

DOMENSKO SPECIFIČNI MODELIRNI JEZIKI 2024/2025

6. predavanje

Povzetek

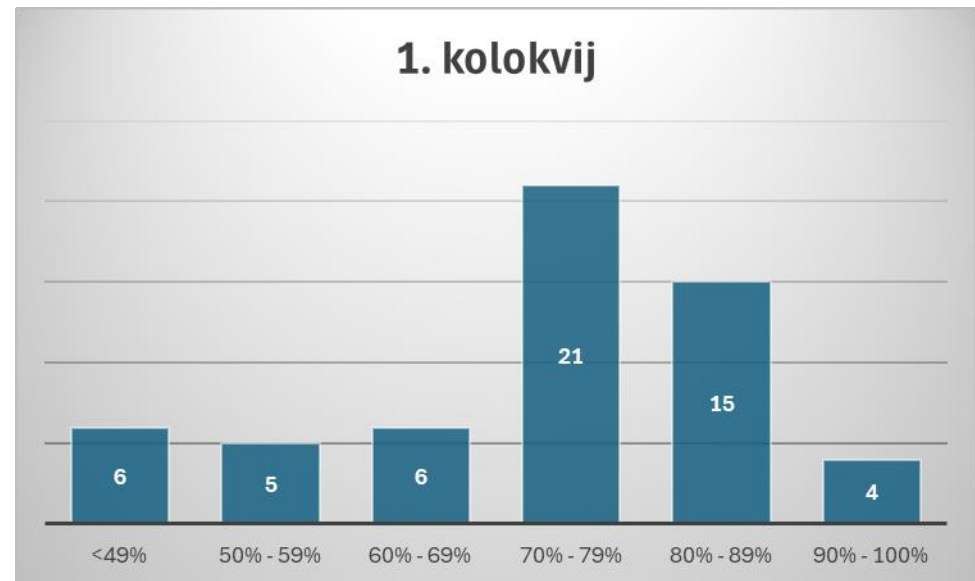
- 1. kolokvij
- Moderni tekstovni urejevalniki
- MPS
 - Označevanje sintakse
 - Dejanja
 - Transformacijski meniji
 - Trenutni kontekst
 - Omejitve
 - Omejitve s sistemom tipov
 - Hitri popravki
 - Projekcijski urejevalniki



1. KOLOKVIJ

Rezultati

- 45 minut, 17 vprašanj
- 1. vmesni izpit
 - Pisalo: 57 / 69 študentov
 - Pozitivnih: 51/57 (89,5%)
 - Nad 35%: 56/57 (98,2%)
 - Nad 90%: 4
 - Pod 50%: 6
 - Pod 35%: 1
- Najboljša ocena
 - 95%
- Povprečni rezultat
 - 71,54%





NALOGE IZ 1. KOLOKVIJA

Vprašanja iz 1. kolokvija

1) Domensko specifični jeziki spadajo **v katero generacijo** programskih jezikov (1-5)?

(5%)

Vprašanja iz 1. kolokvija

2) Katere zahteve mora izpolnjevati programski jeziki? (3) (5%)

Vprašanja iz 1. kolokvija

3) Zapišite en sinonim za domensko-specifične jezike (lahko v angl. jeziku)? (5%)

Vprašanja iz 1. kolokvija

4) Eden izmed vzorcev odločitve za razvoj DSLjev je notacija. Opišite ga. (5%)

Vprašanja iz 1. kolokvija

9) Kaj predstavlja **concept** v MPS? (5%)



Vprašanje iz 1. kolokvija

15) Spodnji program ne ustreza podanemu zapisu EBNF. Zapišite zakaj ne oz. popravite gramatiko tako, da bo pravilna. (10%)

Primer programa

```
count = count + 1
```

```
dir = newDir ;
```

Gramatika

```
Assign      ::=  ID          AssignOper      AssignExpr      ';'
AssignOper ::=  '='
AssignExpr ::=  UnaryExpr   (          ExprOper      UnaryExpr   )+
ExprOper   ::=  '+' |      '-' |      '*' |      '/'
UnaryExpr  ::=  INT | ID
```



Anatomija modernih razvojnih okolij

The screenshot displays a development environment with the following components and callouts:

- Project Explorer:** Shows the project structure, including 'SCAD_CAR', 'CarLanguage.runtime', 'CarLanguage.sandbox', and 'MyFirstCarProgram'.
- Logical View:** Displays the code for 'MyFirstCarProgram' with a callout: **Pomoč uporabniku** (User help).
- Code Editor:** Shows the code for 'MyFirstCarProgram' with a callout: **Poudarjanje ključnih besed** (Highlighting key words).
- Inspector:** Shows the properties of the selected code element, with a callout: **Dokončanje kode** (Code completion).
- Node Explorer:** Shows the node structure, with a callout: **Pogled obrisa kode** (Code outline view).
- Inspector:** Shows the properties of the selected code element, with a callout: **Opis napak** (Error description).
- Inspector:** Shows the properties of the selected code element, with a callout: **Javljanje napak** (Error reporting).

```
car MyFirstCarProgram some name {
  carbody Limo
  color black
  roof black
  roof rails black
  sunroof
  wheels 195/60R16
}
```

Node Explorer: my

- my
- Concept: Car
- properties
- name = my
- references
- body: CarBody
- Concept: CarBody
- properties
- carbody = Limo
- references
- color: Color
- Concept: Color
- properties
- references
- bodyColor: BodyColor
- wheels: <no name> [Wheels]
- Concept: Wheels
- properties
- references

Inspector: CarLanguage.structure.Car

[root] Car "MyFirstCarProgram"[6415494402663...26] in CarLanguage.sandbox

Integrirana razvojna okolja

- IDE (integrated development environment)
- Programska oprema za **razvijalce**, ki omogoča **učinkovito pisanje, preizkušanje in odpravljanje napak**
- Glavne značilnosti
 - Urejevalnik kode
 - Razhroščevalnik
 - Prevajalnik/Interpreter
- Namen
 - Poenostavi razvojni proces z zagotavljanjem okolja vse v enem.
- Primeri
 - Visual Studio, IntelliJ IDEA, Visual Studio Code, Eclipse itd.

Integrirana razvojna okolja in empirične raziskave

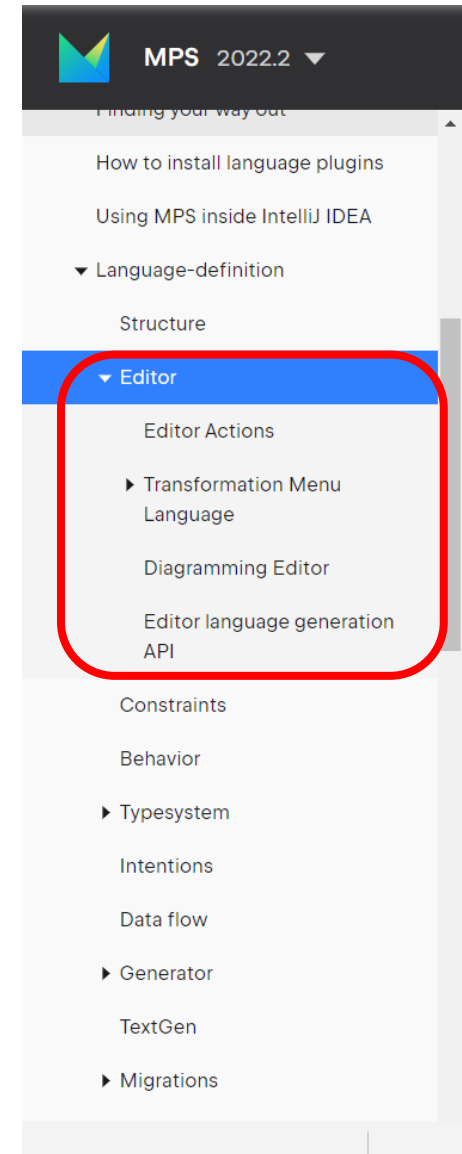
- Programerji so pri uporabi integriranih razvojnih okolij (IDE) **učinkovitejši** v primerjavi z uporabo tradicionalnih urejevalnikov besedila.
- Prednosti IDE-jev izhajajo predvsem iz integriranih orodij, ki zmanjšujejo **ponavljajoča se opravila**, omogočajo boljše odpravljanje **napak** in povečujejo **produktivnost**.
 - Programerji, ki uporabljajo IDE-je so naloge opravili **20–50 % hitreje** (EMSE, 2018).
 - Razvijalci z IDE-ji in vgrajenimi orodji za odpravljanje napak so **napake odkrivali** in odpravljali **30 % učinkoviteje** (Auckland, 2016).
 - **Začetniki**, ki so uporabljali IDE-je z možnostjo samodokončanja in sprotnega preverjanja napak, so **hitreje napredovali** (ACM TCE, 2020)
 - Več kot 80 % od 20.000 razvijalcev je navedlo, da funkcije, kot so **navigacija** po kodi, integracija z orodji za **deljenje kode** in orodja za **preurejanje**, znatno povečajo njihovo produktivnost (Jetbrains, 2020)

Prednosti IDEjev

- **Avtomatizacija**
 - IDE-ji avtomatizirajo ponavljajoča se opravila, kot so prevajanje, testiranje in preurejanje kode.
- **Preprečevanje napak**
 - Sproti preverjajo sintakso in predlagajo samodokončanje, kar zmanjšuje verjetnost napak.
- **Manj preklapljanja med orodji**
 - Integrirana orodja (npr. Git, razhroščevalniki) zmanjšujejo potrebo po preklapljanju med aplikacijami.
- **Boljša berljivost kode**
 - Funkcije, kot so barvno označevanje sintakse in formatiranje kode, izboljšajo berljivost in sodelovanje v ekipi.

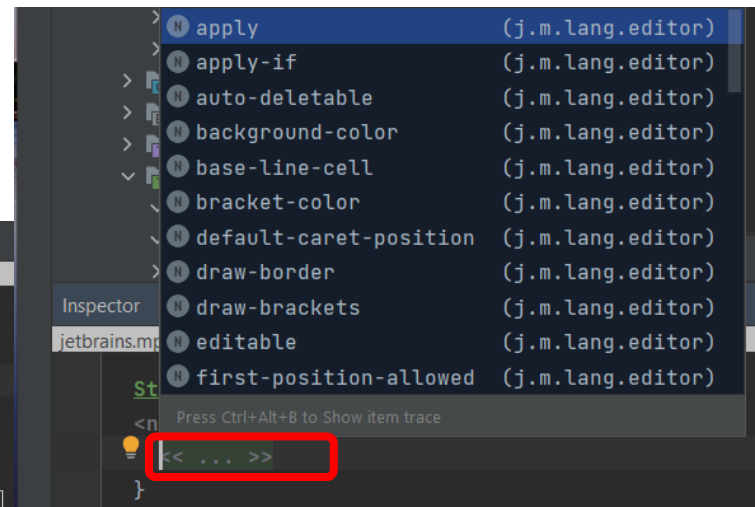
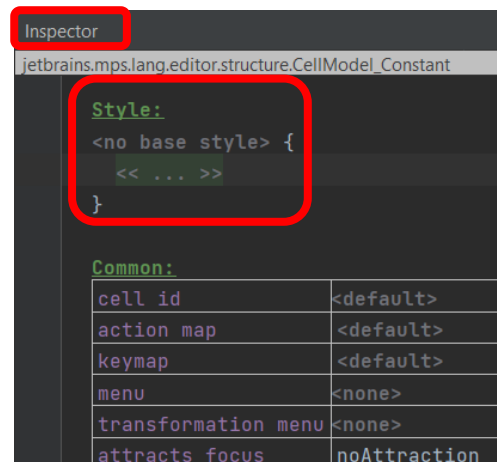
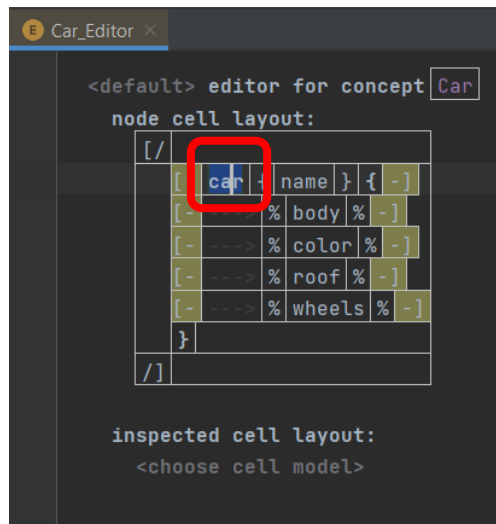
MPS: urejevalnik koncepta (spomnimo)

- Urejevalnik (angl. editor)
- Oblikovalec jezika ustvari urejevalnik **za vsak koncept** v jeziku.
- V MPS je urejevalnik sestavljen iz **celic**, ki vsebujejo:
 - druge celice,
 - besedilo ali
 - komponento uporabniškega vmesnika.
- Vsak koncept ima **največ en** urejevalnik.
 - Če ga nima, je urejevalnik koncepta določen z **njegovim staršem**
- Vir: [link](#)



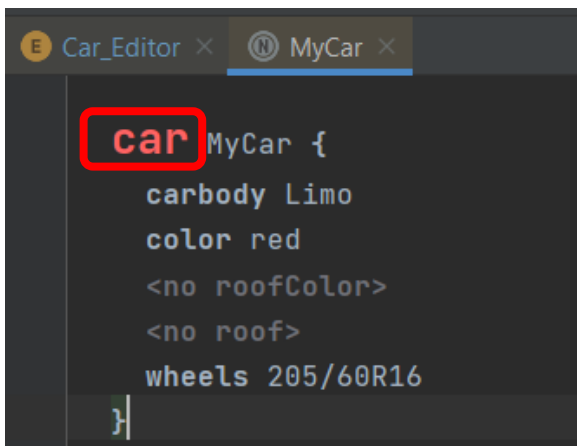
MPS podpora za prilagoditev urejevalnika: slogi

- Označevanje sintakse (angl. syntax highlighting)
 - Editor → inspector (desni klik, inspect)
 - Pogled inspector prikazuje **dodatne nastavitve** v zvezi s celico, izbrano v urejevalniku koncepta.
 - Npr. oblikovalcu jezika omogoča, da celice uredi s **slogi** (angl. styles).

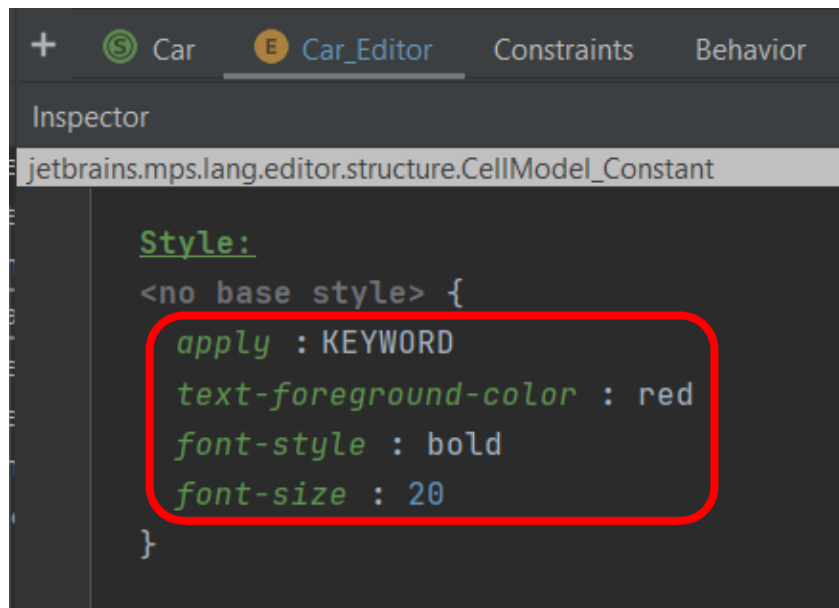


MPS podpora za prilagoditev urejevalnika: slogi

- Oblikovanje celic urejevalnika predstavlja zelo močan način za **izboljšanje berljivosti kode**.
 - Jezikovni elementi prikazani v različnih barvah ali pisavah, razvijalcem pomaga **lažje razumeti sintakso**.
- Kaj lahko s stili uredimo?
 - Ključne besede, konstante, klice, definicije, izraze, komentarje, itd.
- Več: [link](#)



```
car MyCar {  
  carbody Limo  
  color red  
  <no roofColor>  
  <no roof>  
  wheels 205/60R16  
}
```



```
Style:  
<no base style> {  
  apply : KEYWORD  
  text-foreground-color : red  
  font-style : bold  
  font-size : 20  
}
```

MPS podpora za prilagoditev urejevalnika: dejanja

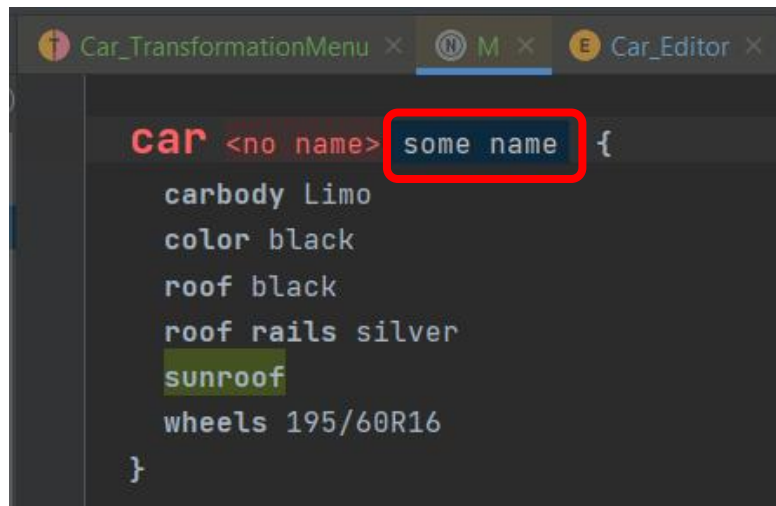
- Urejevalnik MPS ima precej senzibilne privzete nastavitve za **dejanja** (angl. editor actions).
 - Npr. ustvarjanje vozlišč, copy-paste besedila
- Dejanja je možno prilagoditi - uporaba „action language“.
- V projekt dodamo **Actions Aspect**
- Več na: [link](#)

MPS podpora za prilagoditev urejevalnika: transformacijski meniji

- Definiranje **menijev**
- **Preoblikovanje** programske kode
- Preoblikovanje pomeni, da **zamenjamo vozlišče** v AST z novim
 - Novo vozlišče mora biti **kompatibilno** s pozicijo, kjer ga želimo v strukturi AST vstaviti
- Uporaba jezika za transformacije menijev (angl. **transformation menu language**)
- Meniji:
 - side transform menus,
 - substitute menus,
 - **context assistant**,
 - context actions tool.

MPS podpora za prilagoditev urejevalnika: transformacijski meniji

- Pomočnik za trenutni kontekst (angl. Context Assist)
 - Dinamično **sestavljen meni z dejanji**, ki so najbolj primerna za **trenutni kontekst**
 - Ogradne oznake (angl. **placeholders**) v definiciji urejevalnika
- Uporaba
 - **Ctrl + Alt + Enter** (klik miške)
 - Premikanje po meniju (smerne tipke, miška, tabulator)
 - Izberemo pomoč (preslednica, enter, miška)

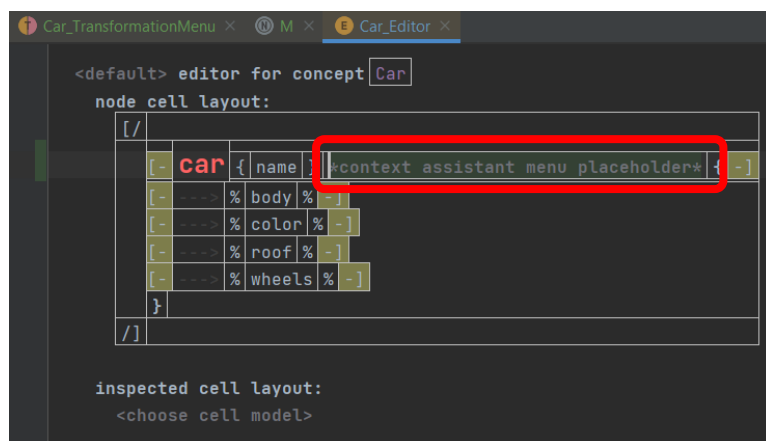


```
car <no name> some name {  
  carbody Limo  
  color black  
  roof black  
  roof rails silver  
  sunroof  
  wheels 195/60R16  
}
```



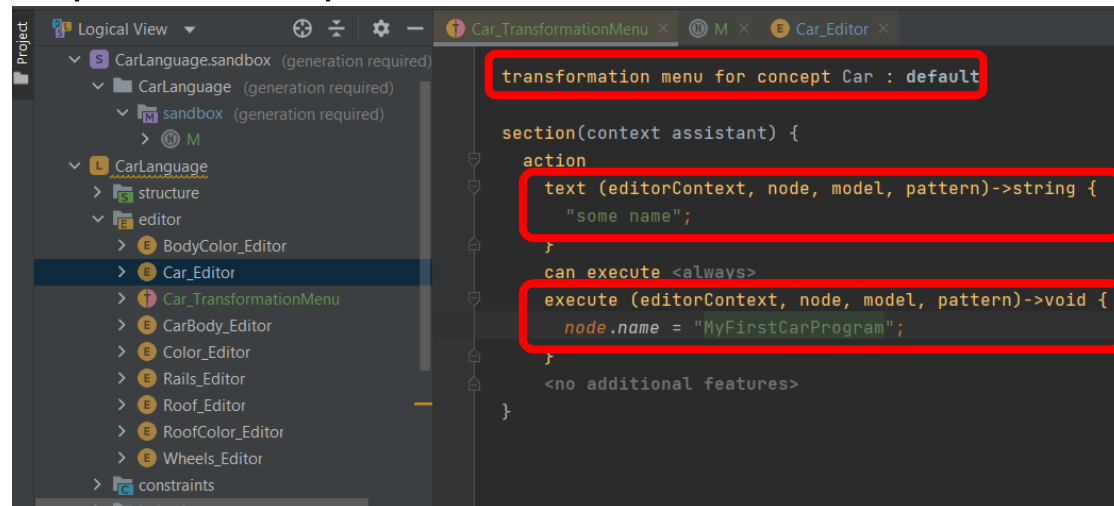
MPS podpora za prilagoditev urejevalnika: meniji

- Editor → Transformation menu
 - Pomočnik za trenutni kontekst (angl. Context Assist)
 - Več: [link](#)
 - Tutorial: [link](#)
- Definiramo jo z:
 - Ogradno oznako (angl. placeholder)
 - Opisom (angl. action), ki ga vidimo
 - Izvedbo (angl. execute), ki poskrbi za spremembo kode



```
<default> editor for concept Car
node cell layout:
[ /
  [- car { name } | *context assistant menu placeholder* | [- ]
  [- --> % body % -]
  [- --> % color % -]
  [- --> % roof % -]
  [- --> % wheels % -]
]
/ ]

inspected cell layout:
<choose cell model>
```



```
Logical View
CarLanguage.sandbox (generation required)
  CarLanguage (generation required)
    sandbox (generation required)
      M
        CarLanguage
          structure
            editor
              BodyColor_Editor
              Car_Editor
              Car_TransformationMenu
              CarBody_Editor
              Color_Editor
              Rails_Editor
              Roof_Editor
              RoofColor_Editor
              Wheels_Editor
          constraints
```

```
transformation menu for concept Car : default

section(context assistant) {
  action
    text (editorContext, node, model, pattern)->string {
      "some name";
    }
    can execute <always>
    execute (editorContext, node, model, pattern)->void {
      node.name = "MyFirstCarProgram";
    }
  <no additional features>
}
```

MPS podpora za prilagoditev urejevalnika: trenutni kontekst

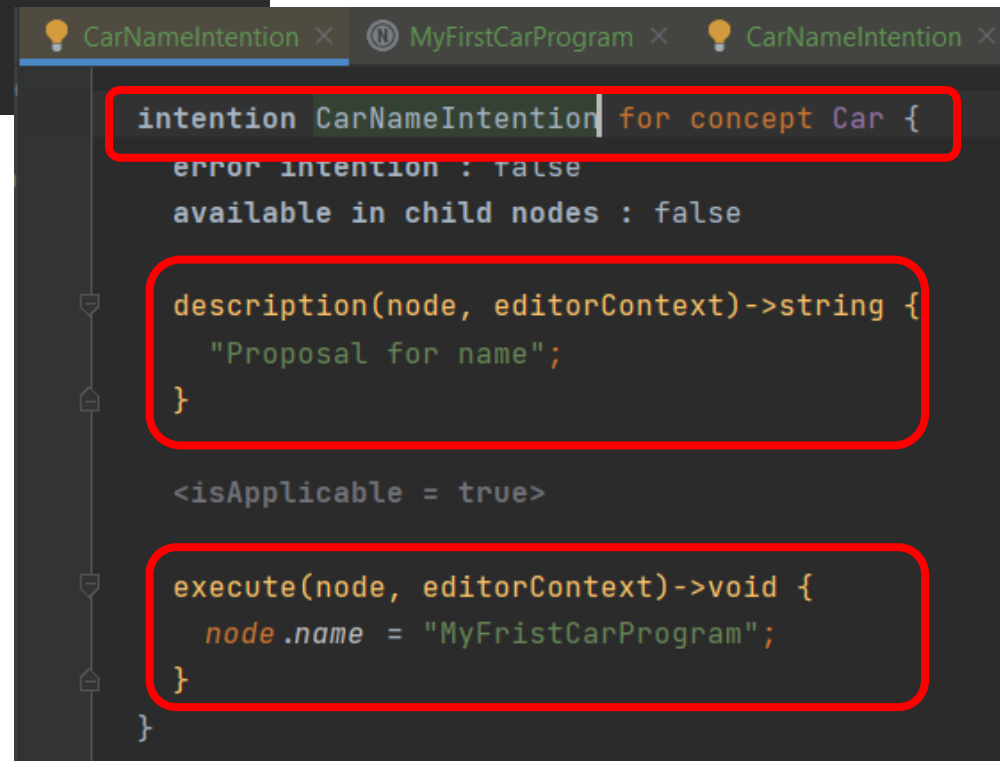
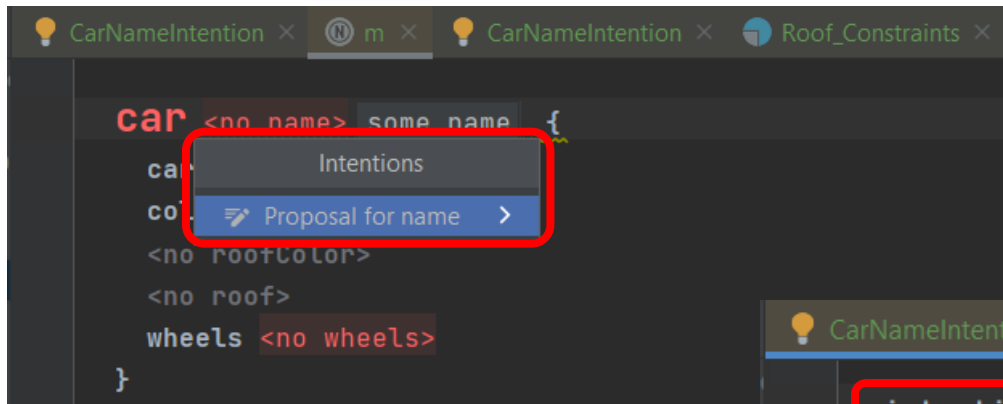
- MPS ponuja več mehanizmov za izvajanje **dejanja v trenutnem kontekstu**:
 - pomočnik za trenutni kontekst (angl. Context Assist) – že pogledali
 - dokončanje kode (angl. code completion, **Ctrl + Space**),
 - **namere** (angl. intention),
 - preoblikovanje (angl. refactorings) in
 - pojavni meniji (angl. popup menus).
- Namere uporabljamo v **dva namena**:
 - Niso takoj vidni novim, neizkušenim uporabnikom, namenjeni **naprednim uporabnikom**
 - Vodenje **novih uporabnikov** skozi pisanje programov

MPS podpora za prilagoditev urejevalnika: trenutni kontekst

- Namera (angl. intention) **izboljšajo programerjevo izkušnjo**
- Namere omogočajo **hiter dostop do najpogostejše uporabljenih operacij** v priročnem meniju (**Alt+Enter**)
 - Podobno uporabi v drugih orodjih
- Uporaba namere sproži spremembo programske kode
- V projekt dodamo **Intention Action**
- Definiramo jo z:
 - **Opisom** (angl. description), ki ga vidimo v priročnem meniju
 - **Izvedbo** (angl. execute), ki poskrbi za spremembo kode
- Več: [link](#)

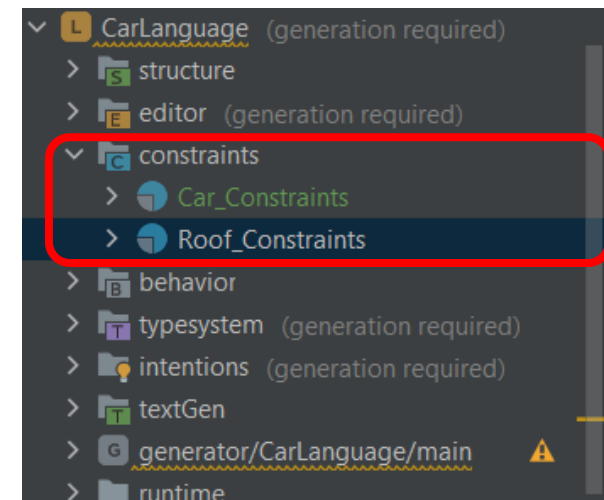


MPS podpora za prilagoditev urejevalnika: trenutni kontekst



MPS podpora za prilagoditev urejevalnika: omejitve v MPS

- **Statična** analiza kode
 - Določena z gramatiko in leksikalnimi pravili oz. v JetBrains MPS s strukturo (koncepti)
 - Vozlišča po meri, npr. STRING, INT
- **Dinamična** analiza kode
 - Izključevanje vozlišč, itd.
- Strukturni jezik v MPS (zapis konceptov) včasih **ne zadostuje za izražanje naprednih omejitev** v jeziku.
 - Omejitve (angl. **constraints**) omogočajo definiranje dodatnih omejitev.

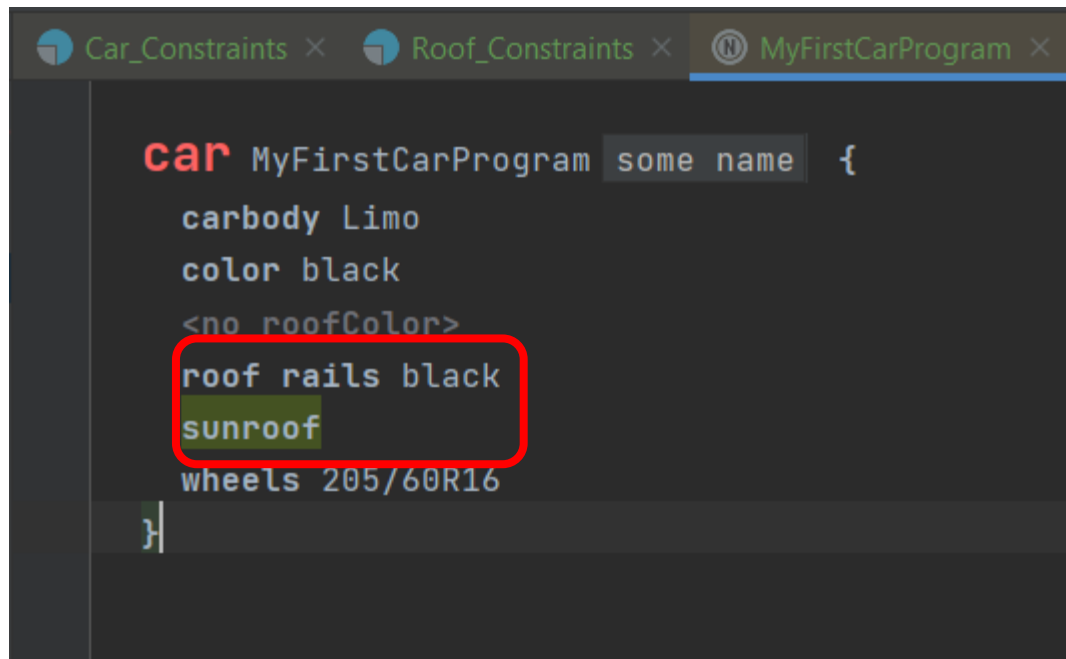


MPS podpora za prilagoditev urejevalnika: omejitve v MPS

- Omejitve določamo za **koncept**
- Omejimo kdaj je vozlišče možno **priklopiti v model** kot otroka/starša/prednika/ikono (**child, parent, ancestor, icon**)
- Vsaka omejitev **vrača boolean** logično vrednost
 - MPS jo pokliče **vsakič**, ko ocenjuje dovoljeno pozicijo za to vozlišče v AST

MPS podpora za prilagoditev urejevalnika: primer programa z napako

- Koncepta sta si izključujoča



```
Car_Constraints x Roof_Constraints x MyFirstCarProgram x
car MyFirstCarProgram some name {
  carbody Limo
  color black
  <no_roofColor>
  roof rails black
  sunroof
  wheels 205/60R16
}
```

The screenshot shows an MPS editor window titled 'MyFirstCarProgram'. The code defines a car configuration. A red rectangle highlights the lines 'roof rails black' and 'sunroof', indicating a conflict between these two mutually exclusive options. The 'sunroof' line is highlighted in green, suggesting it is the active or selected option.

MPS podpora za prilagoditev urejevalnika: zapis lastne omejitve

```
Car_Constraints x MyFirstCarProgram x Roof_Constraints x
concepts constraints Roof {
  can be child (node, parentNode, childConcept, link)->boolean {
    (node.rails.isNotNull && node.sunroof.isNull) || (node.rails.isNull && node.sunroof.isNotNull)
    || (node.rails.isNull && node.sunroof.isNull);
  }

  can be parent <none>

  can be ancestor <none>

  instance icon <none>

  <<property constraints>>

  <<referent constraints>>

  default scope <no default scope>
}
```



MPS podpora za prilagoditev urejevalnika: prikaz delovanja omejitve

```
Car_Constraints x MyFirstCarProgram x Roof_Constraints x
car MyFirstCarProgram {
  carbody Limo
  color black
  <no roofColor>
  roof rails black
  sunroof
  wheels 205/60R16
}
```

```
Car_Constraints x MyFirstCarProgram x Roof_Constraints x
car MyFirstCarProgram some name {
  carbody Limo
  color black
  <no roofColor>
  roof rails black
  <no sunroof>
  wheels 205/60R16
}
```

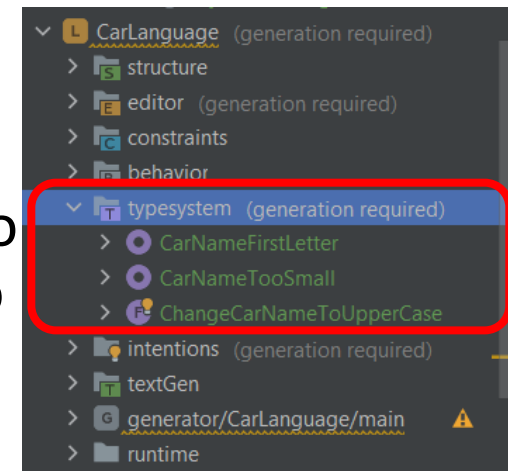
```
Car_Constraints x MyFirstCarProgram x Roof_Constraints x
car MyFirstCarProgram some name {
  carbody Limo
  color black
  <no roofColor>
  <no rails>
  sunroof
  wheels 205/60R16
}
```

```
Car_Constraints x MyFirstCarProgram x Roof_Constraints x
car MyFirstCarProgram some name {
  carbody Limo
  color black
  <no roofColor>
  <no rails>
  <no sunroof>
  wheels 205/60R16
}
```



MPS podpora za prilagoditev urejevalnika: omejitve s sistemom tipov

- Sistem tipov je del definicije jezika, ki **dodeljuje tipe vozliščem** v modelih.
 - V MPS uporabljamo **jezik typesystem** za javljanje **težav**
- Se uporablja tudi za **preverjanje omejitev** vozlišč (ne gre za napake sistema tipov)
 - Dodamo pravila preverjanja (angl. **checking rule**)
 - **Statična** analiza kode
 - Naslednje kategorije omejitev: pravilnost, večnitna pravilnost, težave z i8N, napake ranljivosti kode, težave s slogom, težave z zmogljivostjo
- MPS loči **težave** po resnosti:
 - **napake** (ang. error): prikazane rdeče
 - **opozorila** (angl. warning): prikazana rumeno
 - **informacije** (angl. infos): prikazano sivkasto





MPS podpora za prilagoditev urejevalnika: omejitve s sistemom tipov

The screenshot displays the MPS IDE interface. The top editor shows the definition of the `CarNameFirstLetter` rule, which is highlighted with a red box. The rule is applicable for the `Car` concept and checks if the car's name starts with a lowercase letter, issuing a warning if it does.

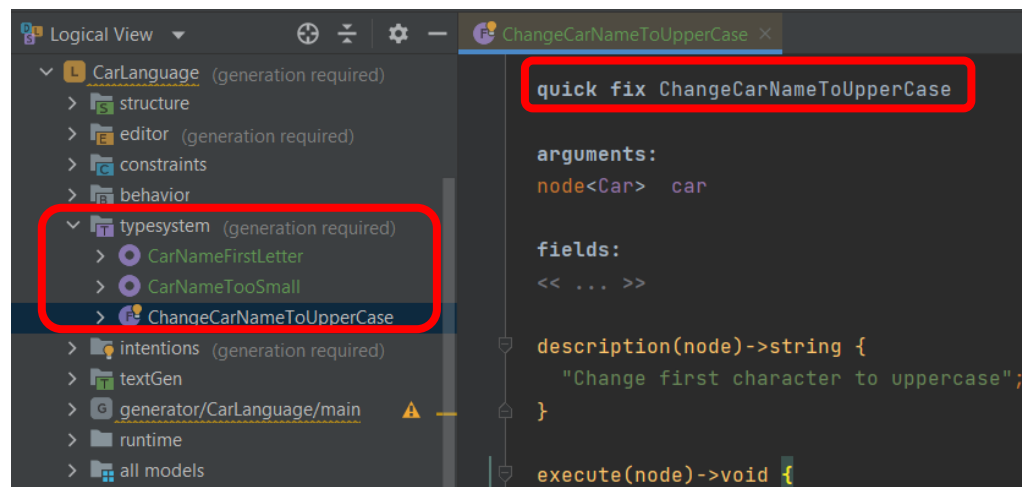
```
checking rule CarNameFirstLetter {  
  applicable for concept = Car as car  
  overrides <none>  
  do not apply on the fly false  
  
  do {  
    if (car.name.isNotEmpty && Character.isLowerCase(car.name.charAt(0))) {  
      info "Name must start with uppercase!" -> car ;  
    }  
  }  
}
```

The bottom editor shows the application of this rule to a car model named `myName`. The model is highlighted with a red box, and a warning message is displayed: "Warning: Name must start with uppercase!". The model's attributes are listed below:

```
car myName  
  carbody Limo  
  color black  
  <no roofColor>  
  <no rails>  
  <no sunroof>  
  wheels 205/60R16  
}
```


MPS podpora za prilagoditev urejevalnika: hitri popravki

- Hitri popravki (angl. **quick-fix**) so pomoč, ki jo nudimo uporabniku jezika, da hitro **popravi težavo** v programu
- Hitri popravek lahko **zapišemo za omejitve v sistemu tipov** (angl. checking rule)
- Zapišemo v typesystem → quickfix





MPS podpora za prilagoditev urejevalnika: hitri popravki

```
ChangeCarNameToUpperCase x CarNameFirstLetter x
quick fix ChangeCarNameToUpperCase

arguments:
node<Car> car

fields:
<< ... >>

description(node)->string {
  "Change first character to uppercase";
}

execute(node)->void {
  car.name = Character.toUpperCase(car.name.charAt(0)) + car.name.substring(1);
}
```

```
ChangeCarNameToUpperCase x CarNameFirstLetter x

checking rule CarNameFirstLetter {
  applicable for concept = Car as car
  overrides <none>
  do not apply on the fly false

  do {
    if (car.name.isNotEmpty && Character.isLowerCase(car.name.charAt(0))) {
      info "Name must start with uppercase!" -> car ;
    }
  }
}
```

Inspector

jetbrains.mps.lang.typesystem.structure.InfoStatement

```
node feature to highlight(optional)
<no messageTarget>

intention linked with info message(optional)
ChangeCarNameToUpperCase (car = car)
apply immediately:
false

foreign message source(optional)
<no foreignMessageSource>
```



Naslednjič: projekcijski urejevalnik

- Projekcijsko urejanje z različnimi projekcijami

```
System.out.println(String.valueOf(( $\sum$   $\begin{bmatrix} 1 & k & 0 \\ 0 & 1.0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$  )));  
System.out.println( $\exp(a + i * b) = \exp(a) * (\cos(b) + i * \sin(b))$ );
```

```
matrix<Double> s =  $\begin{bmatrix} 3.0 & \sin(1) & 1 \\ 2 & 1 & 3 + \frac{1.0}{2} \\ 0 & 7 - \frac{1.0}{2} + 1 & \exp(1) \\ 4 & 0 & 0 \end{bmatrix}$ ;
```

```
if (discount < 400 EUR || discount > 500 EUR) {  
    discount = 300 EUR;  
}
```

```
System.out.println("Your name: " + createPerson());  
System.out.println("Your discount: " + discount);
```

```
}
```

```
public Money create(map<string, Object> person) {  
    return Money.Default(0 EUR);  
}
```

	isLevel_1(person)	isLevel_2(person)
isChild(person)	500 EUR	1000 EUR
isAdult(person)	50 EUR + this.seasonalBonus()	100 EUR + this.seasonalBonus()
isRetired(person)	200 EUR	250 EUR + (person["name"] == "Susan" ? this.seasonalBonus() : 0 EUR)

```
private Money seasonalBonus() {  
    return 100 EUR;  
}
```

Naloge na vajah

- Naloga 6.1. (neobvezna)
 - V jezik iz naloge 5.1 vključite vsaj:
 - 5 slogov za različne koncepte
 - 3 lastne omejitve in
 - 3 lastne omejitve s sistemom tipov.

- Naloga 6.2. (neobvezna)
 - Za jezik iz naloge 5.1 razširite urejevalnik s funkcionalnostmi:
 - pomočnik za trenutni kontekst,
 - namere,
 - hitri popravki, ter
 - 2 funkcionalnost po lastni izbiri.

VPRAŠANJA

