# 4. PROGRAMMATŪRAS TESTĒŠANA

## 4.1 Testēšanas gaita

Testēšana notika laika posmā no. Testēšana tika veikta katram programmatūras modulim [skat. 2.1.4 nodaļu].

Kompilatora modulī bija svarīgi kļūdu fiksēšanas pārbaude jaunkodā – vai tiek fiksētas kļūdas, ja tādas ir un vai nekļūdainos jaunkodos nekas netiek fiksēts. Testēšana tiek padots ievadfails ar jaunkodu un rezultāta tiek izveidots izvadfails ar visām kļūdām. Gadījumos, kad tiek padots nekļūdains kods, tiek izveidots tukšs izvadfails. Ja ir kādas prasības, kuras var apvienot vienā testpiemēra, jo tie savā starpā vai nu nav saistīti, vai tiem ir jādod vienu un to pašu rezultātu, tad tās apvieno. Testpiemēros visi elementi ir savā rinda, lai pārbaudītu, vai tiek pareiza pateikts, kura rindā ir kļūda un kurā rinda ir jāmeklē jau esošs vārds. Kompilatora testē divas lietas:

* Sintaksi
* Funkcionalitāte

Ģeneratora modulī tiek testēts tas, vai ir pareizi ģenerēts kods. Šajā testēšanā visos testpiemēros tiek padots pareizs jaunkods.

Ģenerēto kodu, WebMemory un RemoteWebCalls moduļus testē kopīgi, jo tie ir savā starpā saistīti. Tiek izvēlēts viens starpkods, kurā ir pēc iespējas vairāk dažādu scenāriju un šiem scenārijiem iziet cauri.

LocalWebCalls modulis tiek testēts, kad šī sistēma tiek integrēta WebAppOS. Ir svarīgi pārliecināties, ka viss, ko testējam RemoteWebCalls darbojas arī ar LocalWebCalls.

## 4.2 Testpiemēru apraksts

### 4.2.1 Kompilatora sintaktiskā testēšana

Šajā testēšanas nodaļā koncentrējamies vairāk uz to, vai kods ir sintaktiski pareizs. Tiek skatīts, vai kodā ir izlaistas daļas, pareizi lietoti atslēgvārdi.

#### 4.2.1.1 Bloku testēšana

Izmantojot nodaļā 2.1.1.1 aprakstīto bloka sintakse un nodaļā 2.1.5.2 aprakstītās prasības, secinam, ka dažādās bloku kombinācijās programma uzvedas citādāk, tāpēc, ir nepieciešams izskatīt visas kombinācijas. Ir pieejami trīs dažādi bloku tipa gadījumi – *class, association* un nebloka tips, kas šajā gadījumā ir vai nu datu tips vai aizsardzība (var ņemt jebkuru vienu vērtību; ņemsim *Integer*). Blokam var būt divi dažādi ķermeņi – klases vai asociācijas. Vēl jāpiebilst, ka var nebūt tipa vai ķermeņa, tādējādi iznāk 4\*3=12 kombinācijas. Visās kļūdās trūkstošā/kļūdainā elementa rinda sakrīt ar esošā elementa rindu.

|  |  |  |  |
| --- | --- | --- | --- |
| Nr. | Apraksts | Sagaidāmais rezultāts | Funkcija |
| 1. | Tiek padots tukšs jaunkoda fails. | Nav kļūdu |  |
| 2. | Tiek padota asociācijas definīcija bez atslēgvārda. | Kļūdas paziņojums   * “Trūkst atslēgas vārda ‘association’” |  |
| 3. | Tiek padota klases definīcija bez atslēgvārda. | Kļūdas paziņojums   * “Trūkst atslēgas vārda ‘class’” |  |
| 4. | Tiek padots atslēgvārds *class* bez bloka ķermeņa. | Kļūdas paziņojums   * “Trūkst klases ķermeņa” |  |
| 5. | Tiek padota asociācijas definīcija ar atslēgvārdu *class*. | Kļūdas paziņojums   * “Klasei ir dota asociācijas definīcija” |  |
| 6. | Tiek padota klases definīcija ar atslēgvārdu *class*. | Nav kļūdu |  |
| 7. | Tiek padots atslēgvārds *association* bez bloka definīcijas. | Kļūdas paziņojums   * “Trūkst asociācijas ķermeņa” |  |
| 8. | Tiek padota asociācijas definīcija ar atslēgvārdu *association*. | Nav kļūdu |  |
| 9. | Tiek padota klases definīcija ar atslēgvārdu *association*. | Kļūdas paziņojums   * “Asociācijai ir dota klases definīcija” |  |
| 10. | Tiek padots neatbalstīts bloka tips bez bloka ķermeņā. | Kļūdas paziņojums   * “’Integer’ nav bloka tips. Izmanrtojiet ‘*class*’ vai ‘*association*’!” |  |
| 11. | Tiek padots neatbalstīts bloka tips ar asociācijas definīciju. | Kļūdas paziņojums   * “’Integer’ nav bloka tips. Izmanrtojiet ‘*association*’!” |  |
| 12. | Tiek padots neatbalstīts bloka tips ar klases definīciju. | Kļūdas paziņojums   * “’Integer’ nav bloka tips. Izmanrtojiet ‘*class*’!” |  |

#### 4.2.1.2 Asociāciju testēšana

Šajā testēšanas nodaļā tiek pieņemts, ka asociācijas klases jau ir definētas, šajā gadījumā, *sourceClass* ir avota klase un *targetClass* ir mērķa klase. Tā kā avots un mērķis ir neatkarīgi viens no otra šajā testēšanā, tad to atsevišķos elementus (lomas vārdu, klasi un kolu) varam testēt vienlaicīgi. Ir arī zināms, ka šo daļu testēšana ir līdzīga. Testpiemēros asociācija sākas ar 4. rindu, tas ir, vārds *association* atrodas 4. rinda un katrs nākamais elements ir sava rindā.

|  |  |  |  |
| --- | --- | --- | --- |
| Nr. | Apraksts | Sagaidāmais rezultāts | Funkcija |
| 1. | Tiek padota asociācija bez definīcijas iekavās. | Kļūdas paziņojums   * “5. rindā trūkst asociācijas definīcijas” |  |
| 2. | Tiek padota asociācija, kurai nav definēts avots. | Kļūdas paziņojums   * “5. rindā trūkst asociācijas avota definīcijas” |  |
| 3. | Tiek padota asociācija, kurai nav bultu. | Kļūdas paziņojums   * “8. rindā trūkst bultu asociācijas definīcijā” |  |
| 4. | Tiek padota asociācija, kurai nav definēts mērķis. | Kļūdas paziņojums   * “9. rindā trūkst asociācijas mērķa definīcijas” |  |
| 5. | Tiek padota asociācija, kurai nav bultu un nav definēts mērķis. | Kļūdas paziņojumi   * “8. rindā trūkst bultas asociācijas definīcijā” * ”8. rindā trūkst asociācijas mērķa definīcijas” |  |
| 6. | Tiek padota asociācija, kurā avotam/mērķim nav doti koli. | Kļūdas paziņojumi   * “6. rindā trūkst ‘:’ avota definīcijā” * “9. rindā trūkst ‘:’ mērķa definīcijā” |  |
| 7. | Tiek padota asociācija, kurā avotam/mērķim nav dotas klases. | Kļūdas paziņojumi   * “7. rindā trūkst avota klases” * “9. rindā trūkst ‘:’ mērķa definīcijā” * “9. rindā trūkst mērķa klases” |  |
| 8. | Tiek padota asociācija, kurā avotam/mērķim nav doti lomu vārdi. | Kļūdas paziņojumi   * “5. rindā trūkst avota lomas vārda” * “8. rindā trūkst mērķa lomas vārda” |  |
| 9. | Tiek padota asociācija, kurā avotam/mērķim nav doti lomu vārdi un nav bultu. | Kļūdas paziņojumi   * “5. rindā trūkst avota lomas vārda” * “7. rindā trūkst bultu asociācijas definīcijā” * “7. rindā trūkst mērķa lomas vārda” |  |

#### 4.2.1.3 Klašu testēšana

Šajā testēšanas nodaļā tiek pieņemts, ka virsklase jau ir definēta, šajā gadījumā, *superClass* ir virsklase. Visi testpiemēri sakas ar 1. rindu.

|  |  |  |  |
| --- | --- | --- | --- |
| Nr. | Apraksts | Sagaidāmais rezultāts | Funkcija |
| 1. | Tiek padota klase bez ķermeņa. | Kļūdas paziņojums   * “2. rindā trūkst klases ķermeņa” |  |
| 2. | Tiek padota klase bez galvas. | Kļūdas paziņojumi   * “1. rindā trūkst klases galvas” * “4. rindā trūkst atslēgvārda *class*” * “4. rindā trūkst klases galvas” |  |
| 3. | Tiek padota klases bez klases vārda un ar kolu bez virsklases varda. Virsklase jau pirms tam ir definēta. | Kļūdas paziņojumi   * “3. rindā trūkst klases vārda” * “4. rindā trūkst virsklases vārda” * “7. rindā trūkst atslēgvārda *class*” * “7. rindā trūkst klases vārda” * “11. rindā trūkst atslēgvārda *class*” * “11. rindā trūkst klases vārda” |  |
| 4. | Tiek padota pareizi definēta klase ar virsklasi. Pareizi defineta klase bez virsklases jau tika apskatīta bloku testēsanā. Virsklase jau pirms tam ir definēta. | Nav kļūdu |  |

#### 4.2.1.4 Klases atribūtu testēšana

Šajā nodaļā koncentrējamies tikai uz atribūtu testēšanu, un norādām, kādi atribūti tiek padoti. Semikoli tiek padoti visos testpiemēros, izņemot 10., kurā tiek pārbaudīta semikola iztrūkuma kļūda. Visi testpiemēri sakas ar 1. rindu.

|  |  |  |  |
| --- | --- | --- | --- |
| Nr. | Apraksts | Sagaidāmais rezultāts | Funkcija |
| 1. | Tiek padots tukšs lauks jeb tikai semikols. | Nav kļūdu |  |
| 2. | Tiek padots atribūts, kuram ir tikai vārds. | Kļūdas paziņojums   * “3. rinda trūkst atribūta datu tipa” |  |
| 3. | Tiek padots atribūts, kuram ir tikai datu tips. | Kļūdas paziņojums   * “3. rindā trūkst atribūta vārda” |  |
| 4. | Tiek padots atribūts, kuram ir datu tips un vārds. | Nav kļūdu |  |
| 5. | Tiek padots atribūts, kuram ir tikai aizsardzība. | Kļūdas paziņojums   * “3. rindā trūkst atribūta datu tipa un vārda” |  |
| 6. | Tiek padots atribūts, kuram ir aizsardzība un vārds. | Kļūdas paziņojums   * “3. rindā trūkst atribūta datu tipa” |  |
| 7. | Tiek padots atribūts, kuram ir aizsardzība, datu tips un vārds. | Nav kļūdu |  |
| 8. | Tiek padots pareizi definēts atribūts, kuram datu tips ir bloka tips. | Kļūdas paziņojums   * “4. rindā tips *class* nav pareizs atribūta datu tips” |  |
| 9. | Tiek padots pareizi definēts atribūts, kuram datu tips ir *Void*. | Kļūdas paziņojums   * “4. rindā tips *Void* nav pareizs atribūta datu tips” |  |
| 10. | Tiek padoti vairāki lauki bez semikoliem. | Kļūdas paziņojumi   * “6. rindā trūkst semikola” * “9. rindā trūkst semikola” * “12. rindā trūkst semikola” * “17. rindā trūkst semikola” * “22. rindā trūkst semikola” |  |

#### 4.2.1.5 Klases metožu testēšana

Šajā nodaļā koncentrējamies tikai uz metožu testēšanu, un norādām, kādas metodes tiek padotas. Tā kā vairākas lietas šeit atkārtojas no atribūtu testēšanas, tad tās otro reizi nav nepieciešams. Argumentus un anotācijas testējam atsevišķi. Par metodes galvu tiek uzskatīta metodes aizsardzība, datu tips un vārds kopumā. Visi testpiemēri sakas ar 1. rindu.

|  |  |  |  |
| --- | --- | --- | --- |
| Nr. | Apraksts | Sagaidāmais rezultāts | Funkcija |
| 1. | Tiek padota metode ar URL un ne URL anotāciju, datu tipu, vārdu un iekavām. | Nav kļūdu |  |
| 2. | Tiek padota metode ar URL anotāciju, aizsardzību datu tipu, vārdu un iekavām. | Nav kļūdu |  |
| 3. | Tiek padota metode bez iekavām. | Kļūdas paziņojums   * “6. rindā trūkst argumentu definīcijas” * “10. rindā trūkst argumentu definīcijas” |  |
| 4. | Tiek padota metode bez metodes galvas. | Kļūdas paziņojums   * “4. rindā trūkst metodes datu tipa un vārda” * “8. rindā trūkst metodes datu tipa un vārda” |  |
| 5. | Tiek padota metode, kurai nav anotāciju. | Kļūdas paziņojums   * “3. rindā trūkst metodes URL” |  |
| 6. | Tiek padota metode, kurai nav URL anotācijas. | Kļūdas paziņojums   * “3. rindā trūkst metodes URL” |  |
| 7. | Tiek padota pareizi definēta metode ar datu tipu *Void*. | Nav kļūdu |  |
| 8. | Tiek padota pareizi definēta metode, kuras datu tips ir bloka tips. | Kļūdas paziņojums   * “5. rindā tips *class* nav pareizs metodes datu tips” |  |
| 9. | Tiek padota pareizi definēta metode, kurai ir divas URL anotācijas. | Kļūdas paziņojums   * “8. rindā metodei jau ir definēts URL. Skatīt 3. rindu” |  |

#### 4.2.1.6 Metožu anotāciju testēšana

Šajā nodaļā koncentrējamies tikai uz anotāciju testēšanu, un norādām, kas anotācijas tiek/netiek padots. Visas pārējās metodes daļas ir pareizi padotas. Visos testpiemēros tiek padots URL tips (ja tips tiek padots), izņemot 10. testpiemērā, kur padots cits anotācijas tips. Kaut arī var tikt izdrukāta kļūda “Metodei trūkst URL”, šajā nodaļā tam nepievēršam uzmanību. Visi testpiemēri sakas ar 1. rindu.

|  |  |  |  |
| --- | --- | --- | --- |
| Nr. | Apraksts | Sagaidāmais rezultāts | Funkcija |
| 1. | Tiek padotas tukšas kvadrātiekavas. | Kļūdas paziņojums   * “3. rindā trūkst anotācijas satura” |  |
| 2. | Tiek padotas kvadrātiekavas ar anotācijas ķermeni. | Kļūdas paziņojums   * “3. rindā trūkst anotācijas tipa” |  |
| 3. | Tiek padotas kvadrātiekavas ar anotācijas tipu. | Kļūdas paziņojums   * “4. rindā trūkst anotācijas ķermeņa” |  |
| 4. | Tiek padotas kvadrātiekavas ar anotācijas tipu un tukšu ķermeni. | Kļūdas paziņojums   * “5. rindā trūkst anotācijas definīcijas” |  |
| 5. | Tiek padotas kvadrātiekavas ar anotācijas tipu un ķermeni, kurā ir tikai pēdiņas. | Kļūdas paziņojums   * “6. rindā trūkst anotācijas vērtības” |  |
| 6. | Tiek padotas kvadrātiekavas ar anotācijas tipu un ķermeni, kurā nav pēdiņu, bet ir vērtība. | Kļūdas paziņojums   * “6. rindā trūkst ‘ ” ‘ ” * “6. rindā trūkst ‘ ” ‘ ” |  |
| 7. | Tiek padotas URL anotācija, kuras vērtība nav pēdiņās un kurai nav padots protokols un lokācija. | Kļūdas paziņojums   * “6. rindā trūkst ‘ ” ‘ ” * “6. rindā trūkst URL protokola ” * “6. rindā trūkst URL lokācijas ” * “8. rindā trūkst ‘ ” ‘ ” |  |
| 8. | Tiek padotas URL anotācija, kuras vērtība ir pēdiņās un kurai nav padots protokols un lokācija. | Kļūdas paziņojums   * “6. rindā trūkst URL protokola ” * “7. rindā trūkst URL lokācijas ” |  |
| 9. | Tiek padotas URL anotācija, kuras vērtībā pirmais vai otrais atdalītājs nav kols. | Kļūdas paziņojums   * “7. rindā URL atribūti metodei ‘str’ nav doti” |  |
| 10. | Tiek padotas 9. testpiemērs, tikai anotācijas tips nav URL (šeit būs *path*). | Kļūdas paziņojums   * “3. rindā trūkst URL anotacijas” |  |

#### 4.2.1.7 Klases metožu testēšana

Šajā nodaļā koncentrējamies tikai uz metožu argumentu testēšanu, un norādām, kādi argumenti tiek padoti. Visas parejas metožu daļas ir pareizi padotas. Visi testpiemēri sakas ar 1. rindu.

|  |  |  |  |
| --- | --- | --- | --- |
| Nr. | Apraksts | Sagaidāmais rezultāts | Funkcija |
| 1. | Tiek padota metode ar vienu argumentu. | Nav kļūdu |  |
| 2. | Tiek padota metode ar diviem argumentiem, kas ir atdalīti ar komatu. | Nav kļūdu |  |
| 3. | Tiek padota metode ar vienu argumentu, kuram ir komats pirms un pēc argumenta definīcijas. | Kļūdas paziņojums   * “5. rindā trūkst argumenta” * “9. rindā trūkst argumenta” |  |
| 4. | Tiek padota metode, kurai iekavās ir tikai viens komats. | Kļūdas paziņojums   * “5. rindā trūkst argumenta” * “6. rindā trūkst argumenta” |  |
| 5. | Tiek padota metode, kurai iekavās ir divi komati. | Kļūdas paziņojums   * “5. rindā trūkst argumenta” * “6. rindā trūkst argumenta” * “7. rindā trūkst argumenta” |  |
| 6. | Tiek padota metode, kurai ir divi argumenti, kurus neatdala komats. | Kļūdas paziņojums   * “7. rindā trūkst komata” |  |
| 7. | Tiek padota metode ar vairākiem kļūdainiem argumentiem. Tie iekļauj:   1. Datu tips ir bloka tips 2. Datu tips ir *Void* 3. Datu tips ir aizsardzība 4. Trūkst datu tipa 5. Trūkst vārda | Kļūdas paziņojums   * “6. rindā *class* nav pareizs argumenta tips” * “9. rindā *public* nav pareizs argumenta tips” * “12. rindā *Void* nav pareizs argumenta tips” * “15. rindā trūkst argumenta vārda” * “17. rindā trūkst argumenta datu tipa” |  |
| 8. | Tiek padota metode ar trīs argumentiem, kuriem sakrīt vārds. | Kļūdas paziņojums   * “7. rindā arguments ar vārdu ‘test1’ jau eksistē. Skatīt 6. rindu” * “8. rindā arguments ar vārdu ‘test1’ jau eksistē. Skatīt 6. rindu” |  |

### 4.2.2 Kompilatora funkcionālā testēšana

Šajā testēšanas nodaļā koncentrējamies vairāk uz to, vai atkārtojas klases vārdi, asociāciju lomu vārdi, lauku vārdi, argumentu vārdi, vai atkārtojas lauki un asociāciju galapunkti virsklasēs un apaksklasēs, vai eksistē virsklases, vai rezervētie vārdi netiek izmantoti. Visos testpiemēros tiek padots sintaktiski pareizs kods.

#### 4.2.2.1 Klašu testēšana

Šajā nodaļā koncentrēsimies uz klases vārdu atbilstību. Visi testpiemēri sakas ar 1. rindu.

|  |  |  |  |
| --- | --- | --- | --- |
| Nr. | Apraksts | Sagaidāmais rezultāts | Funkcija |
| 1. | Tiek padota divas klases ar vienu un to pašu vārdu. | Kļūdas paziņojums   * “6. rindā klase ‘test1’ jau eksistē. Skatīt 2. rindu.” |  |
| 2. | Tiek padota divas klases ar dažādiem vārdiem un otrai klasei ir virsklase, kura neeksistē. | Kļūdas paziņojums   * “8. rindā nav klases ‘test3’, no kuras var mantot” |  |
| 3. | Tiek padota klase, kuras virsklase ir pati klase. | Kļūdas paziņojums   * “4. rindā nevar mantot no klases ‘test1’” |  |
| 4. | Tiek padota virsklase un apakšklase, kur virsklase ir definēta pēc apakšklases. | Kļūdas paziņojums   * “4. rindā nav klases ‘test2’, no kuras var mantot” |  |
| 5. | Tiek padota klase, kurā gan klases vārds, gan virsklases vards ir kāds no rezervētajiem vārdiem. | Kļūdas paziņojumi   * “2. rindā klasi nevar saukt par ‘URL’” * “4. rindā virsklasi nevar saukt par ‘BaseObject’” |  |

#### 4.2.2.2 Klases lauku testēšana

Šajā nodaļā koncentrēsimies uz lauku vārdu atbilstību un esamību virsklasēs. Visi testpiemēri sakas ar 1. rindu.

|  |  |  |  |
| --- | --- | --- | --- |
| Nr. | Apraksts | Sagaidāmais rezultāts | Funkcija |
| 1. | Tiek padota klase ar trīs atribūtiem un trīs metodēm, kur   1. Tiek izmantots klases vārds 2. Tiek izmantots rezervētais vārds 3. Vārds sākas ar “\_constructor\_” | Kļūdas paziņojumi   * “6. rindā metodes vārds nevar būt klases vārds.” * “12. rindā atribūta vārds nevar būt klases vārds.” * “18. rindā metodes vārds nevar būt ‘BaseObject’.” * “24. rindā atribūta vārds nevar būt ‘BaseObject’.” * “30. rindā metodes vārds nevar sākties ar ‘\_constructor’.” * “36. rindā atribūta vārds nevar sākties ar ‘\_constructor\_’.” |  |
| 2. | Tiek padota klase ar dažādiem laukiem, kuriem sakrīt vārds. Pirmais lauks ir atribūts. | Kļūdas paziņojumi   * “11. rindā klasē jau eksistē lauks ‘test’. Skatīt 5. rindu.” * “17. rindā klasē jau eksistē lauks ‘test’. Skatīt 5. rindu.” * “22. rindā klasē jau eksistē lauks ‘test’. Skatīt 5. rindu.” |  |
| 3. | Tiek padota klase ar dažādiem laukiem, kuriem sakrīt vārds. Pirmais lauks ir metode. | Kļūdas paziņojumi   * “12. rindā klasē jau eksistē lauks ‘test’. Skatīt 6. rindu.” * “18. rindā klasē jau eksistē lauks ‘test’. Skatīt 6. rindu.” * “25. rindā klasē jau eksistē lauks ‘test’. Skatīt 6. rindu.” |  |
| 4. | Tiek padotas trīs klases, kuras manto viena no otras pēc kārtas. Pirmajā klasē ir atribūts, bet pārējās abās ir atribūts un metode ar tādu pašu vārdu kā pirmās klases atribūts. Viens no atribūtiem ir identisks pirmās klases atribūtam. | Kļūdas paziņojumi   * “14. rindā lauks ‘test’ jau eksistē virsklasē ‘test1’. Skatīt 5. rindu.” * “20. rindā atribūtam ‘test’ nesakrīt datu tips ar lauku, kas atrodas virsklasē ‘test1’. Skatīt 5. rindu.” * “35. rindā lauks ‘test’ jau eksistē virsklasē ‘test1’. Skatīt 5. rindu” |  |
| 5. | Tiek padotas trīs klases, kuras manto viena no otras pēc kārtas. Pirmajā klasē ir metode, bet pārējās abās ir atribūts un metode ar tādu pašu vārdu kā pirmās klases atribūts. Viena no metodēm ir identisks pirmās klases atribūtam. Visām metodēm ir identiski argumenti. | Kļūdas paziņojumi   * “22. rindā lauks ‘test’ jau eksistē virsklasē ‘test1’. Skatīt 5. rindu.” * “31. rindā lauks ‘test’ jau eksistē virsklasē ‘test1’. Skatīt 5. rindu” * “37. rindā metodei ‘test’ nesakrīt datu tips ar lauku, kas atrodas virsklasē ‘test1’. Skatīt 5. rindu.” |  |
| 6. | Tiek padota virsklase un apakšklase. Virsklasē ir metode ar trīs argumentiem. Apakšklasē ir trīs metodes ar vienādu vārdu un atgriežamo tipu, kur   1. Argumentu skaits un datu tipi sakrīt 2. Argumentu skaits nesakrīt 3. Argumentu skaits sakrīt, bet datu tipi nesakrīt. Pirmais arguments sakrīt, otrais nē, bet trešajam nav datu tipa. | Kļūdas paziņojumi   * “24. rindā metodei ‘test’ nav vienāds argumentu skaits kā virsklases metodei ‘test’. Skatīt 5. rindu.” * “33. rindā argumentam trūkst datu tipa.” * “32. rindā argumentam nr. 2 nesakrīt datu tips ar virsklases metodi. Skatīt 5. rindu.” |  |
| 7. | Tiek padotas trīs klases, kuras manto viena no otras pēc kārtas. Pirmajā klasē ir privāts atribūts, bet pārējās abās ir publisks lauks ar tādu pašu vārdu | Kļūdas paziņojums   * “21. rindā lauks ‘test’ jau tiek izmantots virsklasē ‘test2’. Skatīt 13. rindu.” |  |
| 8. | Tiek padotas divas tukšas savā starpā neatkarīgas klases. Tad tiek padota asociācija starp šīm klasēm un visbeidzot trešā klase, kas manto no pirmās klases. Trešajā klasē ir atribūts un metode, kuras vārds sakrīt ar lomas vārdu tam asociācijas galapunktam, kurš tiek saglabāts pirmajā klasē. | Kļūdas paziņojumi   * “19. rindā lauks ‘test’ jau tiek izmantots virsklasē ‘test1’. Skatīt 13. rindu.” * “25. rindā lauks ‘test’ jau tiek izmantots virsklasē ‘test1’. Skatīt 13. rindu.” |  |
| 9. | Tiek padota virsklase un apakšklase. Virsklasē ir lauks un apakšklasē ir divi lauki identiski virsklases laukam. | Kļūdas paziņojums   * “15. rindā lauks ‘test’ jau tiek izmantots klasē ‘test2’. Skatīt 11. rindu” |  |

#### 4.2.2.3 Asociāciju testēšana

Šajā nodaļā koncentrēsimies uz asociācijas klašu vārdu un lomu vārdu atbilstību, kā arī lomu vārdu esamību virsklasēs un apakšklasēs. Visi testpiemēri sakas ar 1. rindu.

|  |  |  |  |
| --- | --- | --- | --- |
| Nr. | Apraksts | Sagaidāmais rezultāts | Funkcija |
| 1. | Tiek padota asociācija, kur kā klases vārdos tiek padots datu tips un bloka tips, šajā gadījumā, avotā ir bloka tips un mērķī ir klases tips. | Kļūdas paziņojumi   * “5. rindā avota klases vārds nevar būt ‘class’.” * “9. rindā mērķa klases vārds nevar būt ‘Integer’.” |  |
| 2. | Tiek padota asociācija, kur kā klases vārdos tiek padota aizsardzība un rezervētais vārds, šajā gadījumā, avotā ir aizsardzība un mērķī ir rezervētais vārds. | Kļūdas paziņojumi   * “5. rindā avota klases vārds nevar būt ‘public’.” * “9. rindā mērķa klases vārds nevar būt ‘URL’.” |  |
| 3. | Tiek padota pareizi definēta asociācija, bet bez definētām klasēm. | Kļūdas paziņojumi   * “5. rindā nav klases ‘sourceClass’, ko izmantot kā avota klasi.” * “9. rindā nav klases ‘targetClass’, ko izmantot kā mērķa klasi.” |  |
| 4. | Tiek padota avotklase un mērķklase, ka arī asociācija starp šīm klasēm. Asociācijas lomu vārdi ir bloka tips un datu tips, šajā gadījumā, avotā ir bloka tips un mērķī ir datu tips. | Kļūdas paziņojumi   * “6. rindā avota lomas vārds nevar būt ‘class’.” * “10. rindā mērķa lomas vārds nevar būt ‘Integer’.” |  |
| 5. | Tiek padota avotklase un mērķklase, ka arī asociācija starp šīm klasēm. Asociācijas lomu vārdi ir aizsardzība un rezervētais vārds, šajā gadījumā, avotā ir aizsardzība un mērķī ir rezervētais vārds. | Kļūdas paziņojumi   * “6. rindā avota lomas vārds nevar būt ‘public’.” * “10. rindā mērķa lomas vārds nevar būt ‘BaseObject’.” |  |
| 6. | Tiek padota avotklase un mērķklase, ka arī asociācija starp šīm klasēm. Avota lomas vārds ir mērķa klases vārds un mērķa lomas vārds ir avota klases vārds. | Kļūdas paziņojumi   * “6. rindā avota lomas vārds nevar būt ‘targetClass’.” * “10. rindā mērķa lomas vārds nevar būt ‘sourceClass’.” |  |
| 7. | Tiek padota avotklase un mērķklase, ka arī asociācija starp šīm klasēm. Lomu vārdi sākas ar “\_constructor\_”. | Kļūdas paziņojumi   * “6. rindā avota lomas vārds nevar sākties ar ‘\_constructor\_’.” * “10. rindā mērķa lomas vārds nevar sākties ar ‘\_constructor\_’.” |  |
| 8. | Tiek padota avotklase un mērķklase, ka arī asociācija starp šīm klasēm. Avotklasē ir atribūts un mērķklasē ir metode. Asociācijas avota lomas vārds sakrīt ar mērķklasē esošās metodes vārdu un mērķa lomas vārds sakrīt ar avotklasē esošā atribūta vārdu. | Kļūdas paziņojumi   * “18. rindā lomas vārds ‘test2’ jau tiek izmantots klasē ‘targetClass’. Skatīt 11. rindu.” * “22. rindā lomas vārds ‘test1’ jau tiek izmantots klasē ‘sourceClass’. Skatīt 4. rindu.” |  |
| 9. | Tiek padotas divas avotklases un mērķklase, ka arī asociācijas starp mērķklasi un abām avotklasēm. Abās asociācijās avota lomas vārds ir vienāds bet ir dažādas avotklases, savukārt mērķa definīcijas abās asociācijās ir vienādas. | Kļūdas paziņojums   * “24. rindā lomas vārds ‘test2’ jau tiek izmantots klasē ‘targetClass’. Skatīt 13. rindu.” |  |
| 10. | Tiek padotas četras klases, kur pirmās trīs klases secīgi manto viena no otras un ceturtā klase ir neatkarīga no pārējam trim klasēm. Vēl ir divas asociācijas, kurā pirmā asociācija ir starp pirmo un ceturto klasi un otra asociācija ir starp trešo un ceturto klasi. Ceturtā klase abās asociācijās ir mērķklase. Abu asociācijas mērķu definīcijas ir vienādas. Avotu lomu vārdi ir dažādi. | Kļūdas paziņojums   * “25. rindā lomas vārds ‘test4’ jau tiek izmantots virsklasē ‘test1’. Skatīt 14. rindu.” |  |
| 11. | Testpiemērs ir identisks 10. testpiemēram, tikai asociācijas ir definētas apgrieztā secībā. | Kļūdas paziņojums   * “25. rindā lomas vārds ‘test4’ jau tiek izmantots apakšklasē ‘test3’. Skatīt 14. rindu.” |  |
| 12. | Tiek padotas piecas klases, kurām ir šāda mantošana:   * 2. 3. un 4. klase manto no 1. klases * 5. klase manto no 4. klases   Vēl tiek padots sestā klase, kura ir neatkarīga no pārējām un četras asociācijas. Mērķa definīcijas visās asociācijās ir identiskas – mērķa klase ir sestā klase. Avotu lomu vārdi ir dažādi un klases ir šādā secībā: 2., 4., 5. un 1. klase. | Kļūdas paziņojumi   * “11. rindā lomas vārds ‘test6’ jau tiek izmantots virsklasē ‘test4’. Skatīt 10. rindu.” * “12. rindā lomas vārds ‘test6’ jau tiek izmantots apakšklasē ‘test2’. Skatīt 9. rindu.” |  |
| 13. | Tiek padotas četras klases, kur pirmās trīs klases secīgi manto viena no otras un ceturtā klase ir neatkarīga no pārējam trim klasēm. Pirmajā klasē ir atribūts. Ir arī definēta asociācija starp trešo klasi un ceturto klasi. Mērķa klase ir ceturtā klase un mērķa lomas vārds sakrīt ar pirmās klases atribūta vārdu. | Kļūdas paziņojums   * “17. rindā lomas vārds ‘targetName’ jau tiek izmantots virsklasē ‘test1’. Skatīt 5. rindu.” |  |
| 14. | Testpiemērs ir identisks 13. testpiemēram, tikai pirmajā klase ir metode nevis atribūts un tas privāts. | Nav kļūdu |  |
| 15. | Testpiemērs ir identisks 13. testpiemēram, tikai lauks ir definēts trešajā klasē, atribūts ir privāts un asociācija ir starp pirmo un ceturto klasi. | Kļūdas paziņojums   * “17. rindā lomas vārds ‘targetName’ jau tiek izmantots apakšklasē ‘test3’. Skatīt 7. rindu.” |  |
| 16. | Testpiemērs ir identisks 13. testpiemēram, tikai lauks ir definēts trešajā klasē, lauks ir metode nevis atribūts un asociācija ir starp pirmo un ceturto klasi. | Kļūdas paziņojums   * “20. rindā lomas vārds ‘targetName’ jau tiek izmantots apakšklasē ‘test3’. Skatīt 8. rindu.” |  |
| 17. | Tiek padota viena klase un asociācija, kur klases vārdi sakrīt, bet lomu vārdi nē. | Nav kļūdu |  |
| 18. | Tiek padota viena klase un asociācija, kur klases vārdi un lomu vārdi sakrīt. | Kļūdas paziņojums   * “9. rindā lomas vārds ‘role’ jau tiek izmantots klasē ‘test1’. Skatīt 5. rindu.” |  |
| 19. | Tiek padota virsklase un apakšklase un asociācija starp šīm klasēm, kur lomu vārdi nesakrīt. | Nav kļūdu |  |
| 20. | Tiek padota virsklase un apakšklase un asociācija starp šīm klasēm, kur lomu vārdi sakrīt. Avotā ir virsklases vārds. | Kļūdas paziņojums   * “10. rindā lomas vārds ‘role’ jau tiek izmantots virsklasē ‘test1’. Skatīt 6. rindu.” |  |
| 21. | Tiek padota virsklase un apakšklase un asociācija starp šīm klasēm, kur lomu vārdi sakrīt. Avotā ir apakšklases vārds. | Kļūdas paziņojums   * “10. rindā lomas vārds ‘role’ jau tiek izmantots apakšklasē ‘test1’. Skatīt 6. rindu.” |  |

#### 4.2.1.4 Vārdtelpu testēšana

Šajā nodaļā koncentrējamies uz vārdtelpu (namespace) formātu. Visos testpiemēros tiek padots tukšs jaunkoda fails.

|  |  |  |
| --- | --- | --- |
| N.P.K | Ievaddati | Sagaidāmais rezultāts |
| 1. | *\_* | Nav kļūdu |
| 2. | *n* | Nav kļūdu |
| 3. | *N* | Nav kļūdu |
| 4. | *1* | Ir kļūda. Nevar sākties ar ciparu |
| 5. | *\** | Ir kļūda. Nevar saturēt simbolus, kas nav apakšsvītra, burts vai cipars |
| 6. | *\_n* | Nav kļūdu |
| 7. | *nN* | Nav kļūdu |
| 8. | *N1* | Nav kļūdu |
| 9. | *\_\** | Ir kļūda. Nevar saturēt simbolus, kas nav apakšsvītra, burts vai cipars |

### 4.2.3 Ģeneratora testēšana

Šajā testēšanas nodaļā testēsim starpkoda ģeneratoru. Tā kā ģenerēšanas funkcija tiek izsaukta tikai tad, ja ir padots pareizs jaunkods, tad visi testpiemēros padotie jaunkodi ir pareizi. Visos testpiemēros ir jābūt uzģenerētam failam “BaseObject.cs”.

|  |  |  |  |
| --- | --- | --- | --- |
| Nr. | Apraksts | Sagaidāmais rezultāts | Funkcija |
| 1. | Tiek padots tukšs jaunkoda fails. | Netiek ģenereti citi faili. |  |
| 2. | Tiek padots jaunkods ar vienu klasi. | Tiek vēl uzģenerēts viens fails, kurš satur definēto klasi. |  |
| 3. | Tiek padots jaunkods ar trīs klasēm. | Tiek vēl uzģenerēts trīs fails, kur katrs fails satur definēto klasi. |  |
| 4. | Tiek padots jaunkods ar trīs klasēm, kur trešā klase manto no otrās klases. | Tiek vēl uzģenerēts trīs fails, kur katrs fails satur definēto klasi. Trešajai klasei virsklase ir otra klase un nevis BaseObject. |  |
| 5. | Tiek padots jaunkods ar vienu klasi, kurā ir pa vienam atribūtam no katra datu tipa. | Tiek vēl uzģenerēts viens fails, kurš satur definēto klasi ar visiem atribūtiem. |  |
| 6. | Tiek padots jaunkods ar vienu klasi, kurā ir pa vienai metodei no katra datu tipa. | Tiek vēl uzģenerēts viens fails, kurš satur definēto klasi ar visām metodēm. |  |
| 7. | Tiek padota virsklase un apakšklase. Virsklasē ir pāris atribūti ar dažādām aizsardzībām. Apakšklasē ir tie paši atribūti un vēl daži citi atribūti. | Tiek vēl uzģenerēti faili, kuri satur definētās klases ar visiem atribūtiem. |  |
| 8. | Tiek padota virsklase un apakšklase. Virsklasē ir pāris metodes ar dažādām aizsardzībām. Apakšklasē ir tās pašas metodes un vēl dažas citas metodes. | Tiek vēl uzģenerēti faili, kuri satur definētās klases ar visām metodēm. |  |
| 9. | Tiek padotas divas neatkarīgas tukšas klases un vairākas asociācijas starp tam. Asociācijām ir dažādas kompozīcijas patiesumvērtības. | Tiek vēl uzģenerēti faili, kuri satur definētās klases ar visām asociācijām. |  |
| 10. | Tiek padotas divas neatkarīgas klases, kur vienā no tam ir atribūti un metodes, bet otra ir tukša un mato no pirmās klases, un vairākas asociācijas starp tam. Asociācijām ir dažādas kompozīcijas patiesumvērtības. | Tiek vēl uzģenerēti faili, kuri satur definētās klases ar visiem atribūtiem, metodēm un asociācijām. |  |
| 11. | Tiek padota virsklase un apakšklase, kā arī divas asociācijas. Viena asociācija ir tikai starp virsklasi un otra ir starp virsklasi un apakšklasi. | Tiek vēl uzģenerēti faili, kuri satur definētās klases ar visām asociācijām. |  |

### 4.2.4 WebMemory testēšana

Šajā testēšanas nodaļā testēsim pašu starpkodu. Tiks testēts viss, izņemot metodes. Šajā testēšanā tiek izmantots starpkods, kurš ir uzģenerēts no šāda starpkoda.

class \_class1

{

private Integer \_int;

public String \_str;

}

class \_class2 : \_class1

{

public Integer \_int;

public Boolean \_bool;

}

class \_class3 : \_class1 {}

class \_class4 : \_class2

{

public Real \_real;

}

class \_class5 : \_class2

{

public Real \_real;

}

class \_class6 : \_class3

{

public Integer \_int;

}

association(source1:\_class2<->target1:\_class6)

class \_class7{}

class \_class8{}

class \_class9{}

association(source1:\_class7<->target1:\_class8)

association(source2:\_class8<->target2:\_class9)

class \_class10{}

class \_class11{}

class \_class12{}

association(source1:\_class10<->target1:\_class11)

association(source2:\_class11<->target2:\_class12)

association(source3:\_class12<->target3:\_class10)

|  |  |  |  |
| --- | --- | --- | --- |
| Nr. | Apraksts | Sagaidāmais rezultāts | Funkcija |
| 1. | Tiek pārbaudīta klašu iegūšana no WebMemory. |  |  |
| 2. | Atsevišķi tiek izveidota katras klases instance un tiek pārbaudīts, kādas klases ir WebMemory izveidotas un kuras nav. |  |  |
| 3. | Tiek pārbaudīts, vai visam klasēm, kad tās tiek izveidotas WebMemory, tiek izveidoti arī tās atribūti. |  |  |
| 4. | Tiek pārbaudīts, vai visam klasēm, kad tās tiek izveidotas WebMemory, tiek izveidoti arī tās asociācijas. |  |  |
| 5. | Tiek pārbaudīts, vai atribūtu vērtības tiek pareizi saglabātas un iegūtas. |  |  |
| 6. | Tiek pārbaudīts, vai asociāciju vērtības tiek pareizi saglabātas un iegūtas. |  |  |

## 4.3 Testēšanas protokols

### 4.3.1 Kompilatora sintaktiskā testēšana

#### 4.3.1.1 Bloku testēšana

|  |  |  |  |
| --- | --- | --- | --- |
| Nr. | Datums un rezultāts | | |
|  |  |  |  |
| 1. |  |  |  |
| 2. |  |  |  |
| 3. |  |  |  |
| 4. |  |  |  |
| 5. |  |  |  |
| 6. |  |  |  |
| 7. |  |  |  |
| 8. |  |  |  |
| 9. |  |  |  |
| 10. |  |  |  |
| 11. |  |  |  |
| 12. |  |  |  |

#### 4.3.1.2 Asociāciju testēšana

|  |  |  |  |
| --- | --- | --- | --- |
| Nr. | Datums un rezultāts | | |
|  |  |  |  |
| 1. |  |  |  |
| 2. |  |  |  |
| 3. |  |  |  |
| 4. |  |  |  |
| 5. |  |  |  |
| 6. |  |  |  |
| 7. |  |  |  |
| 8. |  |  |  |
| 9. |  |  |  |

#### 4.3.1.3 Klašu testēšana

|  |  |  |  |
| --- | --- | --- | --- |
| Nr. | Datums un rezultāts | | |
|  |  |  |  |
| 1. |  |  |  |
| 2. |  |  |  |
| 3. |  |  |  |
| 4. |  |  |  |

#### 4.3.1.4 Klases atribūtu testēšana

|  |  |  |  |
| --- | --- | --- | --- |
| Nr. | Datums un rezultāts | | |
|  |  |  |  |
| 1. |  |  |  |
| 2. |  |  |  |
| 3. |  |  |  |
| 4. |  |  |  |
| 5. |  |  |  |
| 6. |  |  |  |
| 7. |  |  |  |
| 8. |  |  |  |
| 9. |  |  |  |
| 10. |  |  |  |

#### 4.3.1.5 Klases metožu testēšana

|  |  |  |  |
| --- | --- | --- | --- |
| Nr. | Datums un rezultāts | | |
|  |  |  |  |
| 1. |  |  |  |
| 2. |  |  |  |
| 3. |  |  |  |
| 4. |  |  |  |
| 5. |  |  |  |
| 6. |  |  |  |
| 7. |  |  |  |
| 8. |  |  |  |
| 9. |  |  |  |

#### 4.3.1.6 Metožu anotāciju testēšana

|  |  |  |  |
| --- | --- | --- | --- |
| Nr. | Datums un rezultāts | | |
|  |  |  |  |
| 1. |  |  |  |
| 2. |  |  |  |
| 3. |  |  |  |
| 4. |  |  |  |
| 5. |  |  |  |
| 6. |  |  |  |
| 7. |  |  |  |
| 8. |  |  |  |
| 9. |  |  |  |
| 10. |  |  |  |

#### 4.3.1.7 Klases metožu testēšana

|  |  |  |  |
| --- | --- | --- | --- |
| Nr. | Datums un rezultāts | | |
|  |  |  |  |
| 1. |  |  |  |
| 2. |  |  |  |
| 3. |  |  |  |
| 4. |  |  |  |
| 5. |  |  |  |
| 6. |  |  |  |
| 7. |  |  |  |
| 8. |  |  |  |

### 4.3.2 Kompilatora funkcionālā testēšana

#### 4.3.2.1 Klašu testēšana

|  |  |  |  |
| --- | --- | --- | --- |
| Nr. | Datums un rezultāts | | |
|  |  |  |  |
| 1. |  |  |  |
| 2. |  |  |  |
| 3. |  |  |  |
| 4. |  |  |  |
| 5. |  |  |  |

#### 4.3.2.2 Klases lauku testēšana

|  |  |  |  |
| --- | --- | --- | --- |
| Nr. | Datums un rezultāts | | |
|  |  |  |  |
| 1. |  |  |  |
| 2. |  |  |  |
| 3. |  |  |  |
| 4. |  |  |  |
| 5. |  |  |  |
| 6. |  |  |  |
| 7. |  |  |  |
| 8. |  |  |  |
| 9. |  |  |  |

#### 4.3.2.3 Asociāciju testēšana

|  |  |  |  |
| --- | --- | --- | --- |
| Nr. | Datums un rezultāts | | |
|  |  |  |  |
| 1. |  |  |  |
| 2. |  |  |  |
| 3. |  |  |  |
| 4. |  |  |  |
| 5. |  |  |  |
| 6. |  |  |  |
| 7. |  |  |  |
| 8. |  |  |  |
| 9. |  |  |  |
| 10. |  |  |  |
| 11. |  |  |  |
| 12. |  |  |  |
| 13. |  |  |  |
| 14. |  |  |  |
| 15. |  |  |  |
| 16. |  |  |  |
| 17. |  |  |  |
| 18. |  |  |  |
| 19. |  |  |  |
| 20. |  |  |  |
| 21. |  |  |  |

#### 4.3.1.4 Vārdtelpu testēšana

|  |  |  |  |
| --- | --- | --- | --- |
| Nr. | Datums un rezultāts | | |
|  |  |  |  |
| 1. |  |  |  |
| 2. |  |  |  |
| 3. |  |  |  |
| 4. |  |  |  |
| 5. |  |  |  |
| 6. |  |  |  |
| 7. |  |  |  |
| 8. |  |  |  |
| 9. |  |  |  |

### 4.3.3 Ģeneratora testēšana

|  |  |  |  |
| --- | --- | --- | --- |
| Nr. | Datums un rezultāts | | |
|  |  |  |  |
| 1. |  |  |  |
| 2. |  |  |  |
| 3. |  |  |  |
| 4. |  |  |  |
| 5. |  |  |  |
| 6. |  |  |  |
| 7. |  |  |  |
| 8. |  |  |  |
| 9. |  |  |  |
| 10. |  |  |  |
| 11. |  |  |  |

### 4.3.4 WebMemory testēšana

|  |  |  |  |
| --- | --- | --- | --- |
| Nr. | Datums un rezultāts | | |
|  |  |  |  |
| 1. |  |  |  |
| 2. |  |  |  |
| 3. |  |  |  |
| 4. |  |  |  |
| 5. |  |  |  |
| 6. |  |  |  |

# 5. PROJEKTA DARBA ORGANIZĀCIJA

Darbs tika izstrādāts patstāvīgi. Tas iekļauj kodēšanu, dokumentācijas veidošanu un testēšanu.

Tika izmantota spirālveida paradigma. Vispirms tika izrunātas visas programmatūras pamatprasības un tad notika kodēšana. Kad tika uzkodēts veiksmīgs programmkods, tad tas tika projektēts. Gadījumā, ja notiek izmaiņas prasībās, vai tiek pieliktas jaunas prasības, tika veiktas izmaiņas dokumentācijā un kodēšanā uzreiz, kā izmaiņas parādījās.

# 6. KVALITĀTES NODROSINĀŠANA

* Funkcionalitāte
  + Precizitāte - nodrošinājām un testējām to, lai programma strādās pareizi.
  + Sadarbspēja – tiek izmantoti DLL faili, kas izmanto Java virtualo mašīnu.
  + Piemērotība – programma darbojas tam paredzētajās OS un atbilst WebAppOS API
  + Funkcionalitātes atbilstība - tika rūpēts par to, ka ir izveidotas visas funkcijas, kuras prasībās tika pieminētas
* Lietojamība
  + Saprotamība un apgūstamība - WAOS C# sintakse ir līdzīga C# sintaksei, lai varētu vieglāk to apgūt.
  + Darbināmība un lietojamības atbilstība - koda ģeneratoram ir jābūt ērti izsaucamam no komandrindas gan no grafiska OS, gan no konsoles serveros.
  + Pievilcība - Kompilators, noķerot kļūdas, parāda, kurās rindās tās ir sastopamas un kurās rindās tiek izmantoti jau esoši identifikatori. WebMemory tiek veidots kā augsta līmeņa RAAPI, lai tas būtu ērti lietojams izstrādātājiem.
* Uzturamība - Programmkodi tika veidoti dažādos failos, kur katrā failā tiek norādīts, kādas funkcijas tie satur. Failu grupas tika apvienotas direktīvas. Metožu, tās argumentu un atribūtu nosaukumi tiek definēti atbilstoši tās funkcionalitātei.
* Pārnesamība
  + Koeksistence – WebAppOS tika izstrādāta Java, bet šī programmatūra – .NET 5 un tie sadarbojas caur DLL failiem.
  + Pārnesamības atbilstība - Programma tika izstrādāta ar .NET 5 un dotais RAAPI ir izstrādāts Java, kas ir pārnēsājama dažādās OS.

# 7. KONFIGURĀCIJU PĀRVALDĪBA

Konfigurācijā ietilpst ANTLR, kas tika izmantots kompilatorā, pats kompilators un ģenerators, kā arī WebMemory un programmatūras dokumentācija.

Tika izmantots GitHub, lai glabātu programmkodu, dokumentāciju un testpiemērus.

Prims pirmās versijas izveidošanas, katrā atjaunojumā tika glabāts īss apraksts par to, kas tika mainīts/noņemts/pievienots, norādot, vai tas bija kods vai dokumentācija, kāds modulis tika skatīts un pati darbība.

Konfigurācijas atjaunošana notiek katru reizi, kad kādā modulī tiek veiktas izmaiņas, kā arī katras darba dienas beigās, lai cik maz izmaiņu tika veiktas.

# 8. DARBIETILPĪBAS NOVĒRTĒJUMS

# 9. PROGRAMMKODU PIELIKUMS

# 10. IZMANTOTĀ LITERATŪRA