# 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 – vai ir izlaistas koda daļas, pareizi atslēgas vārdi.
* Funkcionalitāte – vai atkārtojas klases vārdi, asociāciju lomu vārdi, lauku vārdi, argumentu vārdi, vai atkārtojas lauki un asociāciju galapunkti virsklasēs un apaksklasēs, vai eksistē virsklases.

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. Netiek skatīts tas, vai atkārtojas/eksistē klases, atribūti, elementi, asociāciju galapunkti, argumenti, vai tiek izmantoti rezervētie vārdi tur, kur nevajag, vai ir pareizs protokols un lokācija.

#### 4.2.1.1 Apzīmējumu izklāsts

Šeit ir visu vērtību, kuras tiks izmantotas testēšanā, izklāsts.

|  |  |
| --- | --- |
| Vērtība | Paskaidrojums |
| class | Klases tips |
| association | Asociācijas tips |
| baseClass, superClass, sourceClass, targetClass | Klases vārdi. sourceClass un targetClass tiks izmantotas asociācijās kā avotklase un mērķa klase respektīvi. superClass tiks izmantota kā virsklase. |
| sourceName, targetname | Asociācijas avota un lomas vārds respektīvi. |
|  |  |

#### 4.2.1.2 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.

|  |  |  |
| --- | --- | --- |
| N.P.K | Ievaddati | Sagaidāmais rezultāts |
| 1. | Tukšs fails | Nav kļūdu |
| 2. | *(sourceName:sourceClass<->targetName:targetClass)* | Kļūdas paziņojums “Trūkst atslēgas vārda ‘association’” |
| 3. | *className : superClass {}* | Kļūdas paziņojums “Trūkst atslēgas vārda ‘class’” |
| 4. | *class* | Kļūdas paziņojums “Trūkst klases ķermeņa” |
| 5. | *class (sourceName:sourceClass<->targetName:targetClass)* | Kļūdas paziņojums “Klasei ir dota asociācijas definīcija” |
| 6. | *class className : superClass {}* | Nav kļūdu |
| 7. | *association* | Kļūdas paziņojums “Trūkst asociācijas ķermeņa” |
| 8. | *association (sourceName:sourceClass<->targetName:targetClass)* | Nav kļūdu |
| 9. | *association className : superClass {}* | Kļūdas paziņojums “Asociācijai ir dota klases definīcija” |
| 10. | *Integer* | Kļūdas paziņojums “’Integer’ nav bloka tips. Izmanrtojiet ‘*class*’ vai ‘*association*’!” |
| 11. | *Integer (sourceName:sourceClass<->targetName:targetClass)* | Kļūdas paziņojums “’Integer’ nav bloka tips. Izmanrtojiet ‘*association*’!” |
| 12. | *Integer className : superClass {}* | Kļūdas paziņojums “’Integer’ nav bloka tips. Izmanrtojiet ‘*class*’!” |

#### 4.2.1.3 Asociāciju testēšana

Nodaļā 2.1.1.1 tika aprakstīta asociācijas sintakse.