# 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.

|  |  |  |
| --- | --- | --- |
| 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” |

### 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.

|  |  |  |
| --- | --- | --- |
| Testpiemēra Nr. | Apraksts | Funkcija |
| 1. | Tiek padota divas klases ar vienu un to pašu vārdu.  Šādā gadījumā ir jābūt kļūdai, ka otrās klases vārds jau tiek izmantots.  Rinda sakrīt ar atkārtojošās klases vārda rindu. Vēl ir jānorāda rinda, kur atkārtojošais klases vārds jau ir sastopams. |  |
| 2. | Tiek padota divas klases ar dažādiem vārdiem un otrai klasei ir virsklase, kura neeksistē.  Šādā gadījumā ir jābūt kļūdai, ka otrās klases virsklase neeksistē.  Rinda sakrīt ar neeksistējošās virsklases vārda rindu. |  |
| 3. | Tiek padota klase, kuras virsklase ir pati klase.  Šādā gadījumā ir jābūt kļūdai, ka virsklase nevar būt pamatklase.  Rinda sakrīt ar “virsklases” vārdu. |  |
| 4. | Tiek padota virsklase un apakšklase, kur virsklase ir definēta pēc apakšklases.  Šādā gadījumā ir jābūt kļūdai, ka neeksistē virsklase no kuras var mantot.  Rinda sakrīt ar virsklases vārda rindu. |  |
| 5. | Tiek padota klase, kurā gan klases vārds, gan virsklases vards ir kāds no rezervētajiem vārdiem.  Šādā gadījumā ir jābūt kļūdai, ka klases un virsklases vārdi nedrīkst būt šie rezervētie vārdi.  Rindas sakrīt ar šo vārdu rindām vārda rindu. |  |

|  |  |  |
| --- | --- | --- |
| N.P.K | Ievaddati | Sagaidāmais rezultāts |
| 1. | *class*  *test1*  *{}*  *class*  *test1*  *{}* | Kļūdas paziņojums   * “6. rindā klase ‘test1’ jau eksistē. Skatīt 2. rindu.” |
| 2. | *class*  *test1*  *{}*  *class*  *test2*  *:*  *test3*  *{}* | Kļūdas paziņojums   * “8. rindā nav klases ‘test3’, no kuras var mantot” |
| 3. | *class*  *test1*  *:*  *test1*  *{}* | Kļūdas paziņojums   * “4. rindā nevar mantot no klases ‘test1’” |
| 4. | *class*  *test1*  *:*  *test2*  *{}*  *class*  *test2*  *{}* | Kļūdas paziņojums   * “4. rindā nav klases ‘test2’, no kuras var mantot” |
| 5. | *class*  *URL*  *:*  *BaseObject*  *{}* | 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.

|  |  |  |
| --- | --- | --- |
| Testpiemēra Nr. | Apraksts | 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\_”   Šādā gadījumā visiem laukiem ir jābūt kļūdainiem.  Rindas sakrīt ar lauku vārdu rindām. |  |
| 2. | Tiek padota klase ar dažādiem laukiem, kuriem sakrīt vārds. Pirmais lauks ir atribūts.  Šādā gadījumā ir jābūt kļūdai, ka lauka vārds jau tiek izmantots.  Rindas sakrīt ar atkārtojošo vārdu rindām. Jānorāda arī rinda, kurā vārds jau tiek izmantots. |  |
| 3. | Tiek padota klase ar dažādiem laukiem, kuriem sakrīt vārds. Pirmais lauks ir metode.  Šādā gadījumā ir jābūt kļūdai, ka lauka vārds jau tiek izmantots.  Rindas sakrīt ar atkārtojošo vārdu rindām. Jānorāda arī rinda, kurā vārds jau tiek izmantots. |  |
| 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.  Šādā gadījumā visām metodēm, kas ir apakšklasēs, ir jābūt kļūdai, ka metodes vārds tiek izmantots virsklasē. Atribūtam, kuram ir atšķirīgs datu tips, ir jābūt kļūdai, ka nesakrīt datu tips. Identiskajam atribūtam kļūdas nav.  Rindas sakrīt ar atkārtojošo lauku vārdu rindām. Jānorāda arī rinda, kurā vārds jau tiek izmantots. |  |
| 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.  Šādā gadījumā visiem atribūtiem, kas ir apakšklasēs, ir jābūt kļūdai, ka atribūta vārds tiek izmantots virsklasē. Metodei, kurai ir atšķirīgs datu tips, ir jābūt kļūdai, ka nesakrīt datu tips. Identiskajai metodei kļūdas nav.  Rindas sakrīt ar atkārtojošo lauku vārdu rindām. Jānorāda arī rinda, kurā vārds jau tiek izmantots. |  |
| 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.   Šādā gadījumā otrajai metodei ir kļūda, ka nesakrīt argumentu skaits, bet trešajai metodei ir kļūda, ka otrajam argumentam nav pareizs datu tips, salīdzinājumā ar virsklases metodi.  Rindas sakrīt ar lauku vārdu rindām. Argumentiem rindas sakrīt ar argumentu pozīciju, ko iegūst pēc argumenta pārbaudes. Jānorāda arī rinda, kurā vārds jau tiek izmantots. |  |
| 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  Šādā gadījumā trešās klases laukam ir jābūt kļūdai, ka lauka vārds tiek izmantots otrajā klasē.  Rindas sakrīt ar atkārtojošo lauku vārdu rindām. Jānorāda arī rinda, kurā vārds jau tiek izmantots. |  |
| 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.  Šādā gadījumā ir jābūt kļūdai, ka abu lauku vārdi tiek izmantoti.  Rinda sakrīt ar atkārtojošo vārdu rindām. Jānorāda arī rinda, kurā vārds jau tiek izmantots. |  |
| 9. | Tiek padota virsklase un apakšklase. Virsklasē ir lauks un apakšklasē ir divi lauki identiski virsklases laukam.  Šādā gadījumā ir jābūt kļūdai, ka apakšklases otrais lauks jau atkārtojas apaksklasē.  Rinda sakrīt ar atkārtojoša vārda rindu. Jānorāda arī rinda, kur vārds jau tiek izmantots apakšklasē. |  |

|  |  |  |
| --- | --- | --- |
| N.P.K | Ievaddati | Sagaidāmais rezultāts |
| 1. | *class test1*  *{*  *[URL("dotnet:local:*  *namespaceName.*  *className#methodName")]*  *public*  *Integer*  *test1*  *()*  *;*    *public*  *Integer*  *test1*  *;*    *[URL("dotnet:local:*  *namespaceName.*  *className#methodName")]*  *public*  *Integer*  *BaseObject*  *()*  *;*    *public*  *Integer*  *BaseObject*  *;*    *[URL("dotnet:local:*  *namespaceName.*  *className#methodName")]*  *public*  *Integer*  *\_constructor\_*  *()*  *;*    *public*  *Integer*  *\_constructor\_maycnwuopg*  *;*  *}* | 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. | *class test1*  *{*  *public*  *Integer*  *test*  *;*    *[URL("dotnet:local:*  *namespaceName.*  *className#methodName")]*  *public*  *Boolean*  *test*  *()*  *;*    *public*  *String*  *test*  *;*    *public*  *Integer*  *test*  *;*  *}* | 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. | *class test1*  *{*  *[URL("dotnet:local:*  *namespaceName.*  *className#methodName")]*  *public*  *Integer*  *test*  *()*  *;*    *public*  *Boolean*  *test*  *;*    *[URL("dotnet:local:*  *namespaceName.*  *className#methodName")]*  *public*  *String*  *test*  *()*  *;*    *[URL("dotnet:local:*  *namespaceName.*  *className#methodName")]*  *public*  *Integer*  *test*  *()*  *;*  *}* | 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. | *class test1*  *{*  *public*  *Integer*  *test*  *;*  *}*  *class test2 : test1*  *{*  *[URL("dotnet:local:*  *namespaceName.*  *className#methodName")]*  *public*  *Integer*  *test*  *()*  *;*    *public*  *String*  *test*  *;*    *}*  *class test3 : test2*  *{*  *public*  *Integer*  *test*  *;*    *[URL("dotnet:local:*  *namespaceName.*  *className#methodName")]*  *public*  *String*  *test*  *()*  *;*  *}* | 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. | *class test1*  *{*  *[URL("dotnet:local:*  *namespaceName.*  *className#methodName")]*  *public*  *Integer*  *test*  *()*  *;*  *}*  *class test2 : test1*  *{*  *[URL("dotnet:local:*  *namespaceName.*  *className#methodName")]*  *public*  *Integer*  *test*  *()*  *;*    *public*  *String*  *test*  *;*    *}*  *class test3 : test2*  *{*  *public*  *Integer*  *test*  *;*    *[URL("dotnet:local:*  *namespaceName.*  *className#methodName")]*  *public*  *String*  *test*  *()*  *;*  *}* | 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. | *class test1*  *{*  *[URL("dotnet:local:*  *namespaceName.*  *className#methodName")]*  *Integer*  *test*  *(Integer a,*  *Integer b,*  *Integer c)*  *;*  *}*  *class test2 : test1*  *{*  *[URL("dotnet:local:*  *namespaceName.*  *className#methodName")]*  *Integer*  *test*  *(Integer d,*  *Integer e,*  *Integer f)*  *;*    *[URL("dotnet:local:*  *namespaceName.*  *className#methodName")]*  *Integer*  *test*  *(Integer d, Integer e)*  *;*    *[URL("dotnet:local:*  *namespaceName.*  *className#methodName")]*  *Integer*  *test*  *(Boolean d,*  *String e,*  *Real f*  *)*  *;*  *}* | 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. | *class test1*  *{*  *private*  *Integer*  *test*  *;*  *}*  *class test2 : test1*  *{*  *[URL("dotnet:local:*  *namespaceName.*  *className#methodName")]*  *Integer*  *test*  *()*  *;*  *}*  *class test3 : test2*  *{*  *Integer*  *test*  *;*  *}* | Kļūdas paziņojums   * “21. rindā lauks ‘test’ jau tiek izmantots virsklasē ‘test2’. Skatīt 13. rindu.” |
| 8. | *class test1{}*  *class test2{}*  *association*  *(*  *test*  *:*  *test2*  *<->*  *target*  *:*  *test1*  *)*  *class test3 : test1*  *{*  *public*  *Integer*  *test*  *;*    *[URL("dotnet:local:*  *namespaceName.*  *className#methodName")]*  *public*  *Integer*  *test*  *()*  *;*  *}* | 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. | *class test1*  *{*  *Integer*  *test*  *;*  *}*  *class test2 : test1*  *{*  *Integer*  *test*  *;*    *Integer*  *test*  *;*  *}* | 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.

|  |  |  |
| --- | --- | --- |
| Testpiemēra Nr. | Apraksts | 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.  Šajā gadījumā ir kļūda, ka klases vārds nedrīkst būt datu tips vai bloka tips.  Rinda sakrīt ar kļūdaino klases vārdu rindām. |  |
| 2. | Tiek padota asociācija, kur kā klases vārdos tiek padota aizsardzība un rezervētais vārds, šajā gadījumā, avotā ir aizsardzība un mērķī ir rezervētais vārds.  Šajā gadījumā ir kļūda, ka klases vārds nedrīkst būt aizsardzība vai rezervētais vārds.  Rinda sakrīt ar kļūdaino klases vārdu rindām. |  |
| 3. | Tiek padota pareizi definēta asociācija, bet bez definētām klasēm.  Šajā gadījumā ir jābūt kļūdai, ka neeksistē klase, kuru var izmantot kā asociācijas klasi.  Rinda sakrīt ar klases vārdu rindām. |  |
| 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.  Šajā gadījumā ir jābūt kļūdai, ka lomu vārds nedrīkst būt bloka tips vai datu tips.  Rinda sakrīt ar kļūdaino lomu vārdu rindām. |  |
| 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.  Šajā gadījumā ir jābūt kļūdai, ka lomu vārds nedrīkst būt bloka tips vai datu tips.  Rinda sakrīt ar kļūdaino lomu vārdu rindām. |  |
| 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.  Šajā gadījumā ir jābūt kļūdai, ka avota lomas vārds nevar sakrist ar mērķa klases vārdu un ka mērķa lomas vārds nevar sakrist ar avota klases vārdu.  Rinda sakrīt ar kļūdaino lomu vārdu rindām. |  |
| 7. | Tiek padota avotklase un mērķklase, ka arī asociācija starp šīm klasēm. Lomu vārdi sākas ar “\_constructor\_”.  Šajā gadījumā ir jābūt kļūdai, ka lomu vārdi nevar sākties ar “\_cosntructor\_”  Rinda sakrīt ar kļūdaino lomu vārdu rindām. |  |
| 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.  Šajā gadījumā ir jābūt kļūdai, ka lomu vārdi jau tiek izmantoti  Rinda sakrīt ar kļūdaino lomu vārdu rindām. Jānorāda arī lauku rindas, kuras atkārtojas. |  |
| 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.  Šajā gadījumā ir jābūt kļūdai, ka otrās asociācijas avota lomas vārds jau tiek izmantots.  Rinda sakrīt ar kļūdainā lomas vārda rindu. Jānorāda arī lomas vārda, kurs atkārtojas, 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.  Šajā gadījumā ir jābūt kļūdai, ka otrās asociācijas mērķa lomas vārds jau ir sastopams virsklasē.  Rinda sakrīt ar atkārtojošā lomas vārda rindu. Jānorāda arī rindu, kur lomas vārds jau ir sastopams. |  |
| 11. | Testpiemērs ir identisks 10. testpiemēram, tikai asociācijas ir definētas apgrieztā secībā.  Šajā gadījumā ir jābūt kļūdai, ka otrās asociācijas mērķa lomas vārds jau ir sastopams apakšklasē.  Rinda sakrīt ar atkārtojošā lomas vārda rindu. Jānorāda arī rindu, kur lomas vārds jau ir sastopams. |  |
| 12. | Sarežģītāks gadījums 10. un 11. testpiemēram. Skatīt testpiemēru tabulu. |  |
| 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.  Šādā gadījumā ir kļūda, ka lomas vārds jau tiek izmantots kā virsklases lauka vārds.  Rinda sakrīt ar atkārtojošā lomas vārda rindu. Jānorāda arī rindu, kur ir sastopams attribūts. |  |
| 14. | Testpiemērs ir identisks 13. testpiemēram, tikai pirmajā klase ir metode nevis atribūts un tas privāts.  Šajā gadījumā kļūdu nav. |  |
| 15. | Testpiemērs ir identisks 13. testpiemēram, tikai lauks ir definēts trešajā klasē, atribūts ir privāts un asociācija ir starp pirmo un ceturto klasi. |  |
| 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. |  |
| 17. | Tiek padota viena klase un asociācija, kur klases vārdi sakrīt, bet lomu vārdi nē.  Šādā gadījumā kļūdu nav. |  |
| 18. | Tiek padota viena klase un asociācija, kur klases vārdi un lomu vārdi sakrīt.  Šādā gadījumā jābūt kļūda, ka mērķa lomas vārds jau tiek izmantots klasē.  Rinda sakrīt ar mērķa lomas vārda rindu. Jānorāda arī avota lomas vārda rindu. |  |
| 19. | Tiek padota virsklase un apakšklase un asociācija starp šīm klasēm, kur lomu vārdi nesakrīt.  Šādā gadījumā kļūdu nav. |  |
| 20. | Tiek padota virsklase un apakšklase un asociācija starp šīm klasēm, kur lomu vārdi sakrīt. Avotā ir virsklases vārds.  Šādā gadījumā jābūt kļūda, ka mērķa lomas vārds jau tiek izmantots virsklasē.  Rinda sakrīt ar mērķa lomas vārda rindu. Jānorāda arī avota lomas vārda rindu. |  |
| 21. | Tiek padota virsklase un apakšklase un asociācija starp šīm klasēm, kur lomu vārdi sakrīt. Avotā ir apakšklases vārds.  Šādā gadījumā jābūt kļūda, ka mērķa lomas vārds jau tiek izmantots apakšklasē.  Rinda sakrīt ar mērķa lomas vārda rindu. Jānorāda arī avota lomas vārda rindu. |  |

|  |  |  |
| --- | --- | --- |
| N.P.K | Ievaddati | Sagaidāmais rezultāts |
| 1. | *association*  *(*  *sourceName*  *:*  *class*  *<->*  *targetName*  *:*  *Integer*  *)* | Kļūdas paziņojumi   * “5. rindā avota klases vārds nevar būt ‘class’.” * “9. rindā mērķa klases vārds nevar būt ‘Integer’.” |
| 2. | *association*  *(*  *sourceName*  *:*  *public*  *<->*  *targetName*  *:*  *URL*  *)* | Kļūdas paziņojumi   * “5. rindā avota klases vārds nevar būt ‘public’.” * “9. rindā mērķa klases vārds nevar būt ‘URL’.” |
| 3. | *association*  *(*  *sourceName*  *:*  *sourceClass*  *<->*  *targetName*  *:*  *targetClass*  *)* | Kļūdas paziņojumi   * “5. rindā nav klases ‘sourceClass’, ko izmantot kā avota klasi.” * “9. rindā nav klases ‘targetClass’, ko izmantot kā mērķa klasi.” |
| 4. | *class sourceClass {}*  *class targetClass {}*  *association*  *(*  *class*  *:*  *sourceClass*  *<->*  *Integer*  *:*  *targetClass*  *)* | Kļūdas paziņojumi   * “6. rindā avota lomas vārds nevar būt ‘class’.” * “10. rindā mērķa lomas vārds nevar būt ‘Integer’.” |
| 5. | *class sourceClass {}*  *class targetClass {}*  *association*  *(*  *public*  *:*  *sourceClass*  *<->*  *BaseObject*  *:*  *targetClass*  *)* | Kļūdas paziņojumi   * “6. rindā avota lomas vārds nevar būt ‘public’.” * “10. rindā mērķa lomas vārds nevar būt ‘BaseObject’.” |
| 6. | *class sourceClass {}*  *class targetClass {}*  *association*  *(*  *targetClass*  *:*  *sourceClass*  *<->*  *sourceClass*  *:*  *targetClass*  *)* | Kļūdas paziņojumi   * “6. rindā avota lomas vārds nevar būt ‘targetClass’.” * “10. rindā mērķa lomas vārds nevar būt ‘sourceClass’.” |
| 7. | *class sourceClass {}*  *class targetClass {}*  *association*  *(*  *\_constructor\_*  *:*  *sourceClass*  *<->*  *\_constructor\_iylagbek*  *:*  *targetClass*  *)* | Kļūdas paziņojumi   * “6. rindā avota lomas vārds nevar sākties ar ‘\_constructor\_’.” * “10. rindā mērķa lomas vārds nevar sākties ar ‘\_constructor\_’.” |
| 8. | *class sourceClass*  *{*  *Integer*  *test1*  *;*  *}*  *class targetClass*  *{*  *[URL("dotnet:local:*  *namespaceName.*  *className#methodName")]*  *String*  *test2*  *()*  *;*  *}*  *association*  *(*  *test2*  *:*  *sourceClass*  *<->*  *test1*  *:*  *targetClass*  *)* | 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. | *class sourceClass {}*  *class targetClass {}*  *class sourceClass2 {}*  *association*  *(*  *test2*  *:*  *sourceClass*  *<->*  *test1*  *:*  *targetClass*  *)*  *association*  *(*  *test2*  *:*  *sourceClass2*  *<->*  *test1*  *:*  *targetClass*  *)* | Kļūdas paziņojums   * “24. rindā lomas vārds ‘test2’ jau tiek izmantots klasē ‘targetClass’. Skatīt 13. rindu.” |
| 10. | *class test1{}*  *class test2 : test1{}*  *class test3 : test2{}*  *class test4 {}*  *association*  *(*  *sourceName1*  *:*  *test1*  *<->*  *targetName*  *:*  *test4*  *)*  *association*  *(*  *sourceName3*  *:*  *test3*  *<->*  *targetName*  *:*  *test4*  *)* | Kļūdas paziņojums   * “25. rindā lomas vārds ‘test4’ jau tiek izmantots virsklasē ‘test1’. Skatīt 14. rindu.” |
| 11. | *class test1{}*  *class test2 : test1{}*  *class test3 : test2{}*  *class test4 {}*  *association*  *(*  *sourceName3*  *:*  *test3*  *<->*  *targetName*  *:*  *test4*  *)*  *association*  *(*  *sourceName1*  *:*  *test1*  *<->*  *targetName*  *:*  *test4*  *)* | Kļūdas paziņojums   * “25. rindā lomas vārds ‘test4’ jau tiek izmantots apakšklasē ‘test3’. Skatīt 14. rindu.” |
| 12. | *class test1{}*  *class test2 : test1{}*  *class test3 : test1{}*  *class test4 : test1{}*  *class test5 : test4{}*  *class test6 {}*  *association(sourceName2:test2*  *<->targetName:test6)*  *association(sourceName4:test4*  *<->targetName:test6)*  *association(sourceName5:test5*  *<->targetName:test6)*  *association(sourceName1:test1*  *<->targetName:test6)* | 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. | *class test1*  *{*  *public*  *Integer*  *targetName;*  *}*  *class test2 : test1{}*  *class test3 : test2{}*  *class test4 {}*  *association*  *(*  *sourceName3*  *:*  *test3*  *<->*  *targetName*  *:*  *test4*  *)* | Kļūdas paziņojums   * “17. rindā lomas vārds ‘targetName’ jau tiek izmantots virsklasē ‘test1’. Skatīt 5. rindu.” |
| 14. | *class test1*  *{*  *[URL("dotnet:local:*  *namespaceName.*  *className#methodName")]*  *private*  *Integer*  *targetName*  *()*  *;*  *}*  *class test2 : test1{}*  *class test3 : test2{}*  *class test4 {}*  *association*  *(*  *sourceName3*  *:*  *test3*  *<->*  *targetName*  *:*  *test4*  *)* | Nav kļūdu |
| 15. | *class test1{}*  *class test2 : test1{}*  *class test3 : test2*  *{*  *private*  *Integer*  *targetName;*  *}*  *class test4 {}*  *association*  *(*  *sourceName1*  *:*  *test1*  *<->*  *targetName*  *:*  *test4*  *)* | Kļūdas paziņojums   * “17. rindā lomas vārds ‘targetName’ jau tiek izmantots apakšklasē ‘test3’. Skatīt 7. rindu.” |
| 16. | *class test1{}*  *class test2 : test1{}*  *class test3 : test2*  *{*  *[URL("dotnet:local:*  *namespaceName.*  *className#methodName")]*  *public*  *Integer*  *targetName*  *()*  *;*  *}*  *class test4 {}*  *association*  *(*  *sourceName1*  *:*  *test1*  *<->*  *targetName*  *:*  *test4*  *)* | Kļūdas paziņojums   * “20. rindā lomas vārds ‘targetName’ jau tiek izmantots apakšklasē ‘test3’. Skatīt 8. rindu.” |
| 17. | *class test1{}*  *association*  *(*  *source*  *:*  *test1*  *<->*  *target*  *:*  *test1*  *)* | Nav kļūdu |
| 18. | *class test1{}*  *association*  *(*  *role*  *:*  *test1*  *<->*  *role*  *:*  *test1*  *)* | Kļūdas paziņojums   * “9. rindā lomas vārds ‘role’ jau tiek izmantots klasē ‘test1’. Skatīt 5. rindu.” |
| 19. | *class test1{}*  *class test2 : test1 {}*  *association*  *(*  *sourceName*  *:*  *test1*  *<->*  *targetName*  *:*  *test2*  *)* | Nav kļūdu |
| 20. | *class test1{}*  *class test2 : test1 {}*  *association*  *(*  *role*  *:*  *test1*  *<->*  *role*  *:*  *test2*  *)* | Kļūdas paziņojums   * “10. rindā lomas vārds ‘role’ jau tiek izmantots virsklasē ‘test1’. Skatīt 6. rindu.” |
| 21. | *class test1{}*  *class test2 : test1 {}*  *association*  *(*  *role*  *:*  *test2*  *<->*  *role*  *:*  *test1*  *)* | Kļūdas paziņojums   * “10. rindā lomas vārds ‘role’ jau tiek izmantots apakšklasē ‘test1’. Skatīt 6. rindu.” |

#### 4.2.1.5 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 |