nekad nevajadzētu vajadzībai pārmērīgi locīt rokas, kamēr lieto klaviatūru.

* + 1. **Trenēšanās:**

Katru dienu rakstot uz klaviatūras, tiks labāk iegaumētas taustiņu pozīcijas un uzlabota rakstīšanas precizitāte, ātrums un kļūdu pamanīšana. Internetā ir atrodami dažādi bezmaksas ātrrakstīšanas testi un treniņi, ar ko var veikt vingrinājumus.

## **Java programmēšanas valoda**

Programma tiek rakstīta Java programmēšanas valodā. Java ir plaši pielietota objektu orientēta programmēšanas valoda, kurā veidotais kods tiek efektīvi kompilēts jebkurā OS platformā, kas atbalsta javu (Hartman, 2022). Šajā projektā java valodas pielietošanai tiek izmantota Eclipse IDE.

## **Eclipse IDE**

Vide, kas atrod balansu starp funkciju daudzumu un veiktspēju ar vieglu konstrukciju. Izlaista 2001. gadā un pieejama uz Linux, Mac, Solaris un Windows operētājsistēmām, Eclipse tiek uzskatīta kā labākais IDE priekš java(Singh, 2022).

* + 1. **Eclipse IDE plusi:**
* pieejami vairāki spraudņi,
* pieeja mākonim,
* pieeja diagrammu, modelēšanas, atskaites un testēšanas rīkiem,
* atbalsta citas programmēšanas valodas, izmantojot spraudņus,
* bezmaksas programma.
  + 1. **Eclipse IDE mīnusi:**
* sarežģīta iestatīšana
* nedaudz novecojusi vide, salīdzinājumā ar citām

## **Projekta pamats**

Projekta pamata ideja ir, ka ziņu rakstu lasīšanās integrēšana ātrrakstīšanas testā padarīs lasīšanas procesu interaktīvāku, interesantāku, un uzlabos lietotāja rakstīšanas prasmi ar datora klaviatūru. Programmā esošās top rezultātu tabulas dēļ lietotājs varēs arī novērtēt savu rakstīšanas ātrumu salīdzinājumā ar citiem lietotājiem, un pat varētu tikt motivēts vairāk trenēt savu prasmi ar klaviatūru, lai nonāktu augstākā pozīcijā uz rezultātu tabulas.

## **RSS plūsmu parsētājs**

RSS plūsma ir XML teksta fails, ko autors no kādas saites papildina ar jaunākajiem ierakstiem. (Johnson, 2021). RSS plūsmu parsētājs nolasa datus no izvēlētām saitēm, kurām ir XML faili.

Ir interneta saites, kas piedāvā RRS plūsmas kā vienu no veidiem kā sekot līdzi jaunākajai informācijai. Faili tiek patstāvīgi atjaunoti ar jauniem ierakstiem. Ziņu portāli, kuriem ir savas RRS plūsmas:

* Delfi
* Latvijas Sabiedriskie Mediji (LSM)
* Dienas bizness (db)

Eclipse IDE vidē RSS plūsmu konstruēšanai nav vajadzīgi lieki importi vai bibliotēkas un nav jākonfigurē classpath vērtības.

Alternatīvs RSS plūsmai ir jsoup (java HTML Parser) - bibliotēka, kas paredzētā strādāšanai ar HTML datiem. Tā var iegūt un manipulēt html datus izmantojot HTML5 DOM metodes. jsoup bibliotēka ir pieejama lejuplādēšanai internetā, vai arī var tikt ieviesta, izmantojot maven atkarīgo mainīgo.

Projektam tiek lietotas RSS plūsmas, jo ar tām var iegūt visus vajadzīgos datus, un to strādāšanai nav vajadzība papildus komponentu lejuplādēšanai un instalēšanai java vidē.

## **Lietotāja rokasgrāmata**

Vispirms projektam ir jātiek importētam, kas Eclipse IDE ir iespējams caur “file->import->Existing Projects into Workspace”, kur var izvēlēties projekta pakotni un to importēt.

Programmu galvenais fails, ar ko tā ir jāuzsāk programma, ir “mainSpeedTypeArticles”, kas atrodama pakotnē “src\newArticelSpeedTyping”.

Atveroties programmai, galvenajā izvēlnē ir iespējams aplūkot info logu un testa sagataves logu, nospiežot uz attiecīgās pogas.

Testa sagataves logā ir iespēja redzēt un naviģēt top rezultātu tabulu, izvēlēties un grūtības pakāpi testam, vai atvērt ziņu portāla izvēlni. Ziņu portāla izvēlnē lietotājs izvēlas saiti, no kuras vēlas saņemt ziņu rakstus, nospiežot uz attiecīgās pogas, un izvēlēties ziņu raksta numuru, ievadot to attiecīgajā laukā. Ir arī iespēja izvēlēties “file” opciju, kura gadījumā testā tiks lietoti vārdi no “tempText.txt” teksta faila. Kad izvēles ir tikušas veiktas, atgriežoties uz testa sagataves logu, var spiest “Play” pogu, lai uzsāktu testu.

Uzsākot ātrrakstīšanas testu, notiek atskaite no 3, pēc kuras no loga augšpuses sāk krist vārdi, kas lietotājam jāuzraksta un tad jāievada, uz klaviatūras nospiežot pogu “enter”. Lai testā rakstītu vārdus, ir jābūt atvērtam programmas logam. Rakstītie burti un simboli, kā arī uzrakstīto vārdu un kļūdu skaits, būs redzami loga apakšdaļā. Punktuālās rakstzīmes un diakritiskās zīmes arī tiek ņemtas vērā vārda pareizas ievades pārbaudē. Krītošie vārdi tiek ņemti attiecīgā secībā no lietotāja izvēlētā ziņu portāla jaunākajiem ziņu virsrakstiem. Jauns vārds uzrodas vai nu pēc noteikta intervāla, kas ir atkarīgs no grūtības pakāpes, vai arī, ja uz ekrāna vairs nav krītošo vārdu.

Testa laikā lietotājs jebkurā brīdī var nospiest pogu “reset”, lai no jauna sāktu testu, vai arī pogu “back”, kas izslēgs testu un atvērs testa sagataves logu.

Tests beidzas, kad lietotājs ir ievadījis vai palaidis garām visus krītošos vārdus. Pēc testa, uz ekrāna ir redzami lietotāja rakstīšanas ātrumi un saite uz pilnu ziņu rakstu. Lietotājam tiks prasīts, lai ievada savu lietotājvārdu, lai to un iegūtos rakstīšanas ātrumus pievienotu top rezultātu tabulai un teksta failam. Pēc vārda

## **Programmatūras prasību specifikācija (PPS)**

Programmas funkcijas nodrošina lietotājam iespēju

1. naviģēt starp logiem, klikšķinot uz pogām,
2. naviģēt top rezultātu tabulu,
3. izvēlēties testa grūtības pakāpi,
4. izvēlēties ziņu portālu,
5. izvēlēties ziņu raksta numuru,
6. ievadīt vārdus testā.
   * 1. **Navigācija**
        1. **Ievads:**

Lietotājs var naviģēt starp logiem, klikšķinot uz pogām, kas redzamas programmas logā.

* + - 1. **Ievade:**

Lietotājs, esot galvenās izvēlnes logā, nospiež uz “READ ARTICLES” pogu, lai nokļūtu testa sagataves logā.

* + - 1. **Apstrāde:**
* Lietotājs programmas logā novieto kursoru uz pogas “READ ARTICLES”.
* Lietotājs noklikšķina uz pogas “READ ARTICLES”, neizvirzot kursoru no tās zonas. Ja kursors neatrodas pogas zonā, nekas nenotiek.
* Programmas interfeiss maina logu pozīciju
  + No sāna sāk virzīties uz centru testa sagataves logs. Tas kustas līdz nonācis līdz aizņem visu programmas ekrānu. Tiek veikta pozīcijas pārbaude, vai logs ir gala pozīcijā. Ja nē: turpinās kustība.
  + Galvenās izvēlnes logs tiek kustināts tajā pašā virzienā, kā testa sagataves logs, līdz galvenā izvēlne vairs nav redzama. Tiek veikta pozīcijas pārbaude, vai logs ir gala pozīcijā. Ja nē: turpinās kustība.
    - 1. **Izvade:**

Programmā izvēlētā loga vērtība ir testa sagataves logs. Galvenās izvēlnes logs vairs nav redzams. Ir ticis atvērts testa sagataves logs.

* + 1. **Grūtības pakāpes izvēle**
       1. **Ievads:**

Lietotājs var izvēlēties grūtības pakāpi, spiežot attiecīgās pogas testa sagataves logā.

* + - 1. **Ievade:**

Lietotājs, esot ziņu izvēlnes logā, nospiež uz “MEDIUM” pogas, lai programma iestatītu testa grūtības pakāpi uz vidēju.

* + - 1. **Apstrāde:**
* Programmas logā kursors tiek novietots uz pogas “MEDIUM”.
* Tiek noklikšķināta poga “MEDIUM”. Ja kursors klikšķināšanas laikā vairs neatrodas pogas zonā, nekas nenotiek.
* Tiek izvadīta informācija par izvēli.
  + Zem grūtības pakāpju pogām, informācijas teksts tiek atjaunots ar vērtību “Current difficulty : MEDIUM”.
  + Poga “MEDIUM” uzrādās kā atzīmēta.
    - 1. **Izvade:**

Programmā grūtības pakāpes vērtība tiek iestatīta kā “MEDIUM”. Grūtības pakāpes teksts uzrāda vērtību “Current difficulty : MEDIUM”. Poga “MEDIUM” uzrādās kā atzīmēta.

* + 1. **Ziņu portāla izvēle**
       1. **Ievade:**

Lietotājs, esot ziņu izvēlnes logā, nospiež uz “Delfi(bizness)” pogas, lai programma nolasītu ziņu rakstus no Delfi biznesa stāstu sadaļas.

* + - 1. **Apstrāde:**
* Programmas logā kursors tiek novietots uz pogas “Delfi(bizness)”.
* Tiek noklikšķina poga “Delfi(bizness)”. Ja kursors klikšķināšanas laikā vairs neatrodas pogas zonā, nekas nenotiek.
* Notiek RRS plūsmas savienojums ar Delfi(bizness) portālu.
* No Delfi(bizness) portāla tiek nolasīti dati par ziņu rakstiem.
* Tiek izvadīta informācija par izvēli.
  + Virs ziņu portālu pogām, informācijas teksts tiek atjaunots ar vērtību “Currently selected : Delfi(bizness)”.
  + Ir redzams teksts, kas paziņo cik ziņu raksti ir pieejami Delfi(bizness) portālā.
  + Saites teksta laukumā atjaunojas teksts ar attiecīgā numura ziņu raksta saiti.
    - 1. **Izvade:**

Programmā ziņu portāla vērtība tiek iestatīta kā “Delfi(bizness)”. Programma ir ieguvusi vērtības par ziņu rakstiem Delfi(bizness) portālā. Ir redzami paskaidrojošie teksti par izvēlēto ziņu portālu un ziņu rakstu skaitu tajā. Ir atjaunota ziņu raksta saite.

* + 1. **Ziņu raksta numura izvēle**
       1. **Ievads:**

Lietotājs var izvēlēties ziņu raksta numuru, ievadot skaitli teksta logā.

* + - 1. **Ievade:**

Lietotājs, esot ziņu izvēlnes logā, ievada ziņu raksta numuru “6”, lai izvēlētos attiecīgo ziņu rakstu.

* + - 1. **Apstrāde:**
* Programmas logā tiek noklikšķināts numura ievades teksta lauks.
* Numura ievades laukā tiek izdzēsts numurs, kas tur bija.
* Numura ievades laukā tiek ierakstīts numurs “6”.
* Kursors tiek novietots uz pogas “Select NR”.
* Tiek noklikšķina poga “Select NR”. Ja kursors klikšķināšanas laikā vairs neatrodas pogas zonā, nekas nenotiek.
* Zem numura izvēles tiek atjaunots informācijas teksts ar vērtību “Currently selected article number : 6”.
* Saites teksta laukā tiek atjaunots teksts ar attiecīgā numura saiti.
  + - 1. **Izvade:**

Programmā saites indeksa vērtība tiek atjaunota. Ir redzams informācijas teksts par izvēlēto saites numuru. Ir atjaunots saites teksts ar attiecīgā numura saiti.

* + 1. **Vārdu ievade testa laikā.**
       1. **Ievads**

Lietotājs testā var uzrakstīt un tad ievadīt vārdus, izmantojot klaviatūru.

* + - 1. **Ievade**

Lietotājs testa laikā ievada vārdu “turpmāk”.

* + - 1. **Apstrāde:**
* Uz klaviatūras spiežot taustiņus, tiek uzrakstīts vārdu “turpmāk”.
* Uz klaviatūras tiek nospiests “enter” taustiņš.
* Tiek veikta pārbaude vai ievadītais vārds ir vienāds ar kādu no krītošajiem vārdiem. Ja nē, tad teksta ievades laukā tiek izdzēsts ierakstītais un pieaug pieļauto kļūdu skaits.
  + - 1. **Izvade**

Attiecīgais krītošais vārds pazūd. Uzrakstīto vārdu skaits paaugstinās. Ja uz ekrāna vairs nepastāv krītošie vārdi, bet vēl ir pieejami vārdi, tad tiek radīts jauns krītošais vārds.

## **Programmatūras projektējuma apraksts (PPS)**

* + 1. **Ievads**

Programma tika veidota izmantojot Eclipse IDE, un ir paredzēta ātrrakstīšanas testa pildīšanai.

* + - 1. **Nolūks**

PPA tiek veidots, lai aprakstītu un paskaidrotu programmas galvenās darbības.

* + - 1. **Darbības sfēra**

Programma tiek veidota izmantojot java programmēšana valodu.

* + - 1. **Definīcijas un saīsinājumi**

Metode – kods, kas tiek izpildīts, kad tas tiek izsaukts. Metodem var tikt padotas vērtības, un tās var atgriest vērtību.

JFrame – objekts, kas veido programmas logu

JLabel - objekts ikonas un/vai teksta uzrādīšanai

JLayeredPane - slāņots konteiners, kas ļauj izvēlēties pārklātu objektu uzrādīšanas secību.

HashMap –datu saraksts, kas izmanto unikālas atslēgas datu piekļuvei.

Parsēt – pārveidot vērtību no viena datu tipa uz citu

* + 1. **Saistības ar citiem dokumentiem**

PPA tika veidots balstoties uz Latvijas valsts programminženierijas ieteicamo praksi programmatūras projektējuma aprakstīšanai – LVS 72:1996

* + 1. **Dekompozīcijas apraksts**
       1. **Programmas logu veidošana**

Uzsākot programmu, tiek iestatītas logu izmēru vērtības : “frameWidth” un “frameHeight”. Izmēru vērtības tiek padotas “MainMenu” konstruktoram, kurā tiek izsaukta “setupScreen” metode (Skat. Pielikums I), padodot tai galvenās izvēlnes JFrame objektu un izmēru vērtības. “setupScreen” metode inicializē JFrame objektu un tam piešķir īpašības, izmantojot JFrame metodes, pēc kurām inicializētais objekts tiek atgriezts.

“GameScreen” klasē atrodas cita “setupScreen” metode, no kuras tiek izsaukta “MainMenu” klases “setupScreen” metode, lai tiktu izveidots programmas logs testam.

* + - 1. **Navigācija pa programmas logiem**

Navigāciju programmas saskarnē nodrošina “ProgrammButton” objekti, “Mouse” klase, “timer” objekti un “JavaLayeredPane” objekti.

“JavaLayeredPane” klase (Skat. Pielikums II) paplašina “JLayeredPane” objektu, lai tam pievienotu jaunas metodes un optimizētu kodu. Katrs “JavaLayeredPane” objekts tiek pievienots konstruktorā padotajam “HashMap” objektam. Pievienotās metodes:

1. “setSlidePoint” nosaka objekta starta koordinātas, uz kurām tam ir jānonāk.
2. “setCoordinates” izsauc objekta “setLocation” metodi, padodot tai atgrieztās vērtības no “moveScreen” metodes.
3. “checkMovement” pārbauda vai objekam jākustas uz uz starta pozīciju vai uz nulles pozīciju (aizņemt visu ekrānu).
4. “moveScreen” kustina objektu uz padoto x vai y koordinātu, salīdzinot ar padoto objekta pozīciju.

“ProgrammButton” klase (Skat. Pielikums III) paplašina “JLabel” objektus. Tās konstruktorā tiek izsaukts “Mouse” konstruktors, kuram padod “JLabel” objektu, lai tam piešķirtu “MouseListener” saskarni. Katrs “ProgrammButon” objekts tiek pievienots tā konstruktora padotajam “HashMap” objektam. Katra “ProgrammButon” izpildes funkcija, ieskaitot logu kustināšanas metožu izsaukšanu, ir ierakstīta “Mouse” klases metodē “mousePressed” un “mouseReleased” (Skat. Attēls 1).

Gan “MainMenu” klasē atrodas “timer” objekts “menuTimer”, kas ar katru ciklu izsauc “actionPerformed” metodi, kurā atrod “for” cikls, kas iterē cauri “JavaLayeredPane” objektiem un tos kustina, ja tie neatrodas noteiktajā pozīcijā.

Text

Description automatically generated

Attēls 1. Koda fragments no "Mouse" klases "mouseReleased" metodes

* + - 1. **Grūtības pakāpes izvēle**

Grūtības pakāpi var izvēlēties spiežot uz kādu no grūtības pakāpes pogām, “EASY”, “MEDIUM” un “HARD”, testa sagataves logā. Nospiežot pogu, “Mouse” klases “mouseReleased” metodē “for” cikls iterē cauri pogu sarakstam (Skat. Attēls 2), lai noteiktu kura tika nospiesta.

Text

Description automatically generated

Attēls 2. Koda fragrments no "Mouse" klases "mouseReleased" metodes

Kad tiek noteikta poga, tās nosaukuma vērtība tiek piešķirta “difficulty” mainīgajam, pēc kā balstoties “GameScreen” klases “setupParameters2” metodē (Skat. Attēls 3) tiek noteikts krītošo vārdu kustības vērtība un uzrašanās intervāls.

Text

Description automatically generated

Attēls 3. Koda fragrments no "GameScreen" klases "setupParameters2" metodes

* + - 1. **Ziņu portāla izvēle**

Ziņu portālu var izvēlēties spiežot uz kādu ziņu portāla pogām, “Delfi(sabiedriba)”, “LSM(populārās)” un “Delfi(visas)”, ziņu izvēlnes logā. Nospiežot pogu, “Mouse” klases “mouseReleased” metodē izpildās attiecīgās pogas kods (Skat. Attēls 4), kurā pārbauda vai ziņu portāls jau ir izvēlēts un izsauc “MainMenu” klases “switchNews” metodi, padodot tai ziņu portāla saiti un pogas nosaukumu.

Text

Description automatically generated

Attēls 4. Koda fragrments no "Mouse" klases "mouseReleased" metodes

“switchNews” metode (Skat. Pielikums IV) izsauc “RssData” klases (Skat. Pielikums V) konstruktoru, kurā tiek inicializēts RSS parsētājs, ar ko tiek iegūti ziņu rakstu dati.

* + - 1. **Ziņu raksta izvēle**

Atrodoties ziņu izvēlnes logā lietotājs teksta logā var ievadīt skaitli. Nospiežot “SELECT NR” pogu, “Mouse” “mouseReleased” pogas koda (Skat. Attēls 5) izpildē tiek veikta pārbaude vai ievadītais skaitlis ir parsējams uz “int” vērtību, un vai vērtība ir starp 0 un kopējo ziņu rakstu skaitu. Pārbaudu izpildes gadījumā, vērtība tiek piešķirta “articleIndex” mainīgajam, kas tiek izmantots kā indekss prekš ziņu rakstiem.

A screenshot of a computer

Description automatically generated with medium confidence

Attēls 5. Koda fragrments no "Mouse" klases "mouseReleased" metodes

* + - 1. **Vārdu krišana testa laikā**

Testa laikā “GameScreen” klasē tiek veidoti “FallingWord” objekti (Skat. Pielikums VI), kas satur vārdus no nolasītā datu avota. “GameScreen” klasē darbojošais “timer” objekts izsauc “actionPerformed” metodi, kurā tiek ģenerēts jauns krītošais vārds, ja laika intervāls ir pagājis vai uz ekrāna vairs nav vārdu, un tiek kustināti visi krītošie vārdi. Ja vārds ir nokritis zemāk par programmas logu, tas tiek izdzēsts (Skat. Attēls 6).

A screenshot of a computer

Description automatically generated with medium confidence

Attēls 6. Koda fragrments no "GameScreen" klases "actionPerformed" metodes

* + - 1. **Vārdu ievade testa laikā**

Testa laikā, lai varētu ievadīt uz ekrāna redzamos krītamos vārdus, loga apakšā ir pieejams “JTextArea” objekts, kurā lietotājs ieraksta vajadzīgos vārdus. “GameScreen” klasē darbojošais “timer” objekts izsauc “actionPerformed” metodi, kurā, ja “JTextArea” objekta tekstā ir parādījusies otra rinda, jeb lietotājs ir nospiedis pogu “enter”, tiek izsaukta “checkInput” metode.

“checkInput” metodē (Skat. Pielikums VII) izdzēš ievadītajā tekstā esošo jaunas rindas simbolu (\n) un pārbauda vai tests nav beidzies. Ja tests nav beidzies, tad tiek izsaukta metode “matchWord” (Skat. Attēls 7), padodot tai ievadītā teksta vērtību, lai pārbaudītu vai ievadītais teksts ir vienāds ar kādu no krītošajiem vārdiem, kuru gadījumā tas vārds tiek izdzēsts, uzrakstīto vārdu skaits tiek paaugstināts un uzrakstīto simbolu skaitam tiek pieskaitīts uzrakstītā vārda simbolu skaits.

Text

Description automatically generated

Attēls 7. "GameScreen" klases metode "matchWord"

Ja tests ir beidzies, tad tiek izsaukta “checkNameValidity” metode, lai pārbaudītu vai ievadītais tests atbilst lietotājvārda nosacījumiem : Ievades lauks nedrīkst būt tukšs, un ievadītais teksts nevar pārsniegt 20 simbolu garumu (Skat. Attēls 8).

Text

Description automatically generated

Attēls 8. "GameScreen" klases metode "checkNameValidity"

* + 1. **Atkarības apraksts**
    2. **Saskarnes apraksts**
    3. **Datu detalizētais projektējums**

# **Programmas struktūra**

Mērķa sasniegšanai autors izstrādā programmu izmantojot Eclipse IDE. Programmas saskarnē lietotās ikonas un attēlus autors pats izveidoja Gimp2 vizuālās rediģēšanas programmā. Programma sastāv no 13 klasēm, 10 konstruktoriem, 53 metodēm un 1212 koda līnijām.

Lai lietotājs varētu darboties ar programmu, tiek veidots vizuālā saskarne, kas sastāv no “javax.swing” un “java.awt” objektiem, kā “JFrame”, “JLabel” un “JLayeredPane”. Lai mazinātu koda rindu skaitu, tiek veidota “JavaLabel” klase (Skat. Pielikums IV), kas optimizē JLabel objektu konstruēšanu, un “JavaLayeredPane” klase (Skat. Pielikums III), optimizē “JLayeredPane” objektu konstruēšanu. Navigācija starp logiem ir iespējama spiežot uz programmas logā redzamajām pogām. Pogu darbību nodrošina “Mouse” klases konstruktors, kurā tam padotajiem “JLabel” objektiem tiek piešķirta “MouseListener” saskarne. Koda optimizācijas dēļ JLabel objekti, kas tiek lietoti kā pogas, tiek konstruēti izmantojot “ProgramButton” klasi (Skat. Pielikums II).

Pirms testa uzsākšanas lietotājam ir iespēja izvēlēties testa grūtības pakāpi, ziņu portāla saiti un ziņu raksta numuru. Grūtības pakāpes izvēle ir domāta, lai testu varētu pildīt gan lietotāji ar lēnu rakstīšanas ātrumu, gan arī lietotāji ar advancētu rakstīšanas ātrumu. Ziņu portālu izvēle ir limitēta uz ziņu izvēlnes logā redzamajām pogām: “Delfi(sabiedrība)”, “LSM(populārās)”, “Delfi(visas)”. Ir arī iespēja lietot teksta failu “//tempText.txt”. Izvēlētais ziņu portāls vai fails nosaka no kurienes tiks nolasīti vārdi, kas tiks lietoti ātrrakstīšanas teksta laikā. Ziņu raksta numura izvēle ir ierobežota ar nolasīto rakstu skaitu un nosaka kurš ziņu raksta apraksts tiks izmantots testā.

Lai nolasītu ziņu rakstus, tika izmantots RSS parsētāja kods, ko darba autors ieguva no pamācību saites par RRS parsētājiem (Vogel, 2016). Iegūtais kods tika samazināts, lai paliktu tikai pats vajadzīgais programmai, un tad modificēts pēc programmas vajadzībām.

Pēc iestatījumu veikšanas, lietotājs var uzsākt testu nospiežot programmas testa sagataves logā pogu “Play”. Tests sākas ar laika atskaiti, pēc kuras no programmas loga augšpuses sāk krist “FallingWord” objekti (Skat. Pielikums V), kas katrs satur savu vārdu, kas ir ņemts no izvēlētā ziņu raksta. Krītošo vārdu secība atbilst vārdu secībai ziņu raksta aprakstā. Lietotājs šos vārdus uzraksta izmantojot klaviatūru un tad ievada, uz klaviatūras nospiežot pogu “enter”.

Kad visi vārdi ir uzrakstīti vai nokrituši līdz loga apakšai, parādās rezultātu logs, kurā ir redzams teksta logs ar saiti uz pilno ziņu rakstu, lietotāja sasniegtais vārdu rakstīšanas ātrums sekundē un minūtē, rakstzīmju rakstīšanas ātrums sekundēs un minūtēs, pieļauto kļūdu skaits un uzrakstīto vārdu skaits. Lietotājs tad var ievadīt savu lietotājvārdu, kas, kopā ar rakstīšanas ātrumiem, tiek saglabāts teksta failā un uzrādīts top rezultātu tabulā.

Pēc lietotājvārda ievadīšanas parādās pogas “PREVIOUS ARTICLE” un “NEXT ARTICLE”, kuras domātas, lai uzreiz varētu uzsākt jaunu testu ar vai nu nākamo vai iepriekšējo pēc kārtas esošo ziņu rakstu. Ja tests tika veikts izmantojot vārdus no teksta faila, tad pēc lietotājvārda ierakstīšanas, teksta ievades lauks pazudīs, un būs iespēja tikai restartēt testu vai iet atpakaļ uz testa sagataves logu.

# **Programmas funkciju demonstrēšana**

## **Navigācija.**

Spiežot programmas logā redzamās pogas, lietotājs var pārvietoties starp logiem:

* + 1. **Galvenās izvēlnes logs:**

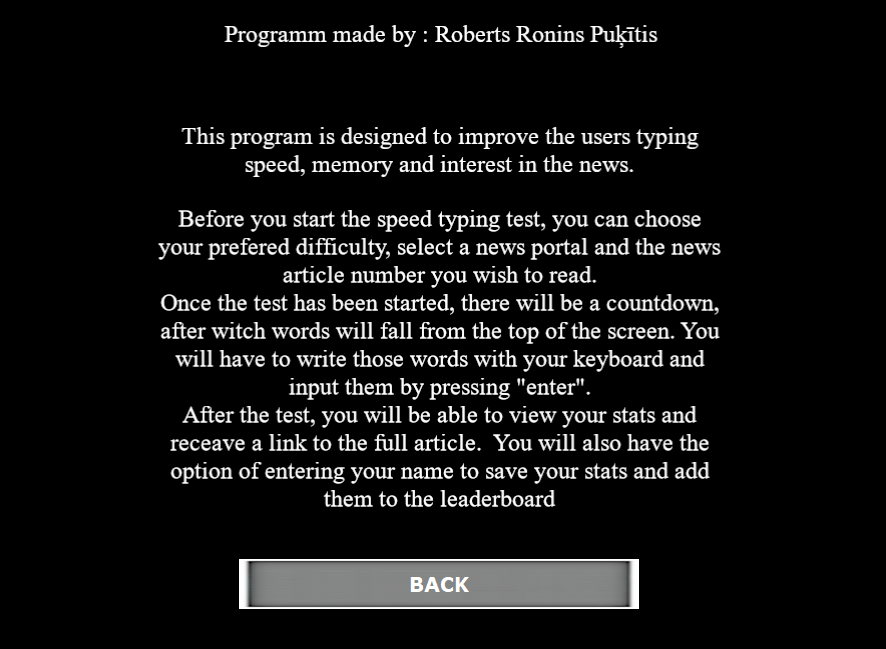
Pirmais logs, kas atveras, startējot programmu. No galvenās izvēlnes var tikt uz informācijas logu, nospiežot pogu “INFO PAGE” vai testa sagataves logu, nospiežot uz “READ ARTICLES” pogu, vai arī beigt darbu, nospiežot “exit” pogu.



Attēls 9. Galvenās izvēlnes logs

* + 1. **Informācijas logs:**

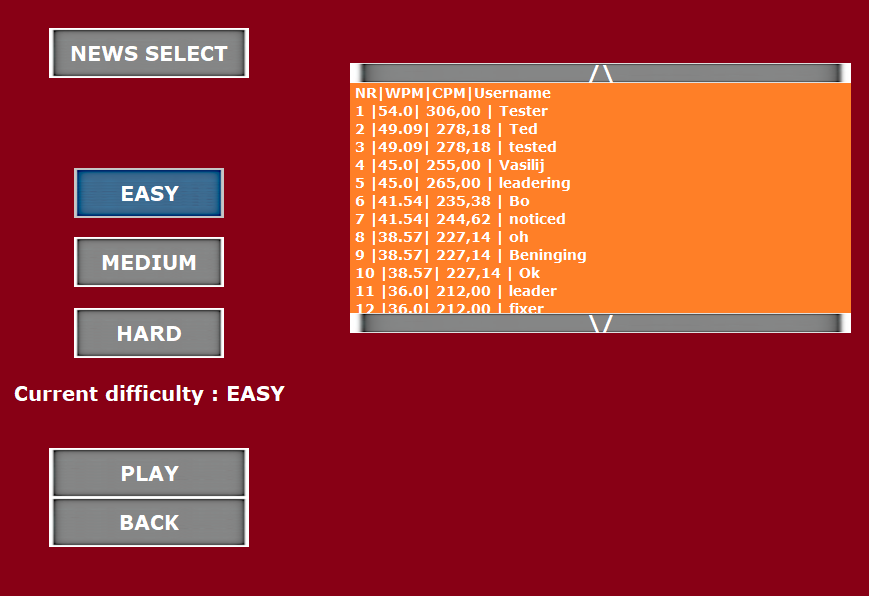
Lietotājs var izlasīt programmas paskaidrojošo tekstu un izlasīt vajadzīgās pamata zināšanas, lai veiktu ātrrakstīšanas testu. Loga apakšā ir redzamās “BACK” pogas nospiešana atvērs galvenās izvēlnes logu.



Attēls 10. Informācijas logs

* + 1. **Testa sagataves logs**

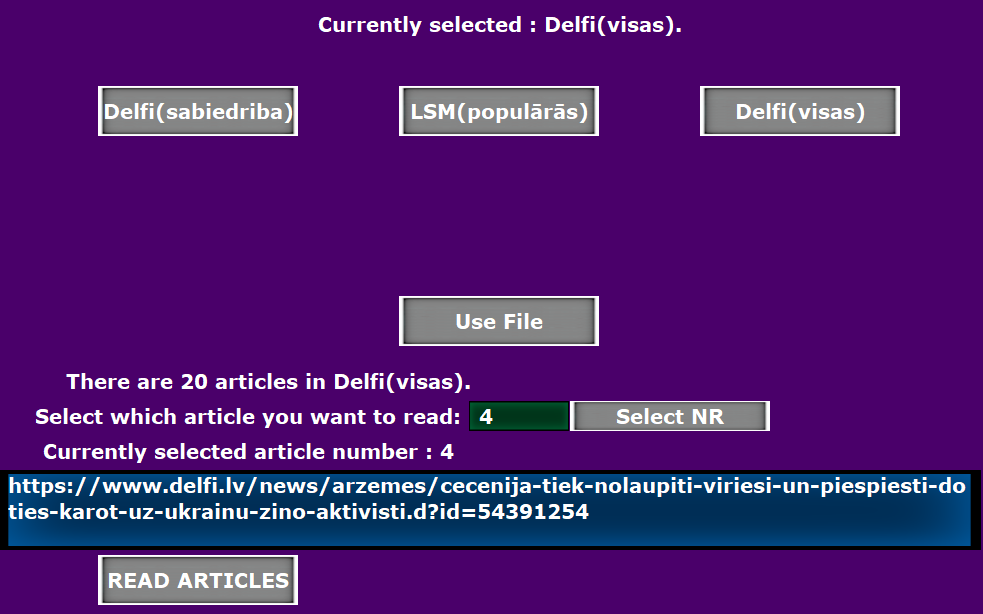
Testa sagataves logā ir iespējams apskatīt top rezultātu tabulu. Kustība uz augšu un uz leju pa top rezultātu tabulu ir iespējama spiežot “/\” un “\/” pogas. Lietotājs var izvēlēties grūtības pakāpi, spiežot vienu no trīs grūtības pakāpju pogām – “EASY”, “MEDIUM”, “HARD”. Lai izvēlētos ziņu ierakstu opcijas, ir iespēja lietotājs var nospiest “NEWS SELECT” pogu, lai dotos uz ziņu izvēlnes logu. Lai startētu ātrrakstīšanas testu, ir jānospiež “PLAY” poga. “BACK” pogas nospiešana atvērs galvenās izvēlnes logu.



Attēls 11. Testa sagataves logs.

* + 1. **Ziņu izvēlnes logs**

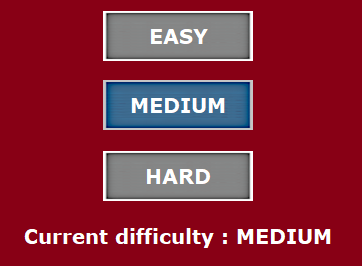
Piedāvā lietotājam izvēlēties starp pieejamajiem ziņu portāliem un to ziņu rakstiem, vai izmantot failu, lai programma iegūtu vārdus priekš ātrrakstīšanas testa.



Attēls 12. Ziņu izvēlnes logs.

## **Grūtības pakāpes izvēle.**

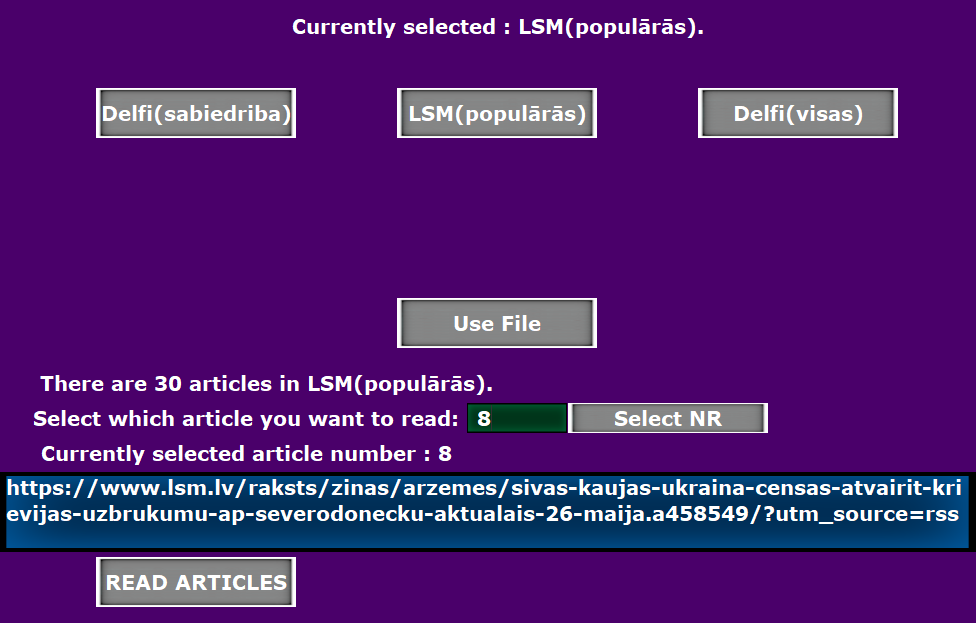
Testa izvēlnes logā, lietotājs var izvēlēties starp trīs grūtības pakāpēm : “EASY”(Viegla), “MEDIUM”(vidēja), “HARD”(grūta). Nospiežot uz kādu no pogām, tās ikona nomainās, lai rādītos kā atzīmēta, un zem pogām paskaidrojošais teksts uzrāda kāda grūtības pakāpe ir izvēlēta. Grūtības pakāpe nosaka testā krītošo vārdu ātrumu un to parādīšanās intervāla īsumu.



Attēls 13. Grūtības pakāpju pogas un grūtības paziņojuma teksts.

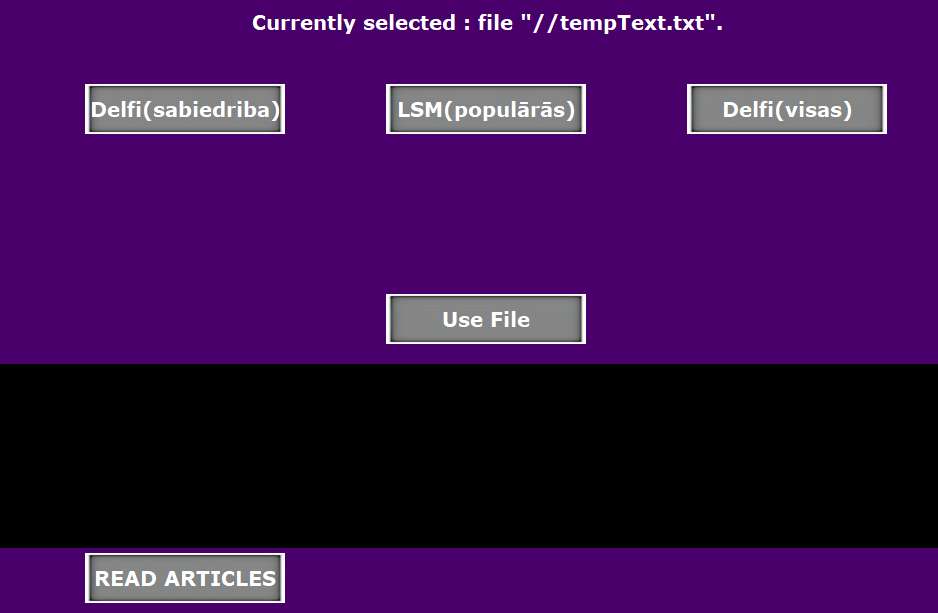
## **Ziņu portāla izvēle.**

Ziņu izvēlnes logā ir iespēja izvēlēties kādu no piedāvātajiem ziņu portāliem, nospiežot respektīvo pogu. Ekrāna augšpusē ir redzams paskaidrojošais teksts, kas uzrāda kāds ziņu portāls ir izvēlēts. Zem ziņu portālu pogām ir teksts, kas uzrāda cik ziņu raksti ir tikuši nolasīti no izvēlētā ziņu portāla. Nomainot ziņu portālu arī mainās ziņu raksta saite, uzņemot iestatītā numura ziņu rakstu. Izvēloties ziņu portālu, RSS plūsmas parsētājs nolasa ziņu rakstus no izvēlētā portāla, un iegūtie dati par ziņu rakstiem tiek saglabāti programmā.



Attēls 14. Ziņu portāla izvēlnes logs, ar izvēlētu ziņu portālu.

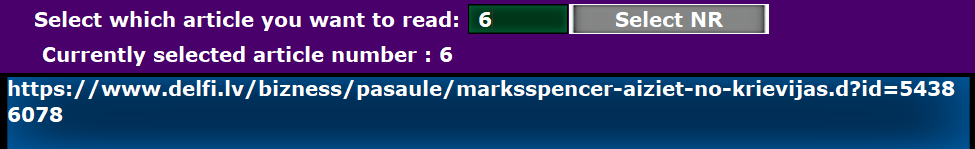
Ja lietotājam nav internets, vai tiek nospiesta “Use File” poga, apakšējie informatīvie lauki tiks aizklāti un programma izmantos teksta failu “tempText.txt”, lai iegūtu vārdus ātrrakstīšanas testam.



Attēls 15. Ziņu portāla izvēlnes logs, ar izvēlētu faila izmantošanu.

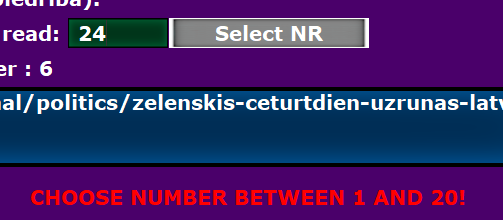
## **Ziņu raksta izvēle.**

Ziņu izvēlnes logā ir ziņu raksta numura izvēle. Ievadot derīgu numuru un nospiežot “Select NR” pogu, tiek izvēlēts attiecīgais ziņu raksts. Pēc izvēles veikšanas, zem numura izvēles redzams teksts, kas uzrāda kurš ziņu raksta numurs ir izvēlēts, un zem tā ir redzams teksta lauks ar saiti uz pilno ziņu rakstu.



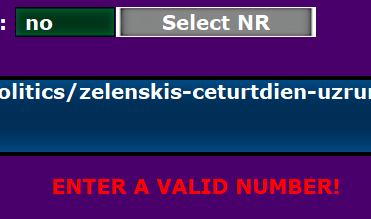
Attēls 16. Ziņu raksta izvēle.

Ja ievadītais numurs ir ārpus atļautajām robežām, uzrādīsies kļūdas paziņojums, kas uzrāda numuru robežas.



Attēls 17. Ziņu raksta nepareiza numura ievades kļūdas paziņojums.

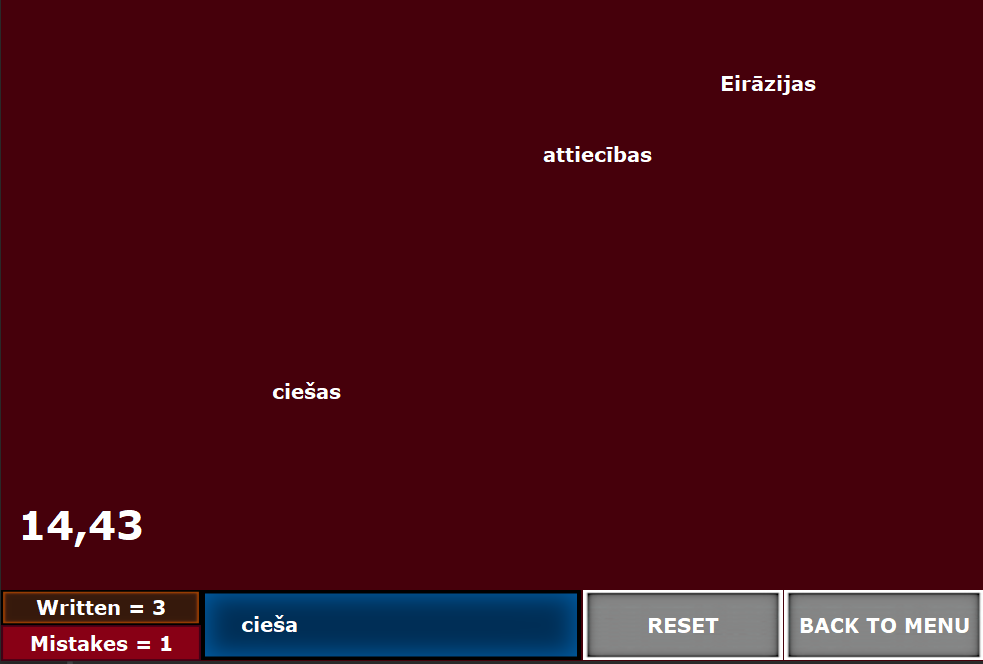
Ja skaitļa vietā tiek ievadīts teksts, uzrādīsies kļūdas paziņojums.



Attēls 18. Ziņu raksta izvēles kļūdainas ievades kļūdas paziņojums.

## **Testa logs.**

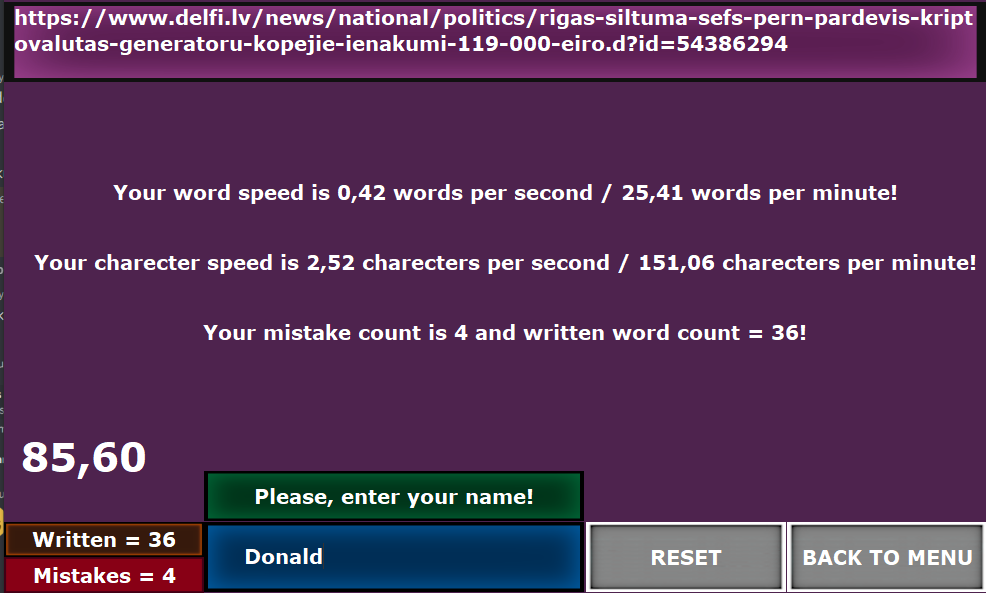
Testa logā ir redzama apakšējā josla, kurā redzams rakstīto vārdu skaits, pieļauto kļūdu skaits, lietotāja ievades teksta lauks, poga “RESET”, lai restartētu testu, un poga “BACK TO MENU”, lai atgriestos uz testa sagataves logu. Pēc atskaites, loga kreisajā pusē var redzēt laiku sekundēs, kopš testa uzsākšanas. Testa laikā no loga augšpuses krīt objekti ar secībā nolasītiem ziņu raksta vārdiem. Pēc krītošā vārda ierakstīšanas un ievadīšanas, tas pazūd no ekrāna.



Attēls 19. Testa logs testa darbības laikā.

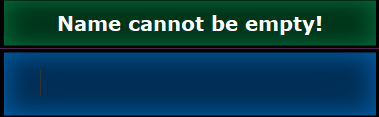
## **Rezultātu logs.**

Pēc testa pabeigšanas tiek uzrādīti lietotāja sasniegtie ātrumi, kļūdu skaits, uzrakstīto vārdu skaits un teksta logs ar saiti uz pilno ziņu rakstu. Teksta ievades logā lietotājam ir iespēja ievadīt savu vārdu, lai vārds un iegūtie ātrumi tiktu saglabāti top rezultātu tabulā.



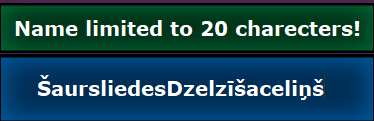
Attēls 20. Testa rezultātu logs.

Ja lietotājs nospiež “enter” pirms uzrakstījis savu lietotājvārdu, uzrādās kļūdas paziņojums par to, ka nav ierakstīts vārds.



Attēls 21. Lietotājvārda ievades kļūdas paziņojums par tukšu ievades lauku.

Ja lietotājs ievada vārdu, kas ir garāks par 20 rakstzīmēm, tad uzrādās kļūdas paziņojums par maksimālo rakstzīmju skaitu.



Attēls 22. Lietotājvārda ievades kļūdas paziņojums par vārda rakstzīmju skaitu.

Pēc lietotājvārda ierakstīšanas parādās pogas “PREVIOUS ARTICLE” un “NEXT ARTICLE”, kuras domātas, lai uzreiz varētu uzsākt jaunu testu ar vai nu nākamo vai iepriekšējo pēc kārtas esošo ziņu rakstu.



Attēls 23. Rezultātu logā esošās pogas, pēc lietotājvārda ievadīšanas, kad testā ir ticis lietots ziņu raksts.

Ja tests tika veikts izmantojot vārdus no teksta faila, tad pēc lietotājvārda ierakstīšanas, teksta ievades lauks pazudīs, un būs iespēja tikai restartēt testu vai iet atpakaļ uz testa sagataves logu.



Attēls 24. Rezultātu logā esošās pogas, pēc lietotājvārda ievadīšanas, kad testā ir ticis lietots teksta fails.

# **DARBA EKONOMISKAIS PAMATOJUMS**

## **Izmaksas**

Par darba izstrādē lietotajām programmām, Eclipse IDE un Gimp2, nav nekādu izmaksu, jo tās ir pieejamas bezmaksas.

## **Potenciālie ienākumi**

1. Reklāmu ievietošana programmā.
2. Iespēja iegādāties “Premium” versiju, kurā nebūtu reklāmu un tiktu piedāvātas jaunas funkcijas un plašāka informācija par lietotāja sasniegumiem testu pildīšanā.

# **SECINĀJUMI**

Gada projekta darba gaitā tika veidota programma, kurā ir iespējams veikt ātrrakstīšanas testu, kas izmanto vārdus vai nu no nolasītiem ziņu rakstu aprakstiem, vai no faila. Programmas veidošanas laikā, darba autors uzlaboja savas prasmes optimizēt java kodu un uzzināja kā izmantot RSS parsētāju, lai nolasītu datus no internetā pieejamiem XML failiem.

# **LITERATŪRA**

1. Hayes, A. (2022, 24. Janvāris). *The Human Attention Span [INFOGRAPHIC]*. Wyzowl. <https://www.wyzowl.com/human-attention-span/>
2. Noorman, S. (2018, 20. Septembris). Words affect visual perception by activating object shape representations. Nature. <https://www.nature.com/articles/s41598-018-32483-2>
3. Sikowski, S. (2019, 23. Decembris). *5 Tips for Improving Your Typing Speed & Accuracy*. Herzing University. <https://www.herzing.edu/blog/5-tips-improving-your-typing-speed-accuracy>
4. ABM College. (2015, 6. Marts). The Importance of Typing Speed. https://www.abmcollege.com/blog/the-importance-of-typing-speed
5. Key Hero. (2022). Typing Test · Create your own test. https://www.keyhero.com/custom-typing-test/
6. Typing. (2022). Take a Free Typing Test. https://www.typing.com/student/tests
7. Ratatype. (2022). *Average typing speed infographic — Ratatype*. Ratatype. <https://www.ratatype.com/learn/average-typing-speed/>
8. Herzing University. (2019, December 23). *5 Tips for Improving Your Typing Speed & Accuracy*. <https://www.herzing.edu/blog/5-tips-improving-your-typing-speed-accuracy>
9. Gordon, E. (2019, January 25). *Benefits of having great posture at work and how to improve bad posture*. Quill.Com Blog. <https://www.quill.com/blog/lifestyle/benefits-of-having-great-posture-at-work-and-how-to-improve-bad-posture.html>
10. Hartman, J. (2022, 16. Aprilis). *What is Java? Definition, Meaning & Features of Java Platforms*. Guru99. <https://www.guru99.com/java-platform.html>
11. Singh, V. (2022, 4. Maijs). *Best Java IDE 2022 | Most Popular Java IDE for Coding*. Hackr.Io. <https://hackr.io/blog/best-java-ides>
12. Johnson, D. (2021, 30. Aprīlis). *A guide to using RSS feeds, the files that contain real-time updates from websites*. Business Insider. <https://www.businessinsider.com/what-is-rss-feed?international=true&r=US&IR=T>
13. Vogel, L. (2016, 26. Septembris). RSS feeds with Java - Tutorial. Vogella. https://www.vogella.com/tutorials/RSSFeed/article.html

# **PIELIKUMI**

1. **“setupScreen” metodes pirmkods**

**static** JFrame setupScreen(JFrame frame, int frameWidth, int frameHeight) {

frame = **new** JFrame("News Article Speed Typing");

frame.setSize(frameWidth, frameHeight);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setLayout(**null**);

frame.setResizable(**false**);

frame.getContentPane().setBackground(Color.black);

frame.setIconImage(**new** ImageIcon("logo.png").getImage());

frame.setLocationRelativeTo(**null**);

frame.setVisible(**true**);

**return** frame;

}

**“JavaLayeredPane” klases pirmkods.**

**package** inputClasses;

**import** java.util.HashMap;

**import** javax.swing.**\***;

**public** **class** JavaLayeredPane **extends** JLayeredPane {

**private** **static** **final** long serialVersionUID = 1L;

**public** **String** name;

int xStart, yStart;

**public** boolean selected;

**public** JavaLayeredPane(**String** name, **Object** location, int xStart, int yStart, int width, int height, **HashMap**<**String**, JavaLayeredPane> list,

**Integer** layer) {

**super**();

**this**.name = name;

**this**.xStart = xStart;

**this**.yStart = yStart;

**this**.setBounds(xStart, yStart, width, height);

**this**.selected = **false**;

**try** {

((JFrame) location).add(**this**);

} **catch** (**Exception** e) {

((JavaLayeredPane) location).add(**this**, layer);

}

list.put(name, **this**);

}

**public** void setSlidePoint(int x, int y) { // SETS LOCATION TO WHERE THE pane HAS TO SLIDE TO

**this**.xStart = x;

**this**.yStart = y;

}

**public** void setCoordinates() { // sets location of the pane

**this**.setLocation(moveScreen(**this**.getX(), **this**.xStart), moveScreen(**this**.getY(), **this**.yStart));

}

**public** void checkMovement() { // checks if pane is in corect possition

**if** (**this**.selected) {

**this**.setLocation(moveScreen(**this**.getX(), 0), moveScreen(**this**.getY(), 0));

} **else** {

**this**.setLocation(moveScreen(**this**.getX(), **this**.xStart), moveScreen(**this**.getY(), **this**.yStart));

}

}

**public** int moveScreen(int currentPos, int destinationPos) { // MOVES pane

**if** (currentPos != destinationPos) {

int direction = -1;

**if** (currentPos - destinationPos < 0) {

direction = 1;

}

**if** ((currentPos > destinationPos) == (direction == -1)) {

currentPos += direction \* (**Math**.abs((currentPos - destinationPos) / 10) + 1);

}

}

**return** currentPos;

}

}

1. **“ProgramButton” klases pirmkods.**

**package** inputClasses;

**import** java.awt.Color;

**import** java.awt.Font;

**import** java.awt.Image;

**import** java.util.HashMap;

**import** javax.swing.ImageIcon;

**import** javax.swing.JLabel;

**import** newsArticleSpeedTyping.**\***;

**public** **class** ProgramButton **extends** JLabel {

**private** **static** **final** long serialVersionUID = 1L;

**public** **static** **String** routeButtons = MainMenu.route + "//Buttons//";

**static** int fWidth, fHeight;

Font font = JavaLabel.font;

**String** name;

boolean inZone;

**public** boolean selected;

ImageIcon[] buttonIcon = **new** ImageIcon[3];

**public** **static** void setSizing(int frameWidth, int frameHeight) {

fWidth = frameWidth;

fHeight = frameHeight;

}

**public** ProgramButton(**String** name, **Object** location, double x, double y, int width, int height,

**HashMap**<**String**, ProgramButton> list) {

**super**(name);

**this**.buttonIcon[0] = **new** ImageIcon(**new** ImageIcon(routeButtons + "button.png").getImage()

.getScaledInstance(fWidth \* width / 1000, fHeight \* height / 700, Image.SCALE\_SMOOTH));

**this**.buttonIcon[1] = **new** ImageIcon(**new** ImageIcon(routeButtons + "buttonOn.png").getImage()

.getScaledInstance(fWidth \* width / 1000, fHeight \* height / 700, Image.SCALE\_SMOOTH));

**this**.buttonIcon[2] = **new** ImageIcon(**new** ImageIcon(routeButtons + "buttonPressed.png").getImage()

.getScaledInstance(fWidth \* width / 1000, fHeight \* height / 700, Image.SCALE\_SMOOTH));

**this**.name = name;

**this**.setForeground(Color.white);

**this**.setHorizontalTextPosition(JLabel.CENTER);

**this**.setFont(font);

**this**.setIcon(buttonIcon[0]);

**this**.inZone = **false**;

**this**.selected = **false**;

**this**.setBounds((int) (fWidth \* x / 1000), (int) (fHeight \* y / 700), (int) (fWidth \* width / 1000),

(int) (fHeight \* height / 700));

**try** {

((JavaLayeredPane) location).add(**this**, (**Integer**) 20);

} **catch** (**Exception** e) {

((JLabel) location).add(**this**);

}

**new** Mouse(**this**);

list.put(name, **this**);

}

**public** void buttonIconSwitch(int iconNumber, boolean inZone) {

**this**.setIcon(buttonIcon[iconNumber]);

**this**.inZone = inZone;

}

}

1. **“switchNews” metodes pirmkods**

**public** **static** void switchNews(String link, String newsSiteName) {

labels.get("blackBar").setVisible(**false**);

menuButtons.get("Select NR").setVisible(**true**);

labels.get("NRerrText").setText("");

**try** {

**new** RssData(link);

useFile = **false**;

**try** {

articleLink.setText(RssData.linkData.get(articleIndex));

} **catch** (**Exception** e) { // if new website doesn't have an article at selected index, index is reset.

articleIndex = 0;

labels.get("CurrentArticleNR").setText("Currently selected article number : " + (articleIndex + 1));

articleLink.setText(RssData.linkData.get(articleIndex));

}

inputFieldArticleNR.setText((articleIndex + 1) + "");

selectedNewsSiteName = newsSiteName;

articleCount = RssData.descriptionData.size();

labels.get("currentNews").setText("Currently selected : " + selectedNewsSiteName + ".");

labels.get("articleNR").setText("There are " + articleCount + " articles in " + selectedNewsSiteName + ".");

} **catch** (**Exception** e) {

switchToFile();

}

}

1. **“RssData” klases pirmkods**

**package** rss;

**import** java.util.ArrayList;

**import** newsArticleSpeedTyping.MainMenu;

**public** **class** RssData {

**public** **static** **ArrayList**<**String**> descriptionData = **new** **ArrayList**<**String**>();

**public** **static** **ArrayList**<**String**> linkData = **new** **ArrayList**<**String**>();

**public** RssData(**String** site) {

descriptionData.clear();

linkData.clear();

RSSFeedParser parser = **new** RSSFeedParser(site);

Feed feed = parser.readFeed();

**for** (FeedMessage message : feed.getMessages()) {

descriptionData.add(message.getDescription().replace("&quot;", "\""));

linkData.add(message.getLink());

}

MainMenu.articleCount = descriptionData.size();

}

**public** **static** **String** getDescription(int index) {

**return** descriptionData.get(index);

}

}

1. **“FallingWord” klases pirmkods.**

**package** newsArticleSpeedTyping;

**import** java.awt.Color;

**import** java.awt.Font;

**import** java.awt.Image;

**import** java.util.ArrayList;

**import** java.util.Random;

**import** javax.swing.ImageIcon;

**import** javax.swing.JLabel;

**import** inputClasses.JavaLabel;

**public** **class** FallingWord **extends** JLabel {

**private** **static** **final** long serialVersionUID = 1L;

**static** **Random** random = **new** **Random**();

**static** int fWidth, fHeight;

**public** **static** void setSizing(int frameWidth, int frameHeight) {

fWidth = frameWidth;

fHeight = frameHeight;

}

**String** word;

double y;

Font font = JavaLabel.font;

**public** FallingWord(int x, int y, int width, int height, **String** name, **ArrayList**<FallingWord> list, int layer,

**String** route) {

**super**(**new** ImageIcon(**new** ImageIcon(route + name + ".png").getImage().getScaledInstance(fWidth \* width / 1000,

fHeight \* width / 1000, Image.SCALE\_SMOOTH)), JLabel.CENTER);

GameScreen.layers.get("game").add(**this**, (**Integer**) layer);

**this**.y = -fHeight \* height / 2000;

**this**.word = name;

**this**.setHorizontalTextPosition(JLabel.CENTER);

**this**.setForeground(Color.white);

**this**.setFont(font);

**this**.setText(name);

**this**.setBounds((random.nextInt(fWidth - fWidth \* width / 1000) + 1), y, fWidth \* width / 1000,

fHeight \* height / 1000);

list.add(**this**);

}

**public** **String** getWord() {

**return** **this**.word;

}

**public** boolean checkForDespawn() {

**if** (**this**.getY() > fHeight) { // if word has fallen out of the screen, it is despawned

GameScreen.layers.get("game").remove(**this**);

**return** **true**;

}

**return** **false**;

}

**public** int despawn(**ArrayList**<FallingWord> list) { // removes word from list and screen

**this**.setVisible(**false**);

GameScreen.layers.get("game").remove(**this**);

list.remove(**this**);

**return** 1;

}

**public** void setPosition(double moveY) {

// moves the object by a distance, depending on speed and word length

**if** (**this**.word.length() < 3) {

**this**.y += moveY / 3;

} **else** {

**this**.y += moveY / **this**.word.length();

}

**this**.setLocation(**this**.getX(), (int) **this**.y);

}

}

1. **“checkInput” metodes pirmkods**

**public** **static** void checkInput() {

**String** word = textField.getText().replace("\n", ""); // removes \n from String

**if** (!end) {

matchWord(word); // checks if input matches any falling word

textField.setText(""); // resets text field

} **else** { // when the game has ended

**if** (checkNameValidity(word)) {

MainMenu.updateLeaderboard(speed[1] + " | " + speed[3] + " | " + word + "\n",

**new** **File**(MainMenu.leaderboardRoute));

textField.setEditable(**false**);

**if** (!MainMenu.useFile) {

labels.get("writtenWords").setVisible(**false**);

labels.get("mistakesBox").setVisible(**false**);

**new** ProgramButton("PREVIOUS ARTICLE", labels.get("bottomLabel"), 0, 0, 300, 70, gameButtons);

**new** ProgramButton("NEXT ARTICLE", labels.get("bottomLabel"), fWidth \* 0.3, 0, 300, 70, gameButtons);

}

labels.get("nameInfo").setVisible(**false**);

textField.setVisible(**false**);

labels.get("textBox").setVisible(**false**);

}

}

}

1. **“JavaLabel” klases pirmkods**

**package** inputClasses;

**import** java.awt.Color;

**import** java.awt.Font;

**import** java.awt.Image;

**import** java.util.HashMap;

**import** javax.swing.**\***;

**public** **class** JavaLabel **extends** JLabel {

**private** **static** **final** long serialVersionUID = 1L;

**static** int fWidth, fHeight;

**public** **static** Font font;

**public** **static** void setSizing(int frameWidth, int frameHeight) {

fWidth = frameWidth;

fHeight = frameHeight;

font = **new** Font("Verdana", Font.BOLD, (int) (fWidth \* 0.02));

}

**public** JavaLabel(**String** pngName, **Object** location, double x, double y, double width, double height,

**HashMap**<**String**, JLabel> list, **Integer** layer, **String** route) {// CONSTRUCTS JLABEL

**super**(**new** ImageIcon(**new** ImageIcon(route + pngName + ".png").getImage()

.getScaledInstance((int) (width \* fWidth / 1000), (int) (height \* fHeight / 700), Image.SCALE\_SMOOTH)),

JLabel.CENTER);

**this**.setBounds((int) (fWidth \* x / 1000), (int) (fHeight \* y / 700), (int) (fWidth \* width / 1000),

(int) (fHeight \* height / 700));

**try** {

((JavaLayeredPane) location).add(**this**, layer);

} **catch** (**Exception** e) {

((JLabel) location).add(**this**);

}

**this**.setHorizontalTextPosition(JLabel.CENTER);

**this**.setForeground(Color.white);

**this**.setFont(**new** Font("Verdana", Font.BOLD, fWidth \* 20 / 1000));

list.put(pngName, **this**);

}

}