# 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 – 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. 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.

|  |  |  |
| --- | --- | --- |
| Testpiemēra Nr. | Apraksts | Funkcija |
| 1. | Tiek padots tukšs jaunkoda fails.  Tukša faila gadījumā kļūdu nav. |  |
| 2. | Tiek padota asociācijas definīcija bez atslēgvārda.  Šādā gadījumā ir jābūt kļūdai, ka nav atslēgvārda *association*. |  |
| 3. | Tiek padota klases definīcija bez atslēgvārda.  Šādā gadījumā ir jābūt kļūdai, ka nav atslēgvārda *class*. |  |
| 4. | Tiek padots atslēgvārds *class* bez bloka ķermeņa.  Šādā gadījumā ir jābūt kļūdai, ka nav klases definīcijas. |  |
| 5. | Tiek padota asociācijas definīcija ar atslēgvārdu *class*.  Šādā gadījumā ir jābūt kļūdai, ka klasei ir dota asociācijas definīcija. |  |
| 6. | Tiek padota klases definīcija ar atslēgvārdu *class*.  Šādā gadījumā kļūdu nav. |  |
| 7. | Tiek padots atslēgvārds *association* bez bloka definīcijas.  Šādā gadījumā ir jābūt kļūdai, ka nav asociācijas definīcijas. |  |
| 8. | Tiek padota asociācijas definīcija ar atslēgvārdu *association*.  Šādā gadījumā kļūdu nav. |  |
| 9. | Tiek padota klases definīcija ar atslēgvārdu *association*.  Šādā gadījumā ir jābūt kļūdai, ka asociācijai ir dota klases definīcija. |  |
| 10. | Tiek padots neatbalstīts bloka tips bez bloka ķermeņā.  Šadā gadījuma ir kļūda, ka padotais tips ir nepareizs un ka ir jāizmanto *class* vai *association*. |  |
| 11. | Tiek padots neatbalstīts bloka tips ar asociācijas definīciju.  Šadā gadījuma ir kļūda, ka padotais tips ir nepareizs un ka ir jāizmanto *association*. |  |
| 12. | Tiek padots neatbalstīts bloka tips ar klases definīciju.  Šadā gadījuma ir kļūda, ka padotais tips ir nepareizs un ka ir jāizmanto *class*. |  |

|  |  |  |
| --- | --- | --- |
| 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*’!” |

#### 4.2.1.2 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.

|  |  |  |
| --- | --- | --- |
| Testpiemēra Nr. | Apraksts | Funkcija |
| 1. | Tiek padota asociācija bez definīcijas iekavās.  Šādā gadījumā ir jābūt kļūdai, ka asociācijai trūkst definīcijas.  Rinda sakrīt ar atverošās iekavas rindu. |  |
| 2. | Tiek padota asociācija, kurai nav definēts avots.  Šādā gadījumā ir jābūt kļūdai, ka asociācijai trūkst avota.  Rinda sakrīt ar atverošās iekavas rindu. |  |
| 3. | Tiek padota asociācija, kurai nav bultu.  Šādā gadījumā ir jābūt kļūdai, ka asociācijai trūkst bultu.  Rinda sakrīt ar atverošās asociācijas avota beigu rindu, jo nevar būt tāds gadījums, ka ir uzrakstīts tikai asociācijas mērķis. |  |
| 4. | Tiek padota asociācija, kurai nav definēts mērķis.  Šādā gadījumā ir jābūt kļūdai, ka asociācijai trūkst mērķa.  Rinda sakrīt ar bultu rindu. |  |
| 5. | Tiek padota asociācija, kurai nav bultu un nav definēts mērķis.  Šādā gadījumā ir jābūt kļūdai, ka asociācijai trūkst bultu un ka trūkst mērķa.  Abām kļūdām rinda sakrīt ar asociācijas avota beigu rindu. |  |
| 6. | Tiek padota asociācija, kurā avotam/mērķim nav doti koli.  Šādā gadījumā ir jābūt kļūdai, ka asociācijas avotā/mērķī nav kola.  Abām kļūdām rinda sakrīt ar lomu vārdu rindām, jo nevar būt tāds gadījums, kad ir definēta tikai avota/mērķa klase. |  |
| 7. | Tiek padota asociācija, kurā avotam/mērķim nav dotas klases.  Šādā gadījumā ir jābūt kļūdai, ka asociācijas avotā/mērķī nav klašu.  Abām kļūdām rinda sakrīt ar kola rindām. Ja nav kola, tad rinda sakrīt ar lomas vārda rindu. Šajā testpiemērā avotā kolu liekam, bet mērķī – nē. Pretēja secībā funkcija darbosies vienādi. |  |
| 8. | Tiek padota asociācija, kurā avotam/mērķim nav doti lomu vārdi.  Šādā gadījumā ir jābūt kļūdai, ka asociācijas avotā/mērķī nav lomu vārdu.  Avota lomas vārda kļūdā rinda sakrīt ar atverošo iekavu rindu, bet mērķa lomas vārda gadījumā ar bultu rindu. Ja bultu nav, tad ar avota beigām (skat. 9. testpiemēru). Nevar būt gadījums, kad ir definēts tikai mērķis. |  |
| 9. | Tiek padota asociācija, kurā avotam/mērķim nav doti lomu vārdi un nav bultu.  Šādā gadījumā uzvedība ir identiska 8. testpiemēram, tikai mērķa lomas vārda kļūdas rinda sakrīt ar avota beigu rindu. |  |

|  |  |  |
| --- | --- | --- |
| N.P.K | Ievaddati | Sagaidāmais rezultāts |
| 1. | *association*  *(*  *)* | Kļūdas paziņojums   * “5. rindā trūkst asociācijas definīcijas” |
| 2. | *association*  *(*  *<->*  *targetname*  *:*  *targetClass*  *)* | Kļūdas paziņojums   * “5. rindā trūkst asociācijas avota definīcijas” |
| 3. | *association*  *(*  *sourceName*  *:*  *sourceClass*  *targetname*  *:*  *targetClass*  *)* | Kļūdas paziņojums   * “8. rindā trūkst bultu asociācijas definīcijā” |
| 4. | *association*  *(*  *sourceName*  *:*  *sourceClass*  *<->*  *)* | Kļūdas paziņojums   * “9. rindā trūkst asociācijas mērķa definīcijas” |
| 5. | *association*  *(*  *sourceName*  *:*  *sourceClass*  *)* | Kļūdas paziņojumi   * “8. rindā trūkst bultas asociācijas definīcijā” * ”8. rindā trūkst asociācijas mērķa definīcijas” |
| 6. | *association*  *(*  *sourceName*  *sourceClass*  *<->*  *targetName*  *targetClass*  *)* | Kļūdas paziņojumi   * “6. rindā trūkst ‘:’ avota definīcijā” * “9. rindā trūkst ‘:’ mērķa definīcijā” |
| 7. | *association*  *(*  *sourceName*  *:*  *<->*  *targetName*  *)* | Kļūdas paziņojumi   * “7. rindā trūkst avota klases” * “9. rindā trūkst ‘:’ mērķa definīcijā” * “9. rindā trūkst mērķa klases” |
| 8. | *association*  *(*  *:*  *sourceClass*  *<->*  *:*  *targetClass*  *)* | Kļūdas paziņojumi   * “5. rindā trūkst avota lomas vārda” * “8. rindā trūkst mērķa lomas vārda” |
| 9. | *association*  *(*  *:*  *sourceClass*  *:*  *targetClass*  *)* | 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.

|  |  |  |
| --- | --- | --- |
| Testpiemēra Nr. | Apraksts | Funkcija |
| 1. | Tiek padota klase bez ķermeņa.  Šādā gadījumā ir jābūt kļūdai, ka klasei trūkst ķermeņa.  Rinda sakrīt ar klases galvas beigu rindu. |  |
| 2. | Tiek padota klase bez galvas.  Šādā gadījumā ir jābūt kļūdai, ka klasei trūkst galvas.  Rinda sakrīt ar atslēgvārda *class* rindu. Ja atslēgvārda *class* nav, tad rinda sakrīt ar ķermeņa rindu. |  |
| 3. | Tiek padota klases bez klases vārda un ar kolu bez virsklases varda. Virsklase jau pirms tam ir definēta.  Šādā gadījumā ir jābūt kļūdai, ka klasei trūkst klases vārda un virsklases vārda.  Virsklases vārda kļūdas rinda sakrīt ar kola rindu.  Klases vārda kļūdas rinda sakrīt ar atslēgvārda *class* rindu. Ja atslēgvārda *class* nav, tad rinda sakrīt ar kola rindu. Ja nav kola, tad rinda sakrīt ar klases ķermeņa rindu |  |
| 4. | Tiek padota pareizi definēta klase ar virsklasi. Pareizi defineta klase bez virsklases jau tika apskatīta bloku testēsanā. Virsklase jau pirms tam ir definēta.  Šādā gadījumā kļūdu nav. |  |

|  |  |  |
| --- | --- | --- |
| N.P.K | Ievaddati | Sagaidāmais rezultāts |
| 1. | *class*  *className* | Kļūdas paziņojums   * “2. rindā trūkst klases ķermeņa” |
| 2. | *class*  *{}*  *{}* | Kļūdas paziņojumi   * “1. rindā trūkst klases galvas” * “4. rindā trūkst atslēgvārda *class*” * “4. rindā trūkst klases galvas” |
| 3. | *class superClass{}*  *class*  *:*  *{}*  *:*  *superClass*  *{}*  *{}* | 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. | *class superClass{}*  *class*  *className*  *:*  *superClass*  *{}* | 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.

|  |  |  |
| --- | --- | --- |
| Testpiemēra Nr. | Apraksts | Funkcija |
| 1. | Tiek padots tukšs lauks jeb tikai semikols.  Šādā gadījumā kļūdu nav. |  |
| 2. | Tiek padots atribūts, kuram ir tikai vārds.  Šādā gadījumā ir jābūt kļūdai, ka trūkst datu tipa.  Rinda sakrīt ar lauka vārda rindu |  |
| 3. | Tiek padots atribūts, kuram ir tikai datu tips.  Šādā gadījumā ir jābūt kļūdai, ka trūkst vārda.  Rinda sakrīt ar datu tipa rindu |  |
| 4. | Tiek padots atribūts, kuram ir datu tips un vārds.  Šādā gadījumā kļūdu nav. |  |
| 5. | Tiek padots atribūts, kuram ir tikai aizsardzība.  Šādā gadījumā ir jābūt kļūdai, ka trūkst datu tipa un vārda.  Rinda abām kļūdām sakrīt ar aizsardzības rindu. |  |
| 6. | Tiek padots atribūts, kuram ir aizsardzība un vārds.  Šādā gadījumā ir jābūt kļūdai, ka trūkst datu tipa.  Rinda sakrīt ar aizsardzības rindu. |  |
| 7. | Tiek padots atribūts, kuram ir aizsardzība, datu tips un vārds.  Šādā gadījumā kļūdu nav. |  |
| 8. | Tiek padots pareizi definēts atribūts, kuram datu tips ir bloka tips.  Šādā gadījumā ir jābūt kļūdai, ka bloka tips nav pareizs atribūta datu tips.  Rinda sakrīt ar padota bloka tipa rindu. |  |
| 9. | Tiek padots pareizi definēts atribūts, kuram datu tips ir *Void*.  Šādā gadījumā ir jābūt kļūdai, ka *Void* nav pareizs atribūta datu tips.  Rinda sakrīt ar vārda *Void* rindu. |  |
| 10. | Tiek padoti vairāki lauki bez semikoliem.  Šādā gadījuma ir kļūdām, ka trūkst kolu.  Rindas sakrīt ar lauku beigu rindām. |  |

|  |  |  |
| --- | --- | --- |
| N.P.K | Ievaddati | Sagaidāmais rezultāts |
| 1. | *class className*  *{*  *;*  *}* | Nav kļūdu |
| 2. | *class className*  *{*  *\_int*  *;*  *}* | Kļūdas paziņojums   * “3. rinda trūkst atribūta datu tipa” |
| 3. | *class className*  *{*  *Integer*  *;*  *}* | Kļūdas paziņojums   * “3. rindā trūkst atribūta vārda” |
| 4. | *class className*  *{*  *Integer*  *\_int*  *;*  *}* | Nav kļūdu |
| 5. | *class className*  *{*  *public*  *;*  *}* | Kļūdas paziņojums   * “3. rindā trūkst atribūta datu tipa un vārda” |
| 6. | *class className*  *{*  *public*  *\_int*  *;*  *}* | Kļūdas paziņojums   * “3. rindā trūkst atribūta datu tipa” |
| 7. | *class className*  *{*  *public*  *Integer*  *\_int*  *;*  *}* | Nav kļūdu |
| 8. | *class className*  *{*  *public*  *class*  *\_Class*  *;*  *}* | Kļūdas paziņojums   * “4. rindā tips *class* nav pareizs atribūta datu tips” |
| 9. | *class className*  *{*  *public*  *Void*  *\_void*  *;*  *}* | Kļūdas paziņojums   * “4. rindā tips *Void* nav pareizs atribūta datu tips” |
| 10. | *class className*  *{*  *[URL("dotnet:local:*  *namespaceName.*  *className#methodName")]*  *Integer*  *test1*  *()*    *Integer*  *integer1*    *Integer*  *integer2*    *[URL("dotnet:local:*  *namespaceName.*  *className#methodName")]*  *Integer*  *test2*  *()*    *[URL("dotnet:local:*  *namespaceName.*  *className#methodName")]*  *Integer*  *test3*  *()*  *}* | 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.

|  |  |  |
| --- | --- | --- |
| Testpiemēra Nr. | Apraksts | Funkcija |
| 1. | Tiek padota metode ar URL un ne URL anotāciju, datu tipu, vārdu un iekavām.  Šādā gadījumā kļūdu nav. |  |
| 2. | Tiek padota metode ar URL anotāciju, aizsardzību datu tipu, vārdu un iekavām.  Šādā gadījumā kļūdu nav. |  |
| 3. | Tiek padota metode bez iekavām.  Šādā gadījumā ir jābūt kļūdai, ka trūkst iekavu.  Rinda sakrīt ar metodes galvas beigu rindu. Ja tā nav dota, tad rinda sakrīt ar pēdējās anotācijas beigu rindu. |  |
| 4. | Tiek padota metode bez metodes galvas.  Šajā gadījuma ir kļūda, ka trūkst metodes datu tipa un vārda.  Rinda sakrīt ar pēdējās anotācijas beigu rindu. Ja anotācijas nav dotas, tad sakrīt ar iekavu sākumrindu. |  |
| 5. | Tiek padota metode, kurai nav anotāciju.  Šādā gadījumā ir kļūda, ka trūkst URL anotācijas.  Rinda sakrīt ar metodes sākumrindu. |  |
| 6. | Tiek padota metode, kurai nav URL anotācijas.  Šādā gadījuma ir jābūt kļūdai, ka trūkst URL anotācijas.  Rinda sakrīt ar metodes sākumrindu. |  |
| 7. | Tiek padota pareizi definēta metode ar datu tipu *Void*.  Šādā gadījuma kļūdu nav. |  |
| 8. | Tiek padota pareizi definēta metode, kuras datu tips ir bloka tips.  Šādā gadījuma ir jābūt kļūdai, ka padotais bloka tips nav pareizs metodes datu tips. |  |
| 9. | Tiek padota pareizi definēta metode, kurai ir divas URL anotācijas.  Šādā gadījuma ir jābūt kļūdai, ka URL anotācija jau ir definēta.  Rinda sakrīt ar liekās URL anotācijas tipa rindu (tur, kur anotācijā ir rakstīts *URL*). Vēl jānorada rinda, kurā jau ir URL anotācija. |  |

|  |  |  |
| --- | --- | --- |
| N.P.K | Ievaddati | Sagaidāmais rezultāts |
| 1. | *class className*  *{*  *[URL("dotnet:local:*  *namespaceName*  *.className#methodName")]*  *[path("C:/test")]*  *Integer*  *\_int*  *()*  *;*  *}* | Nav kļūdu |
| 2. | *class className*  *{*  *[URL("dotnet:local:*  *namespaceName.*  *className#methodName")]*  *public*  *Integer*  *\_int*  *()*  *;*  *}* | Nav kļūdu |
| 3. | *class className*  *{*  *[path("C:/test")]*  *[URL("dotnet:local:*  *namespaceName.*  *className#methodName")]*  *Integer*  *\_int*  *;*    *[path("C:/test")]*  *[URL("dotnet:local:*  *namespaceName.*  *className#methodName")]*  *;*  *}* | Kļūdas paziņojums   * “6. rindā trūkst argumentu definīcijas” * “10. rindā trūkst argumentu definīcijas” |
| 4. | *class className*  *{*  *[URL("dotnet:local:*  *namespaceName.*  *className#methodName")]*  *[path("C:/test")]*  *()*  *;*    *()*  *;*  *}* | Kļūdas paziņojums   * “4. rindā trūkst metodes datu tipa un vārda” * “8. rindā trūkst metodes datu tipa un vārda” |
| 5. | *class className*  *{*  *Integer*  *\_int*  *()*  *;*  *}* | Kļūdas paziņojums   * “3. rindā trūkst metodes URL” |
| 6. | *class className*  *{*  *[path("C:/test")]*  *Integer*  *\_int*  *()*  *;*  *}* | Kļūdas paziņojums   * “3. rindā trūkst metodes URL” |
| 7. | *class className*  *{*  *[path("C:/test")]*  *[URL("dotnet:local:*  *namespaceName.*  *className#methodName")]*  *Void*  *vi*  *()*  *;*  *}* | Nav kļūdu |
| 8. | *class className*  *{*  *[URL("dotnet:local:*  *namespaceName.*  *className#methodName")]*  *[path("C:/test")]*  *class*  *cl*  *()*  *;*  *}* | Kļūdas paziņojums   * “5. rindā tips *class* nav pareizs metodes datu tips” |
| 9. | *class className*  *{*  *[*  *URL*  *("dotnet:local:*  *namespaceName.*  *className#methodName1")*  *]*  *[*  *URL*  *("dotnet:local:*  *namespaceName.*  *className#methodName2")*  *]*  *Integer*  *\_int*  *()*  *;*  *}* | 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.

|  |  |  |
| --- | --- | --- |
| Testpiemēra Nr. | Apraksts | Funkcija |
| 1. | Tiek padotas tukšas kvadrātiekavas.  Šādā gadījumā ir jābūt kļūdai, ka trūkst anotācijas satura.  Rinda sakrīt ar atverošās iekavas rindu. |  |
| 2. | Tiek padotas kvadrātiekavas ar anotācijas ķermeni.  Šādā gadījumā ir jābūt kļūdai, ka trūkst anotācijas tipa.  Rinda sakrīt ar atverošās iekavas rindu. |  |
| 3. | Tiek padotas kvadrātiekavas ar anotācijas tipu.  Šādā gadījumā ir jābūt kļūdai, ka trūkst anotācijas ķermeņa.  Rinda sakrīt ar anotācijas tipa rindu. |  |
| 4. | Tiek padotas kvadrātiekavas ar anotācijas tipu un tukšu ķermeni.  Šādā gadījumā ir jābūt kļūdai, ka trūkst anotācijas definīcijas.  Rinda sakrīt ar apaļo atverošo iekavu rindu. |  |
| 5. | Tiek padotas kvadrātiekavas ar anotācijas tipu un ķermeni, kurā ir tikai pēdiņas.  Šādā gadījumā ir jābūt kļūdai, ka trūkst anotācijas vērtības.  Rinda sakrīt ar atverošo pēdiņu rindu. |  |
| 6. | Tiek padotas kvadrātiekavas ar anotācijas tipu un ķermeni, kurā nav pēdiņu, bet ir vērtība.  Šādā gadījumā ir jābūt kļūdai, ka trūkst pēdiņu.  Atverošajām pēdiņām rinda sakrīt ar anotācijas vērtības sākumrindu, bet aizverošajām pēdiņām – ar beigu rindu. |  |
| 7. | Tiek padotas URL anotācija, kuras vērtība nav pēdiņās un kurai nav padots protokols un lokācija.  Šādā gadījumā ir jābūt kļūdai, ka trūkst pēdiņu un ka trūkst protokola un lokācijas.  Rinda protokola kļūdai ir pirms pirmā kola, bet lokācijai – pēc pirmā kola. |  |
| 8. | Tiek padotas URL anotācija, kuras vērtība ir pēdiņās un kurai nav padots protokols un lokācija.  Šādā gadījumā ir jābūt kļūdai, ka trūkst protokola un lokācijas.  Rinda protokola kļūdai ir pēc atverošajām pediņām, bet lokācijai – pēc pirmā kola. |  |
| 9. | Tiek padotas URL anotācija, kuras vērtībā pirmais vai otrais atdalītājs nav kols.  Šādā gadījumā ir jābūt kļūdai, ka trūkst URL atribūtu.  Rinda sakrīt ar anotācijas vērtības sākumrindu. |  |
| 10. | Tiek padotas 9. testpiemērs, tikai anotācijas tips nav URL (šeit būs *path*).  Šādā gadījumā ir tikai kļūda ka trūkst URL anotacijas. Citadi kļūdu nav. |  |

|  |  |  |
| --- | --- | --- |
| N.P.K | Ievaddati | Sagaidāmais rezultāts |
| 1. | *class Class1*  *{*  *[*  *]*  *String str*  *()*  *;*  *}* | Kļūdas paziņojums   * “3. rindā trūkst anotācijas satura” |
| 2. | *class Class1*  *{*  *[*  *("dotnet:local:*  *namespaceName.*  *className#methodName")*  *]*  *String str*  *()*  *;*  *}* | Kļūdas paziņojums   * “3. rindā trūkst anotācijas tipa” |
| 3. | *class Class1*  *{*  *[*  *URL*  *]*  *String str*  *()*  *;*  *}* | Kļūdas paziņojums   * “4. rindā trūkst anotācijas ķermeņa” |
| 4. | *class Class1*  *{*  *[*  *URL*  *(*  *)*  *]*  *String str*  *()*  *;*  *}* | Kļūdas paziņojums   * “5. rindā trūkst anotācijas definīcijas” |
| 5. | *class Class1*  *{*  *[*  *URL*  *(*  *"*  *"*  *)*  *]*  *String str*  *()*  *;*  *}* | Kļūdas paziņojums   * “6. rindā trūkst anotācijas vērtības” |
| 6. | *class Class1*  *{*  *[*  *URL*  *(*  *dotnet:local:*  *namespaceName.*  *className#methodName*  *)*  *]*  *String str*  *()*  *;*  *}* | Kļūdas paziņojums   * “6. rindā trūkst ‘ ” ‘ ” * “6. rindā trūkst ‘ ” ‘ ” |
| 7. | *class Class1*  *{*  *[*  *URL*  *(*  *:*  *:*  *namespaceName.*  *className#methodName*  *)*  *]*  *String str*  *()*  *;*  *}* | Kļūdas paziņojums   * “6. rindā trūkst ‘ ” ‘ ” * “6. rindā trūkst URL protokola ” * “6. rindā trūkst URL lokācijas ” * “8. rindā trūkst ‘ ” ‘ ” |
| 8. | *class Class1*  *{*  *[*  *URL*  *(*  *"*  *:*  *:*  *namespaceName.*  *className#methodName*  *"*  *)*  *]*  *String str*  *()*  *;*  *}* | Kļūdas paziņojums   * “6. rindā trūkst URL protokola ” * “7. rindā trūkst URL lokācijas ” |
| 9. | *class Class1*  *{*  *[*  *URL*  *(*  *"*  *dotnet*  *:*  *local*  *;*  *namespaceName.*  *className#methodName*  *"*  *)*  *]*  *String str*  *()*  *;*  *}* | Kļūdas paziņojums   * “7. rindā URL atribūti metodei ‘str’ nav doti” |
| 10. | *class Class1*  *{*  *[*  *path*  *(*  *"*  *dotnet*  *:*  *local*  *;*  *namespaceName.*  *className#methodName*  *"*  *)*  *]*  *String str*  *()*  *;*  *}* | 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.

|  |  |  |
| --- | --- | --- |
| Testpiemēra Nr. | Apraksts | Funkcija |
| 1. | Tiek padota metode ar vienu argumentu.  Šādā gadījumā kļūdu nav. |  |
| 2. | Tiek padota metode ar diviem argumentiem, kas ir atdalīti ar komatu.  Šādā gadījumā kļūdu nav. |  |
| 3. | Tiek padota metode ar vienu argumentu, kuram ir komats pirms un pēc argumenta definīcijas.  Šādā gadījumā ir jābūt kļūdām, ka pirms pirmā komata un pēc otrā komata tiek sagaidīti argumenti.  Rinda pirmajai komata kļūdai ir atverošo iekavu rinda.  Rinda otrajai komata kļūdai ir tā paša komata rinda. |  |
| 4. | Tiek padota metode, kurai iekavās ir tikai viens komats.  Šādā gadījumā ir jābūt kļūdām, ka pirms un pēc komata tiek sagaidīti argumenti.  Rinda pirmajai komata kļūdai ir atverošo iekavu rinda.  Rinda otrajai komata kļūdai ir tā paša komata rinda. |  |
| 5. | Tiek padota metode, kurai iekavās ir divi komati.  Šādā gadījumā ir jābūt kļūdām, ka starp komatiem tiek sagaidīti argumenti (kopā trūkst 3 argumenti).  Rinda pirmajam argumentam ir atverošās iekavas rinda.  Rinda otrajam argumentam ir pirmā komata rinda.  Rinda trešajam argumentam ir otrā komata rinda. |  |
| 6. | Tiek padota metode, kurai ir divi argumenti, kurus neatdala komats.  Šādā gadījuma ir jābūt kļūdai, ka nav komata.  Rinda kļūdai ir pirmā argumenta beigu rinda. |  |
| 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   Visas šīs kļūdas ir jāfiksē.  Pirmajos trīs gadījumos kļūdas rinda ir nepareizā tipa rinda. Datu tipa iztrūkuma gadījumā rinda sakrīt ar vārda rindu un otrādi. |  |
| 8. | Tiek padota metode ar trīs argumentiem, kuriem sakrīt vārds.  Šādā gadījuma pie atkārtotiem vārdiem ir jābūt kļūdām.  Rinda sakrīt ar atkārtotā vārda rindu un vel ir jānorada rinda, kur padotais vārds jau tiek izmantots. |  |

|  |  |  |
| --- | --- | --- |
| N.P.K | Ievaddati | Sagaidāmais rezultāts |
| 1. | *class className*  *{*  *[URL("dotnet:local:*  *namespaceName.*  *className#methodName")]*  *String str*  *(*  *Integer*  *test*  *)*  *;*  *}* | Nav kļūdu |
| 2. | *class className*  *{*  *[URL("dotnet:local:*  *namespaceName.*  *className#methodName")]*  *String str*  *(*  *Integer*  *test1*  *,*  *Integer*  *test2*  *)*  *;*  *}* | Nav kļūdu |
| 3. | *class className*  *{*  *[URL("dotnet:local:*  *namespaceName.*  *className#methodName")]*  *String str*  *(*  *,*  *Integer*  *test1*  *,*  *)*  *;*  *}* | Kļūdas paziņojums   * “5. rindā trūkst argumenta” * “9. rindā trūkst argumenta” |
| 4. | *class className*  *{*  *[URL("dotnet:local:*  *namespaceName.*  *className#methodName")]*  *String str*  *(*  *,*  *)*  *;*  *}* | Kļūdas paziņojums   * “5. rindā trūkst argumenta” * “6. rindā trūkst argumenta” |
| 5. | *class className*  *{*  *[URL("dotnet:local:*  *namespaceName.*  *className#methodName")]*  *String str*  *(*  *,*  *,*  *)*  *;*  *}* | Kļūdas paziņojums   * “5. rindā trūkst argumenta” * “6. rindā trūkst argumenta” * “7. rindā trūkst argumenta” |
| 6. | *class className*  *{*  *[URL("dotnet:local:*  *namespaceName.*  *className#methodName")]*  *String str*  *(*  *Integer*  *test1*  *Integer*  *test2*  *)*  *;*  *}* | Kļūdas paziņojums   * “7. rindā trūkst komata” |
| 7. | *class className*  *{*  *[URL("dotnet:local:*  *namespaceName.*  *className#methodName")]*  *String str*  *(*  *class*  *test1*  *,*  *public*  *test2*  *,*  *Void*  *test3*  *,*  *Integer*  *,*  *test4*  *)*  *;*  *}* | 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. | *class className*  *{*  *[URL("dotnet:local:*  *namespaceName.*  *className#methodName")]*  *String str*  *(*  *Integer test1,*  *String test1,*  *Integer test1*  *)*  *;*  *}* | Kļūdas paziņojums   * “7. rindā arguments ar vārdu ‘test1’ jau eksistē. Skatīt 6. rindu” * “8. rindā arguments ar vārdu ‘test1’ jau eksistē. Skatīt 6. rindu” |