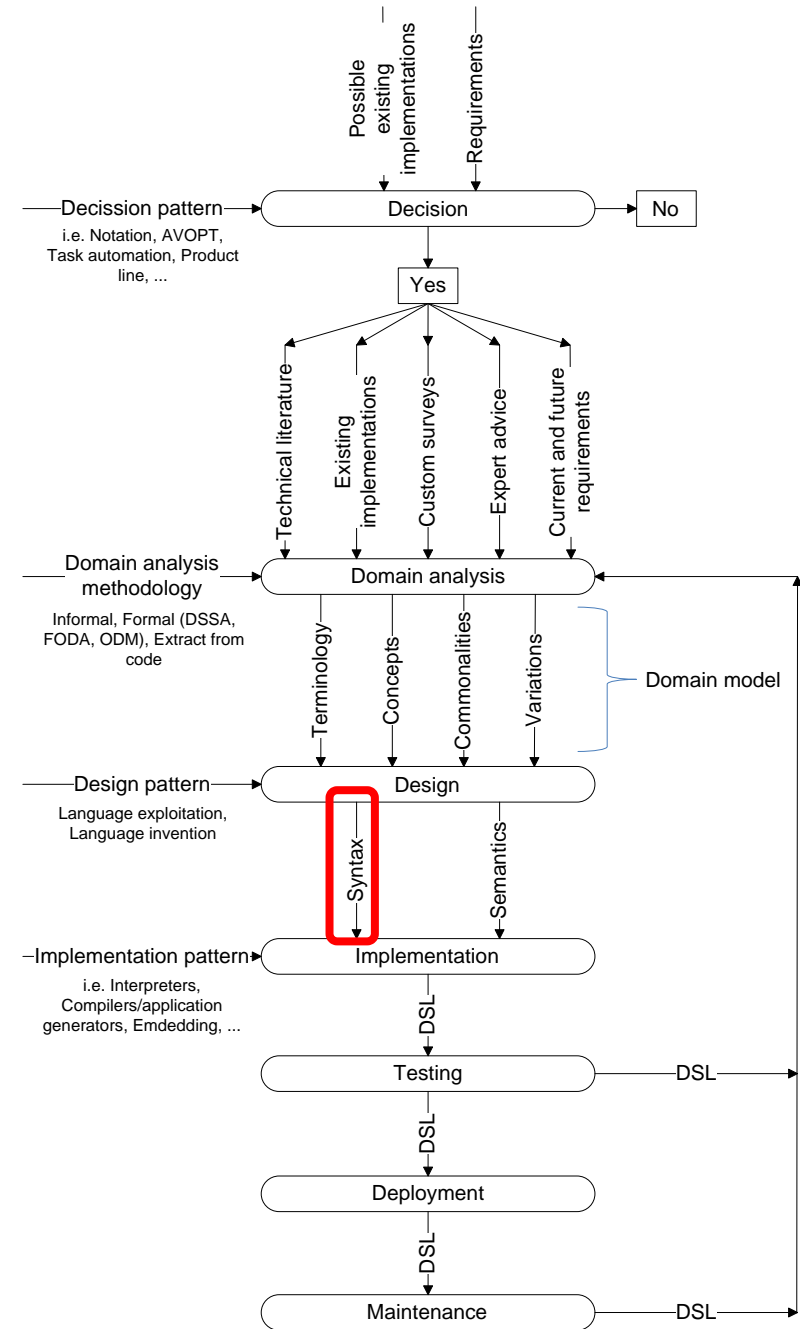


# DOMENSKO SPECIFIČNI MODELIRNI JEZIKI 2024/2025

5. predavanje

# Življenjski cikel



# Tekstovni jeziki (ponovitev)

- **Razširjena Backus-Naur-Form (EBNF)**
  - **Niklas Wirth** je formalizem predstavil, da je določil sintakso programskega jezika Pascal
  - Uporabimo ga za določitev kontekstno-prostih gramatik
  - Vključen v ISO standard
- Temeljna predpostavka: Koda je **zaporedje terminalnih simbolov**.
- EBNF določa vsa **veljavna zaporedja** terminalov s **produkcijskimi pravili** → gramatika
- Produkcije so sestavljene iz **leve** (neterminal) in **desna stran** (veljavna zaporedje terminalov in neterminalov)

# Tekstovni jeziki (ponovitev)

- Druge oblike kako podamo domensko specifičen jezik
- JetBrains MPS
  - Abstraktno sintaksno drevo
    - Koncepti
    - Projekcija

```
<default> editor for concept ReakylDSL
node cell layout:
[ / ]
[ - reakyll { name } { - ]
[ - ---> name : { dsl_name } - ]
[ - ---> title : { title } - ]
[ - ---> description : { description } - ]
[ - ---> port : { port } - ]
[ - ---> components : ( / % listOfComponents % / ) - ]
[ - ---> api : % api % - ]
[ - ---> render : ( / % renders % / ) - ]
[ - } - ]
[ / ]

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

```
concept ReakylDSL extends BaseConcept
    implements INamedConcept

instance can be root: true
alias: <no alias>
short description: <no short description>

properties:
    dsl_name      : string
    title         : string
    description   : string
    port          : integer

children:
    listOfComponents : RComponent[0..n]
    api               : RApi[0..1]
    renders           : ComponentRender[0..n]

references:
    << ... >>
```

# Sintaksa (generična vs. specifična)

- Generična sintaksa

- **Tekstovni jeziki**

- Abstraktna sintaksa
    - Privzeta sintaksa v MPS, **XML** (serijski modeli notacije), **JSON**

- **Modelirni jeziki**

- Metamodel
    - Metamodel v **XMI**

- Slabost: **brez sintaktičnih sladkorjev**

- Specifična sintaksa

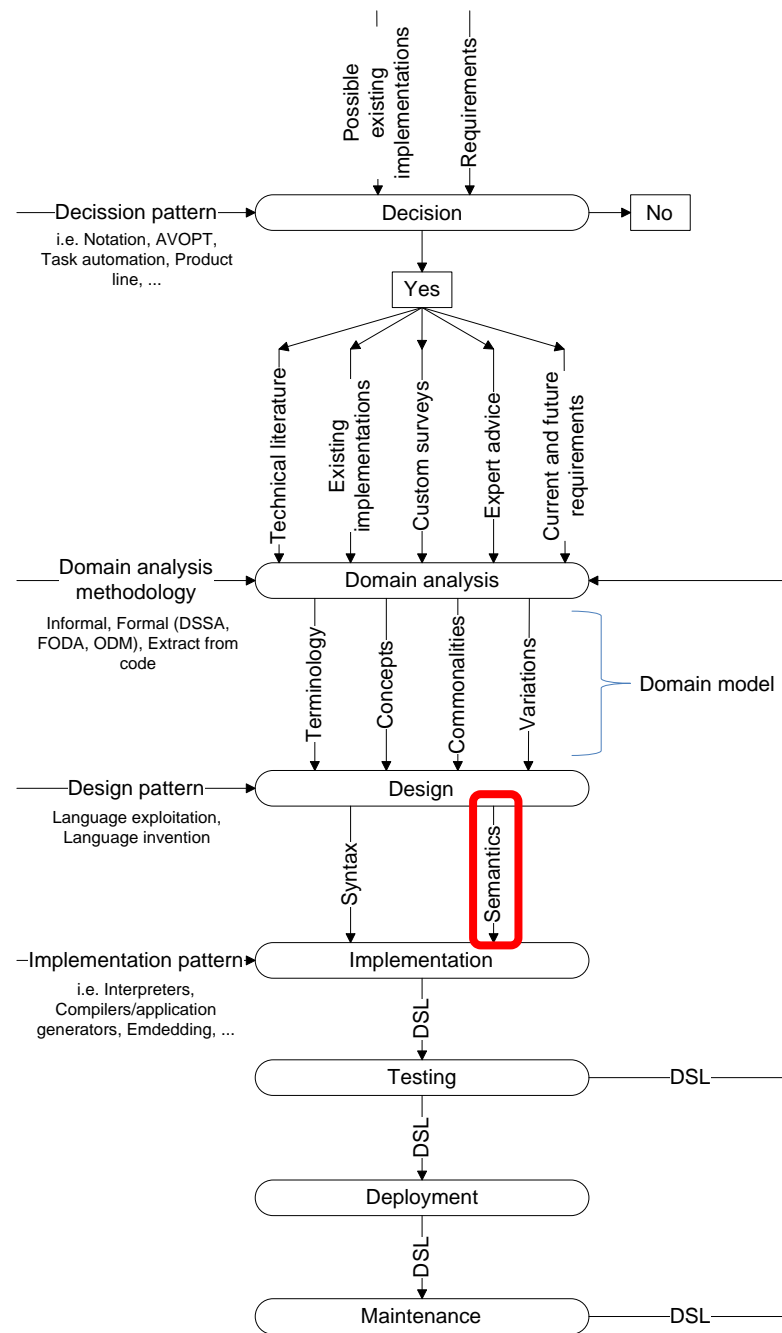
- **Tekstovni jeziki**

- 1. korak: Sintaksa je določena z gramatiko oz. AST (abstraktna sintaksa in konkretna sintaksa), npr. MPS (koncepti)
    - 2. korak: razširimo s specifičnimi konstrukti (konkretna sintaksa), npr. MPS (editor, projekcija)

- **Modelirni jeziki**

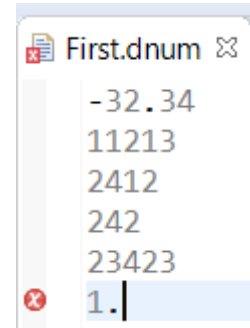
- 1. korak: metamodel (abstraktna sintaksa)
    - 2. korak: določite vizualno sintakso (konkretna sintaksa)

# Življenjski cikel



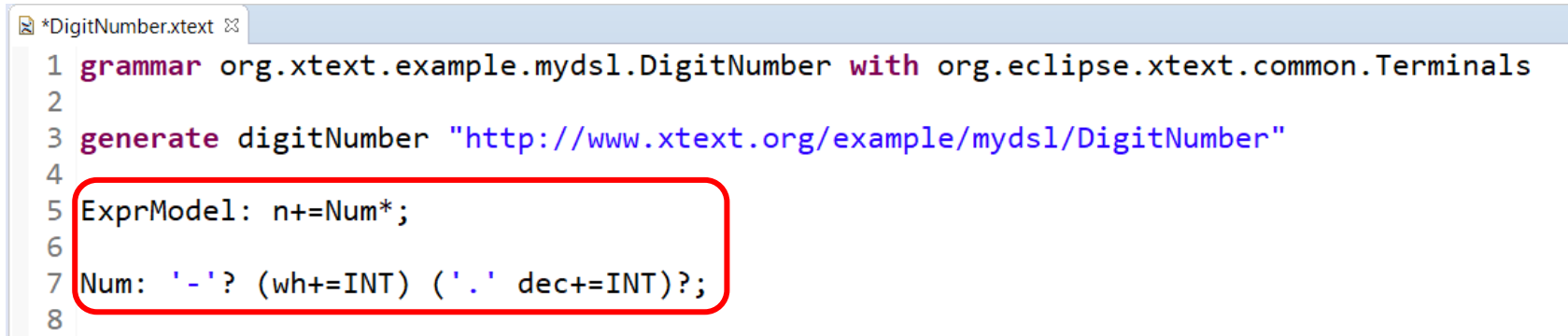
# Sintaksa in semantika jezika

- Primer digitalnih številok



First.dnum

-32.34  
11213  
2412  
242  
23423  
1.



```
*DigitNumber.xtext
1 grammar org.xtext.example.mydsl.DigitNumber with org.eclipse.xtext.common.Terminals
2
3 generate digitNumber "http://www.xtext.org/example/mydsl/DigitNumber"
4
5 ExprModel: n+=Num*;
6
7 Num: '-'? (wh+=INT) ('.' dec+=INT)?;
8
```

- Kaj pa pomen?
  - Polje, CSV, bazo podatkov,...

# Kaj naredi program

- Kaj je **rezultat**?
- **Kako** dobimo rezultat?
  - Formalni pristop
    - Več o tem v naslednjem predavanju
  - Orodje za zapis semantike
- Potrebujemo **semantiko** jezika



# Kaj je pomen programa

- Primer aritmetičnih izrazov:  $3 + 4$ 
  - Pomen:
    - 7, 7.00
    - izris/izris računa (na različne načine)
- Kaj pa spodnji program?

test.mydsl

```
default Skip 1 1 1 1
default Size 2 5

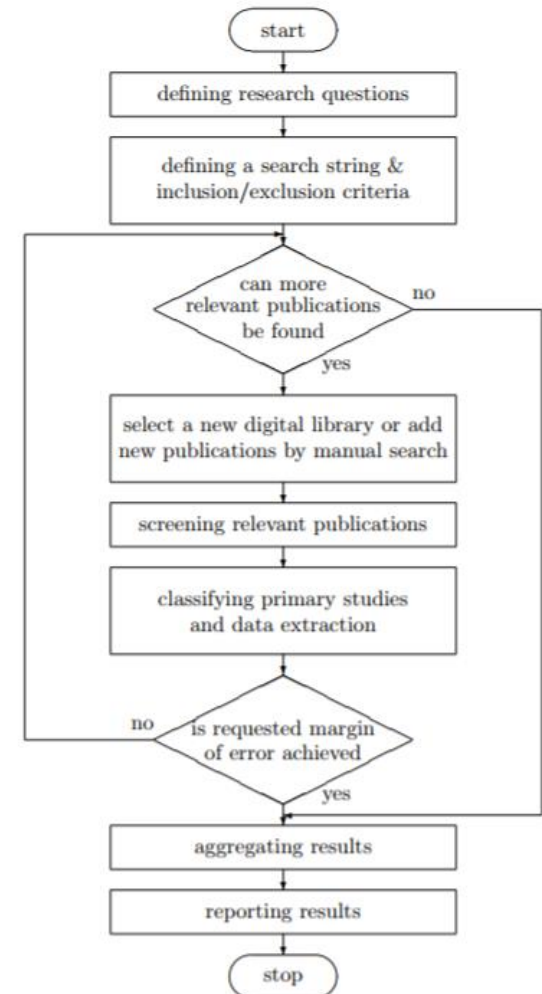
oval "start"
box "defining research questions"
box 4 5 "defining a search string & inclusion/exclusion criteria"

choice . no yes . "can more relevant publications be found"

box 4 5 "select a new digital library or add new publications by manual search"
box "screening relevant publications"
box 4 5 "classifying primary studies and data extraction"

choice . . yes no "is requested margin of error achieved"

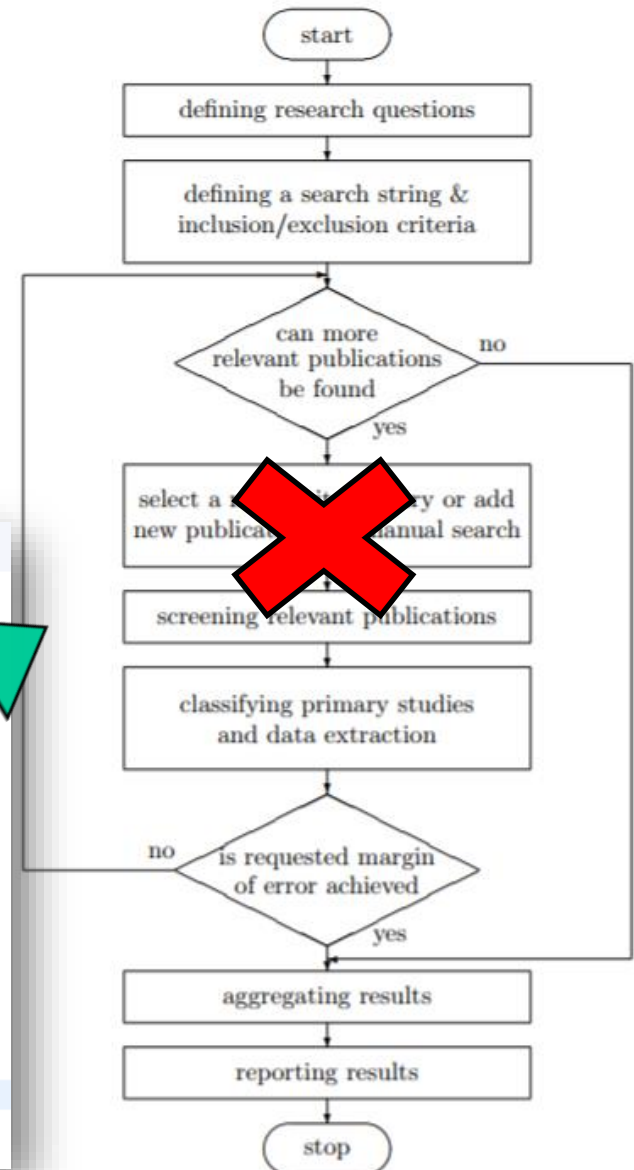
box "aggregating results"
box "reporting results "
oval "stop"
```



# Kaj naredimo v semantiki jezika

- Pomenov istega jezika je lahko **več**
  - Primer jezika EasyWorkFlow
    - Latex (slika desno)
    - SVG (glej naslednjo stran)
- A je desno res rezultat naše semantike?

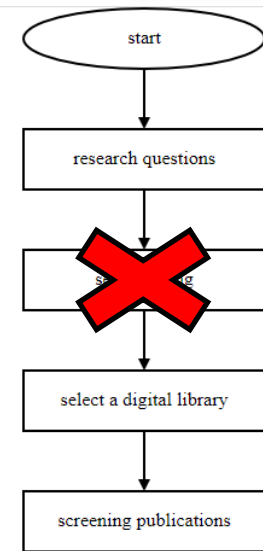
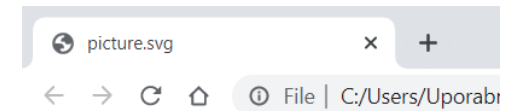
```
test.mydsl ✕  
  
default Skip 1 1 1 1  
default Size 2 5  
  
oval "start"  
box "defining research questions"  
box 4 5 "defining a search string & inclusion/exclusion criteria"  
  
choice . no yes . "can more relevant publications be found"  
  
box 4 5 "select a new digital library or add new publications by manual search"  
box "screening relevant publications"  
box 4 5 "classifying primary studies and data extraction"  
  
choice . . yes no "is requested margin of error achieved"  
  
box "aggregating results"  
box "reporting results "  
oval "stop"
```





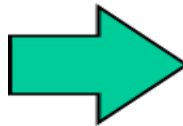
# Kaj naredimo v semantiki jezika

- Ločimo:
  - generirano kodo in
  - zagon generirane kode
- Tokrat primer v SVG
  - Translacija
  - Generiranje kode



program1.svg1

```
oval "start"
box "research questions"
box "search string"
box "select a digital library"
box "screening publications"
box "classifying primary studies"
box "aggregating results"
box "reporting results"
oval "end"
```

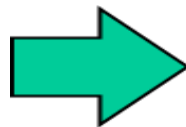


```
program1.svg1 C:\Users\Uporabnik\runtime-Eclipse\Text\SvgShapes1\src-gen\picture.svg picture.svg
<svg xmlns="http://www.w3.org/2000/svg" width="400.0" height="1000.0">
<defs>
<marker id="arrow" viewBox="0 -5 10 10" refX="5" refY="0" markerWidth="5" markerHeight="5" orient="auto">
<path d="M0,-5L10,0L0,5" class="arrowHead"></path>
</marker>
</defs>
<ellipse cx="200.0" cy="26" rx="100" ry="25" stroke="black" stroke-width="2px" fill="none"/>
<text x="200.0" y="26" width="200" alignment-baseline="middle" text-anchor="middle">start</text>
<line x1="200.0" y1="51" x2="200.0" y2="96" style="stroke:rgb(0,0,0);stroke-width:2" marker-end="url(#arrow)" />
<rect x="100.0" y="101" width="200" height="50" stroke="black" stroke-width="2px" fill="white"/>
<text x="200.0" y="126" width="200" alignment-baseline="middle" text-anchor="middle">research questions</text>
<line x1="200.0" y1="151" x2="200.0" y2="196" style="stroke:rgb(0,0,0);stroke-width:2" marker-end="url(#arrow)" />
<rect x="100.0" y="201" width="200" height="50" stroke="black" stroke-width="2px" fill="white"/>
<text x="200.0" y="226" width="200" alignment-baseline="middle" text-anchor="middle">search string</text>
<line x1="200.0" y1="251" x2="200.0" y2="296" style="stroke:rgb(0,0,0);stroke-width:2" marker-end="url(#arrow)" />
<rect x="100.0" y="301" width="200" height="50" stroke="black" stroke-width="2px" fill="white"/>
<text x="200.0" y="326" width="200" alignment-baseline="middle" text-anchor="middle">select a digital library</text>
<line x1="200.0" y1="351" x2="200.0" y2="396" style="stroke:rgb(0,0,0);stroke-width:2" marker-end="url(#arrow)" />
<rect x="100.0" y="401" width="200" height="50" stroke="black" stroke-width="2px" fill="white"/>
<text x="200.0" y="426" width="200" alignment-baseline="middle" text-anchor="middle">screening publications</text>
<line x1="200.0" y1="451" x2="200.0" y2="496" style="stroke:rgb(0,0,0);stroke-width:2" marker-end="url(#arrow)" />
<rect x="100.0" y="501" width="200" height="50" stroke="black" stroke-width="2px" fill="white"/>
<text x="200.0" y="526" width="200" alignment-baseline="middle" text-anchor="middle">classifying primary studie</text>
<line x1="200.0" y1="551" x2="200.0" y2="596" style="stroke:rgb(0,0,0);stroke-width:2" marker-end="url(#arrow)" />
<rect x="100.0" y="601" width="200" height="50" stroke="black" stroke-width="2px" fill="white"/>
<text x="200.0" y="626" width="200" alignment-baseline="middle" text-anchor="middle">aggregating results</text>
<line x1="200.0" y1="651" x2="200.0" y2="696" style="stroke:rgb(0,0,0);stroke-width:2" marker-end="url(#arrow)" />
<rect x="100.0" y="701" width="200" height="50" stroke="black" stroke-width="2px" fill="white"/>
<text x="200.0" y="726" width="200" alignment-baseline="middle" text-anchor="middle">reporting results</text>
<line x1="200.0" y1="751" x2="200.0" y2="796" style="stroke:rgb(0,0,0);stroke-width:2" marker-end="url(#arrow)" />
<ellipse cx="200.0" cy="826" rx="100" ry="25" stroke="black" stroke-width="2px" fill="none"/>
<text x="200.0" y="826" width="200" alignment-baseline="middle" text-anchor="middle">end</text>
</svg>
```



# Kaj bo naš pomen

```
reykill MyToDoApp {  
  name : NotesApplication  
  title : Notes application  
  description : This is a Notes Application  
  port : 4100  
  components : title : Home  
                url : /  
                menu : false  
                title : About  
                url : /about  
                menu : false  
                title : ToDo  
                url : /todo  
                menu : false  
  api : base : http://localhost:3000/  
        endpoints : name : items  
                    url : : itmes-todo  
  render : ( Home =>  
    # Hello to Notes Application  
    This is our main page  
    # Second title  
    After second title
```



```
function Home() {  
  return (  
    <div>  
      <h1>Hello to Notes Application</h1>  
      <p>This is our main page</p>  
      <h1>Second title</h1>  
      <p>After second title</p>  
      <p>Another paragraph</p>  
    </div>  
  )  
}
```

```
src > components > JS items-todo.js > ...  
1  import React, {} from 'react'  
2  import {Alert, Snackbar, Table, TableBody, TableCell, TableContainer, TableHead, TableRow} from "@mui/material";  
3  import Typography from "@mui/material/Typography";  
4  import Button from "@mui/material/Button";  
5  import axios from "axios";  
6  import Box from "@mui/material/Box";  
7  import {useState} from 'react'  
8  import {useSelector, useDispatch} from "react-redux";  
9  import {getItems} from "../redux/slices/itemsSlice";  
10  
11  function ItemsTodo() {  
12    const items = useSelector((state) => state.items.value);  
13    const [item, setItem] = useState('');  
14    const dispatch = useDispatch();  
15  
16    > const sendUpdate = (e) => { ...  
39  }  
40  
41  > return (...  
74  )  
75  }  
76  
77  export default ItemsTodo  
78  
79  
80
```

# Generiranje kode

- **Compiler Engineering:** angl. component of the synthesis phase
  - Wikipedia: „Code generation is the process by which a compiler’s code generator **converts a syntactically-correct program into a series of instructions** that can be executed by a machine.“
  - program, ki generira strojno kodo
- **Software Engineering:** angl. program to generate source code
  - Code Generation in Action (Herrington 2003): „Code generation is the technique of using or writing programs **that write source code.**“
  - program, ki generira izvorno kodo

# Generiranje kode

- Tri osnovna vprašanja
  - **Koliko** kode se generira?
    - Generiramo **celotno ali delno** kodo?
  - **Kaj** generiramo?
    - Katere **vrste izvorne kode** želimo ustvariti?
  - **Kako** generiramo?
    - Katere **jezike in orodja** za razvijanje generatorjev kode lahko uporabimo?
      - GPL vs. DSL

# Koliko kode generiramo (1)

- **Celotna** generacija kode
- **Delna** generacija kode
  - **Manj generirane kode je bolje!**
  - Katere dele lahko **samodejno generiramo?**

# Kaj generiramo? (2)

- Generiramo lahko:
  - Programska koda
  - Dokumentacija
  - Testne primere
  - Serializacija (XMI, XML) → glej naslednjo stran
- Neposredna translacija v **strojno kodo** je možna, vendar je delo neprijetno, podvrženo napakam, kodo pa je težko optimizirati
  - Ponovna uporaba obstoječih generatorjev kode
  - Uporaba obstoječih funkcionalnosti (ogrodja, API-jev, komponente)





# Program v MPS

- Zapis programa v peskovniku MPS

```
<?xml version="1.0" encoding="UTF-8"?>
<model ref="r:7996ea4f-b871-43ef-86c3-504babc0020a(CarLanguage.sandbox)">
  <persistence version="9" />
  <languages>
    <use id="d8f58dc2-94c6-43a8-981e-119e761d5a0b" name="CarLanguage" version="0" />
  </languages>
  <imports />
  <registry>
    <language id="d8f58dc2-94c6-43a8-981e-119e761d5a0b" name="CarLanguage">
      <concept id="5446538344419919630" name="CarLanguage.structure.Car" flags="ng" index="x97Fa">
        <child id="5446538344420041767" name="wheels" index="x9DBz" />
        <child id="5446538344419962704" name="color" index="x9Xak" />
        <child id="5446538344420454766" name="body" index="xb4ME" />
      </concept>
      <concept id="5446538344420041772" name="CarLanguage.structure.Wheels" flags="ng" index="x9DBC">
        <property id="5446538344420041867" name="wheels" index="x9D_f" />
      </concept>
      <concept id="5446538344420454763" name="CarLanguage.structure.CarBody" flags="ng" index="xb4MJ">
        <property id="5446538344420454764" name="carbody" index="xb4MC" />
      </concept>
      <concept id="5446538344420454772" name="CarLanguage.structure.Color" flags="ng" index="xb4MK">
        <child id="5446538344420454773" name="bodyColor" index="xb4ML" />
      </concept>
      <concept id="5446538344420454775" name="CarLanguage.structure.BodyColor" flags="ng" index="xb4MN">
        <property id="5446538344420454776" name="bodyColor" index="xb4MW" />
      </concept>
    </language>
    <language id="ceab5195-25ea-4f22-9b92-103b95ca8c0c" name="jetbrains.mps.lang.core">
      <concept id="1169194658468" name="jetbrains.mps.lang.core.structure.INamedConcept" flags="ng" index="TrEIO">
        <property id="1169194664001" name="name" index="TrG5h" />
      </concept>
    </language>
  </registry>
  <node concept="x97Fa" id="1oDSDtuEaUE">
    <property role="TrG5h" value="MyCar" />
    <node concept="xb4MJ" id="1oDSDtuEaUF" role="xb4ME">
      <property role="xb4MC" value="4IlZshEGvFJ/Limo" />
    </node>
    <node concept="xb4MK" id="1oDSDtuEaUG" role="x9Xak">
      <node concept="xb4MN" id="1oDSDtuEaUH" role="xb4ML">
        <property role="xb4MW" value="4IlZshEGyWM/red" />
      </node>
    </node>
    <node concept="x9DBC" id="1oDSDtuEaUI" role="x9DBz">
      <property role="x9D_f" value="4IlZshEGQgI/medium" />
    </node>
  </node>
</model>
```

# Primer generiranja kode

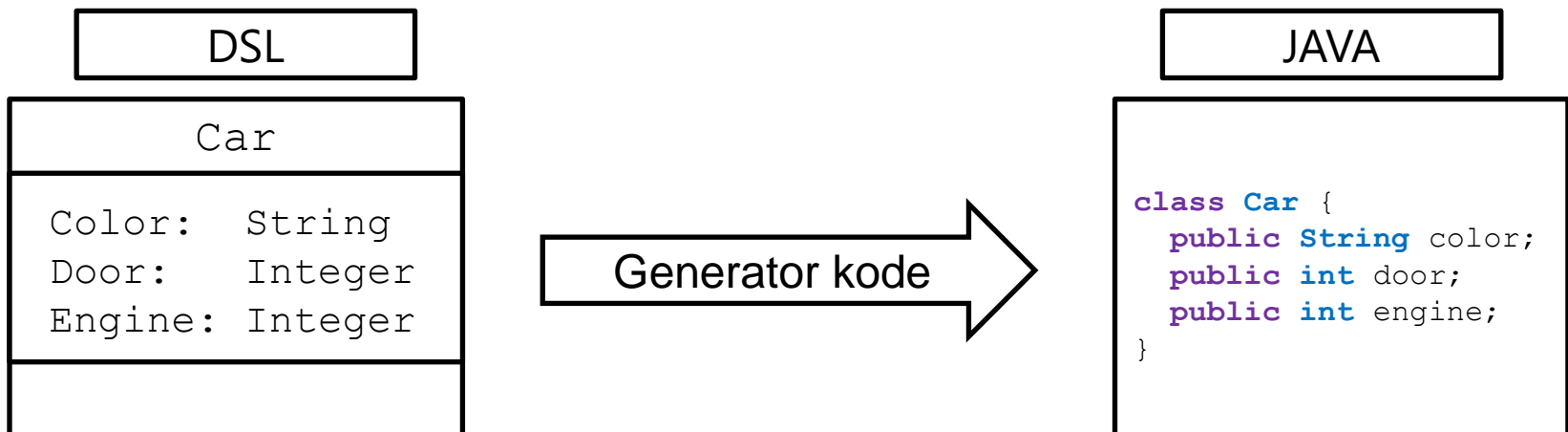
- Primer: **razvoj generatorja kode za spletne aplikacije (CLI)**
- Kakšne so želje in zahteve za generiranje kode?
  - **Kaj:** vsebina, povezave, predstavitev (domena)
  - **Programski jeziki:** JavaScript, Java, C#, Ruby, PHP, ...
  - **Arhitekture:** MVC, ...
  - **Ogrodja:** Express, Spring, Struts, Hibernate, Ruby on Rails, ASP, ...
  - **Produkti:** Node.js, MySQL, Tomcat, WebLogic, ...
- Katere kombinacije so primerne?
  - **Izkušnje** iz preteklih projektov generatorjev kode
  - **Referenčne (želene)** arhitekture, ogrodja, itd.

# Zakaj generiranje kode?

- Generiranje kode omogoča
  - Ločevanje **modeliranja znotraj domene** in **tehnične kode**
  - Povečuje lažje **vzdrževanje, razširljivosti, prenosljivosti** na:
    - novo strojno opremo, operacijske sisteme in platforme
  - **Hitro prototipiranje**
  - Hitre povratne informacije zaradi **testnih zagonov**
  - Generiranje kode omogoča **združevanje** generirane (**podvojene**) kode

# Zakaj ne UML?

- Pogosto ni možna "resnična" simulacija modela
  - UML okolja v glavnem **ne omogočajo simulacij**
    - Zagotavljajo pa **preoblikovanje** v C, C#, Java, ...
  - UML virtualni stroji
    - Interpreter – generiranje **prazne kode** za določene platforme
- **Semantika** modelirnih jezikov, zlasti DSML, je pogosto **definirana z generiranjem kode**

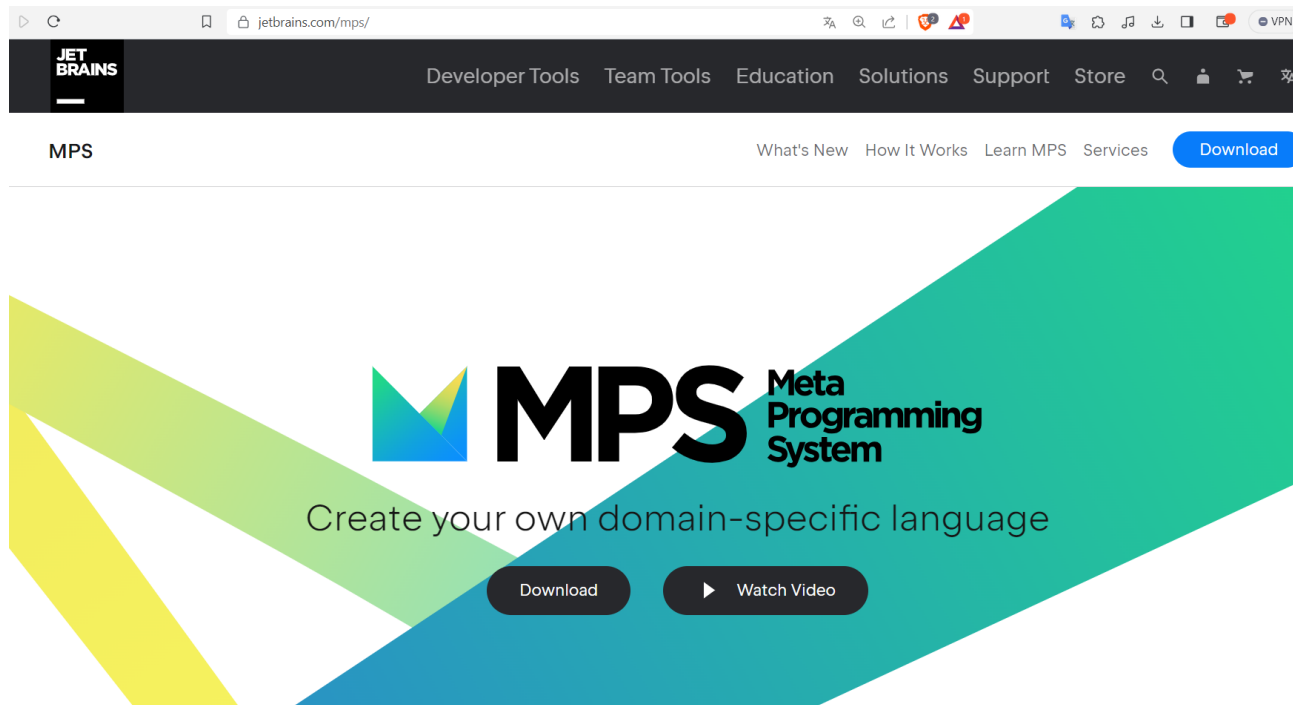


# Tipi generiranja kode

- Generiranje kode na podlagi:
  - M2C / M2T (model to code - model to text)
  - M2M (model to model)

# Language workbench

- **Jezikovno delovno okolje** (angl. language workbench) je orodje, ki omogoča razvoj programske opreme z jeziki.
- Vključuje **orodja za definicijo**, ponovno uporabo in sestavo DSL **skupaj z integriranim razvojnim okoljem (IDE)**.
- JetBrains MPS, Xtext, Racket, ...



# Model-to-text (M2T, M2C)

- Generiranje kode v Language Workbench s pomočjo transformacijskih jezikov:
  - JET, JET2
  - **Xpand**
  - **Xtend**
  - **TextGen**
  - MOFScript
  - Acceleo
  - XSLT
  - ...

# XPAND

- **Šabloni** (angl. templates) so dobro uveljavljena tehnika v programskem inženiringu
  - Področja aplikacij: obdelava besedila, spletno inženirstvo, ...
  - Primer:

## *E-Mail Text*

Dear **Homer Simpson**,  
Congratulations! You have won ...

## *Template Text*

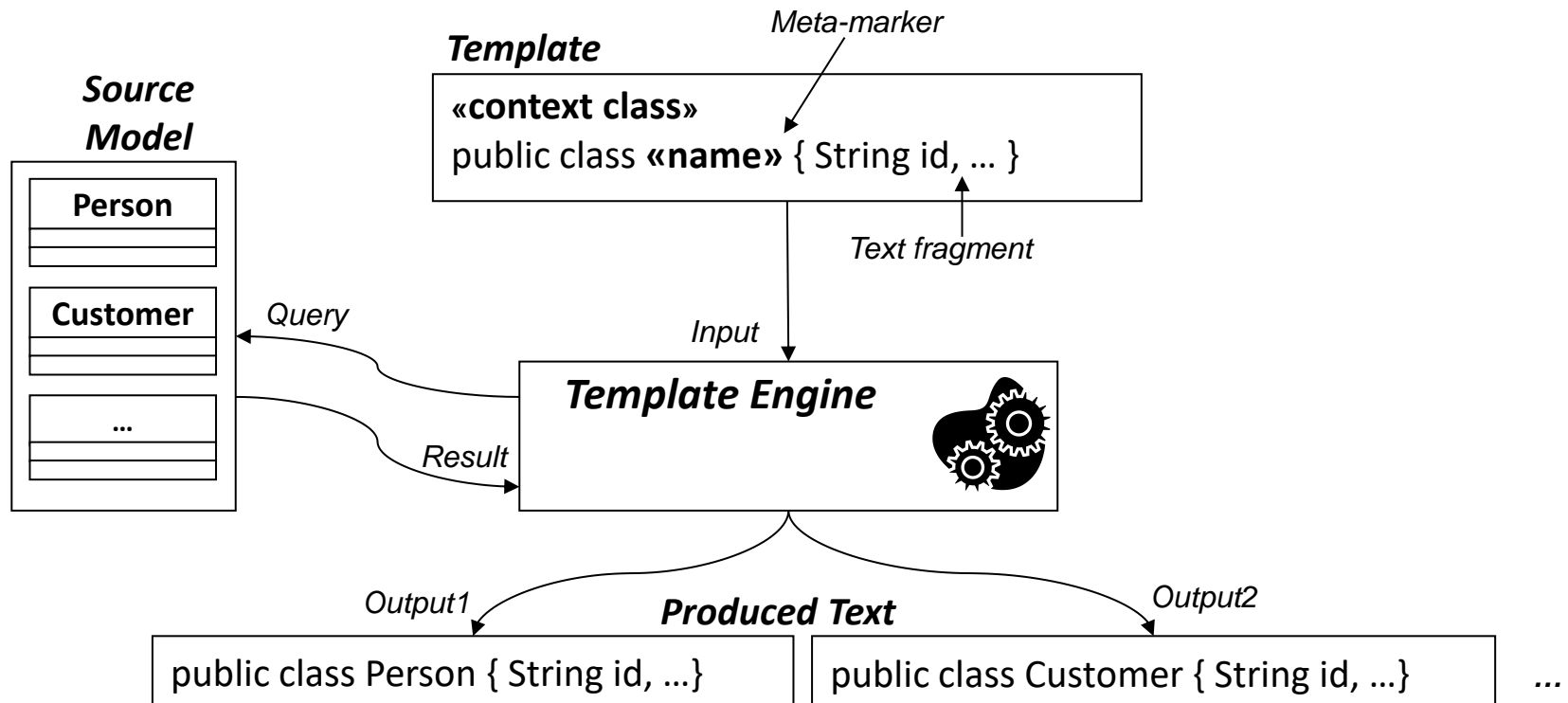
Dear «**firstName**» «**lastName**»,  
Congratulations! You have won ...

- Komponente pristopa, ki temelji na šablonah:
  - **Šablone**
    - Tekst in vgrajeni meta-markerji
  - **Meta-markerji** poizvedujejo po podatkih
    - Treba jih je razlagati in ocenjevati v nasprotju z besedilnimi deli
    - Deklarativna poizvedba modela: jeziki poizvedb (OCL, XPath, SQL)
    - Imperativna poizvedba po modelu: programski jeziki (Java, C #)
  - **Pogon šablon**
    - Zamenja meta-markerje s podatki v času izvajanja in naredi izhod (datoteke)



# XPAND

- Generiranje kode s pomočjo šablon



# XPAND

- Ločena statična / dinamična koda
  - Šablone ločijo statično kodo (fiksno kodo) od dinamične, ki jo opisujejo meta-markerji
- Eksplicitna izhodna struktura
  - Primarna struktura šablone je izhodna struktura
  - Logika je vgrajena v to strukturo
- Deklarativni **jezik poizvedb**
  - OCL se uporablja za poizvedovanje vhodnih modelov
- Večkratna uporaba
  - Podpora za **branje iz modela**, serializacija kode v datoteke, ...

# XPAND

- Primer transformacijskega jezika - Xpand

```
«IMPORT MK1»
```

```
«DEFINE main FOR RootCalc»
```

```
  «FILE "Kalkulator.cpp"»
```

```
    #include <cstdlib>
```

```
    #include <ctime>
```

```
    #include <iostream>
```

```
    using namespace std;
```

```
    int main()
```

```
    {
```

```
        cout << «EXPAND InstanceNumber FOREACH toOperation » ;
```

```
        return 0;
```

```
    }
```

```
  «ENDFILE»
```

```
«ENDDEFINE»
```

```
«DEFINE InstanceNumber FOR Operation»
```

```
  «this.leftN.value» «this.sign» «this.rightN.value»
```

```
«ENDDEFINE»
```

# Xtend

- Xtend je splošno namenski programski jezik
- Omogoča **generiranje kode** na podlagi gramatike podane v Xtext
- Sintaksa je fleksibilna in omogoča zapis notranjih (internal) DSL-jev
- Prevede se v **Java izvorno kodo** (angl. transpiler)
  - kompatibilen s standardnimi in ostalimi javanskimi knjižnicami
  - Kompatibilen z Java 8, Androidom, in Google Web Toolkit
- Hitra **učna krivulja**
- **Lahko uporabljamo** tudi **šablone** (kot pri Xpand)



# Xtend – primer

```
override void doGenerate(Resource resource, IFileSystemAccess2 fsa, IGeneratorContext context) {  
    var picture_width=400.0  
    var picture_height=1000.0  
    var current_x = picture_width / 2  
    var current_y = 1  
    var element_width=200  
    var element_height=50  
    var distance = 50  
    var arrow_head = 5  
    var result = "<svg xmlns='\"http://www.w3.org/2000/svg\"' width='\" + picture_width + '\" height='\" + picture_height + '\">\n"  
    result += "<defs>\n<marker id='\"arrow\"' viewBox='\"0 -5 10 10\"' refX='\"5\"' refY='\"0\"' markerWidth='\"5\"' markerHeight='\"5\"' orient='\"auto\"'>\n<path d='\"  
  
    var shapes_size = resource.allContents.toIterable.filter(Shape).size  
    var index = 1  
    for(shape: resource.allContents.toIterable.filter(Shape))  
    {  
        if (shape.sty=="oval"){  
            result += "<ellipse cx='\" + (current_x)  
                + "\" cy='\" + (current_y+element_height/2)  
                + "\" rx='\" + element_width/2  
                + "\" ry='\" + element_height/2  
                + "\" stroke='\"black\"' stroke-width='\"2px\"' fill='\"none\"'/>" + "\n"  
            result += "<text x='\" + (current_x)  
                + "\" y='\" + (current_y+element_height/2)  
                + "\" width='\" + element_width  
                + "\" alignment-baseline='\"middle\"' text-anchor='\"middle\">"  
                + shape.txt + "</text>" + "\n"  
        }  
        else if (shape.sty=="box"){  
            result += "<rect x='\" + (current_x-element_width/2)  
                + "\" y='\" + (current_y)  
                + "\" width='\" + element_width  
                + "\" height='\" + element_height  
                + "\" stroke='\"black\"' stroke-width='\"2px\"' fill='\"white\"'/>" + "\n"  
            result += "<text x='\" + (current_x)  
                + "\" y='\" + (current_y+element_height/2)  
                + "\" width='\" + element_width  
                + "\" alignment-baseline='\"middle\"' text-anchor='\"middle\">"  
                + shape.txt + "</text>" + "\n"  
        }  
        if(index != shapes_size)  
            result += "<line x1='\" + (current_x)  
                + "\" y1='\" + (current_y+element_height)  
                + "\" x2='\" + (current_x)  
                + "\" y2='\" + (current_y + element_height+ distance - arrow_head)  
                + "\" style='\"stroke:rgb(0,0,0);stroke-width:2\"' marker-end='\"url(#arrow)\"' />" + "\n"  
            current_y += element_height + distance  
            index++;  
    }  
  
    result += "</svg>"  
    fsa.generateFile('picture.svg', result)  
}
```

# JetBrains MPS generiranje kode

- V JetBrains MPS **dva možna pristopa** za generiranje kode:
  - Generator (model-to-model, M2M)
    - Izhod postopka je **model**
  - TextGen (model-to-text, M2T)
    - Izhod postopka je golo **besedilo**
- Privzeti pristop v MPS je transformacija iz **modela v model**

# MPS Generator

- Del JetBrains MPS, ki definira **denotacijsko semantiko** za koncepte v jeziku.

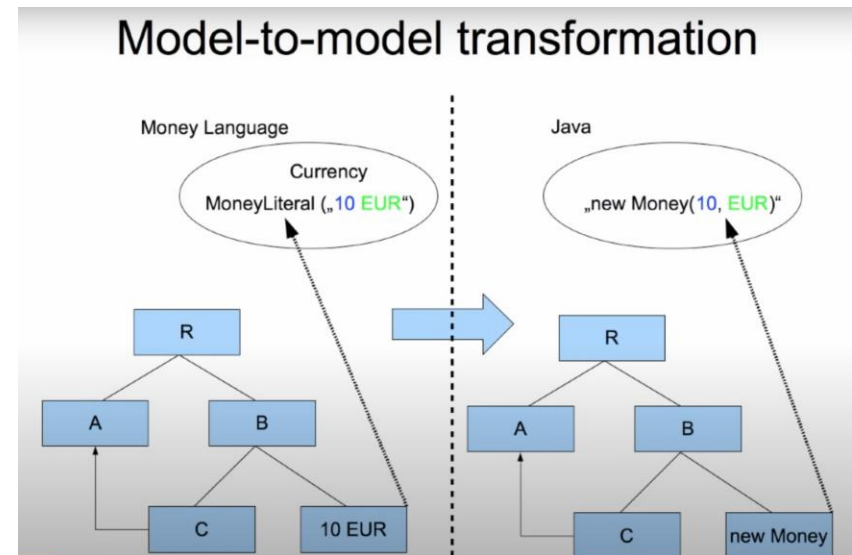
- En AST v drug AST

- Pristop model2model

- Generator MPS podaja translacijo za

- konstrukcij, kodiranih v vhodnem jeziku, v konstrukcije, kodiranih v izhodnem jeziku

- Translacija lahko vsebuje tudi **niz vmesnih modelov**



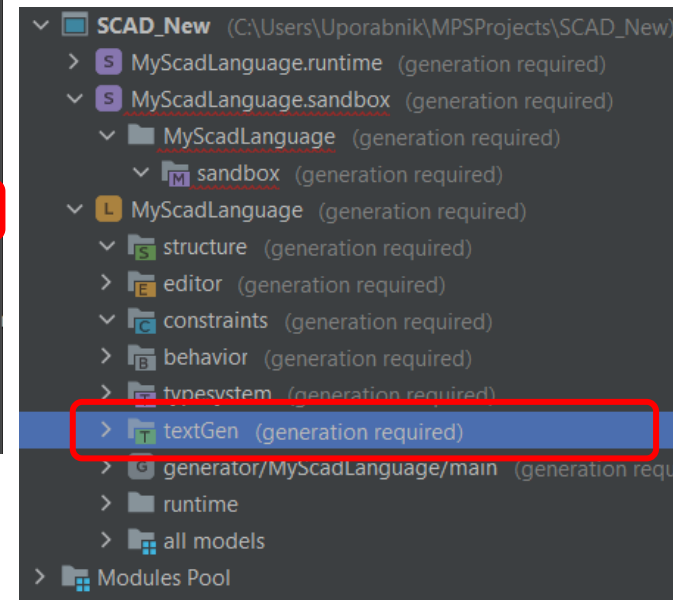
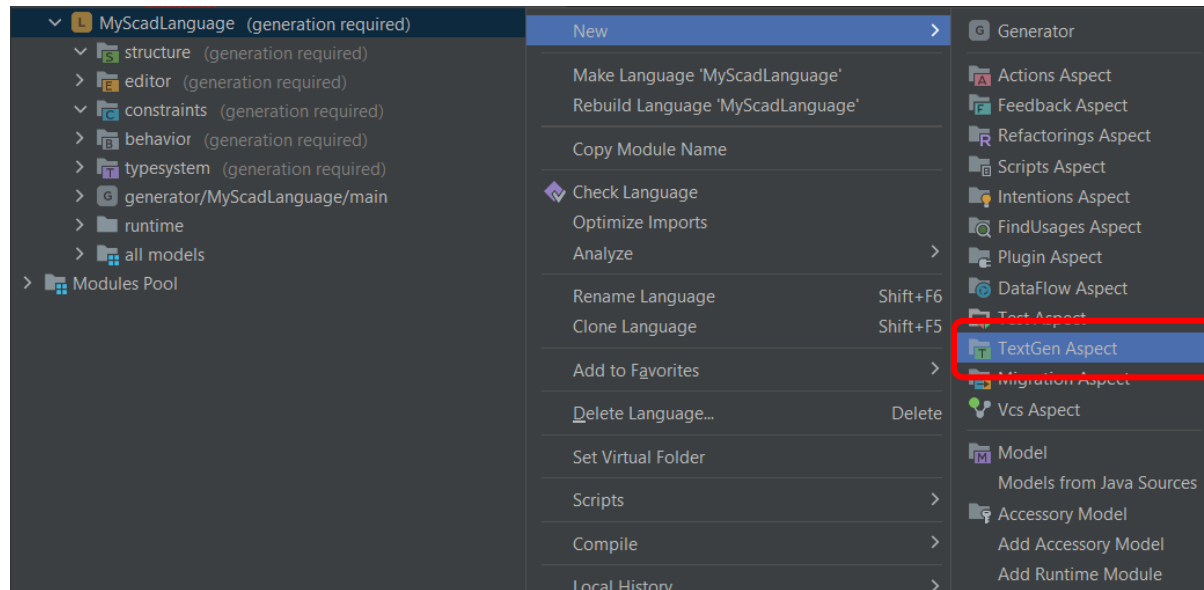
# TextGen

- Druga vrsta generatorja
  - Model2Text (M2T, M2C)
- Ciljna sredstva so ustvarjena z uporabo **transformacije modela v besedilo**, ki je program v „izhodnem“ jeziku.
- MPS podpora za definiranje transformacije modela v besedilo se imenuje **TextGen**
- Na voljo kot **ločen generator** v urejevalniku jezika.



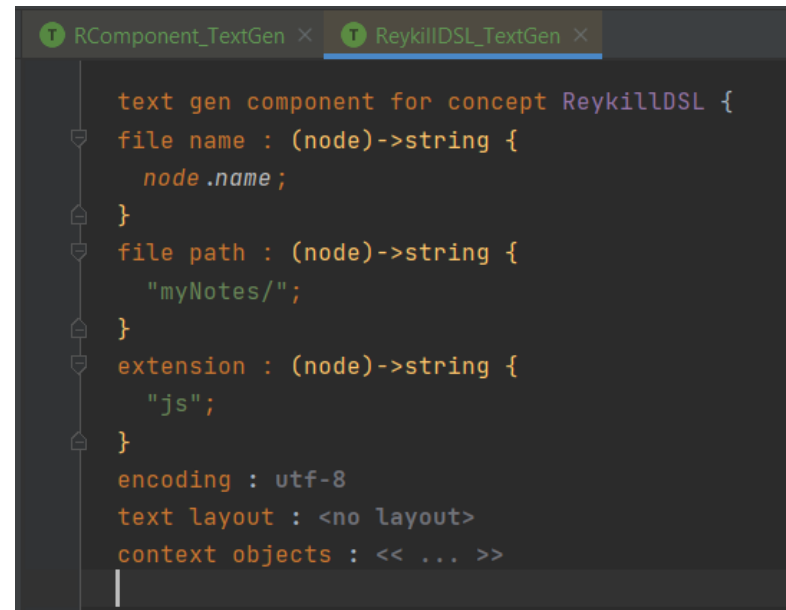
# TextGen aspekt

- Potrebno vključiti podporo TextGen v projekt



# TextGen

- TextGen ustvari datoteke na naslednjem mestu:
  - *< izhodna pot generatorja >\*  
*< pot datoteke >\*  
*< ime datoteke > . <končnica>*
- **Izhodna pot generatorja** - je določena v jeziku, ki je lastnik vhodnega modela.
- **Pot datoteke** – je segment poti, ustvarjen z zamenjavo.



```
text gen component for concept ReykillDSL {  
  file name : (node)->string {  
    node.name;  
  }  
  file path : (node)->string {  
    "myNotes/";  
  }  
  extension : (node)->string {  
    "js";  
  }  
  encoding : utf-8  
  text layout : <no layout>  
  context objects : << ... >>  
}
```

# TextGen

- Vsebuje konstrukte za generiranje besedila:
  - **Operacije**
  - **Pretvorbo** vozlišč iz AST v besedilne vrednosti
  - Podajanje razumne **postavitve** izhodnega besedila z zamiki

# TextGen - operacije

- **append**
  - Najpomembnejša operacija
  - Izhodu dodamo besedilo
- **error**
  - Besedilo napake

# TextGen - parametri operacije append

- **{string value}**
  - za vstavljanje uporabite "
- **\n**
- **\$list{node.list}**
  - seznam
- **\$list{node.list with ,}**
  - z ločilom
- **\$ref{node.reference}**
  - npr. `$ref{node.reference<target>}`
- **\${node.child}**
  - Potomci vozlišča
- **\${attributed node}\$**
  - Na voljo v atributih vozliščih, delegira v atributno vozlišče
- Append lahko vsebuje **poljubno število** zgornjih elementov

# TextGen - zamiki

- **with ident { ... }**
  - Podamo bloku zamik
- **Indent buffer**
  - uporaba trenutnega zamika
- **Increase/decrease depth**
  - Povečamo zamik ne glede na trenutno postavitev bloka

# TextGen - zamiki

- TextGen vsebuje močno orodje za **pravilno zamikanje** generiranega besedila
  - V medpomnilnik se shrani **izhodno besedilo** za vsako vozlišče posebej
  - Spremenljivka, ki vodi trenutno globino zamika (**indentation buffer**) za vsak korenski element
  - Kljub temu je potrebno zamik vstaviti ročno
    - Ukaz: **indent buffer**
    - Zastavica: **with indent** (ukaza append)
  - Blok **with indent** poveča vrednost zamika



# Primer: React komponenta

```
text gen component for concept RComponent {
  (node)->void {
    append {import React, {} from 'react';} \n;
    append \n;
    append {function } ${node.name} {} \n;

    with indent {
      indent buffer;
      append {return () \n;

      with indent {
        indent buffer;
        append {<div>} \n;

        with indent {
          foreach render in node.model.roots(ReakylDSL).renders {
            if (node.name == render.cName.name) {
              append ${render};
            }
          }
        }
      }
    }
  }
}
```

```
function Home() {
  return (
    <div>
      <h1>Hello to Notes Application</h1>
      <p>This is our main page</p>
      <h1>Second title</h1>
      <p>After second title</p>
      <p>Another paragraph</p>
    </div>
  )
}
```



# TextGen - korenski koncepti

- Dve vrsti generiranja kode za korenske koncepte
  - Concept text gen
    - **Transformacija koncepta** v besedilo
    - Za korenske koncepte je mogoče definirati tudi **ciljno datoteko**
  - Language text gen declaration
    - Ponovno uporabne **operacije** in **podporne metode**
    - Te je mogoče priklicati iz drugih komponent

# TextGen - podkoncepti

- MPS ne ustvari datotek za korenski koncept avtomatično
  - Enako velja tudi za ostale koncepte (podkoncepte)
  - Ustvari se samo za natančna ujemanja

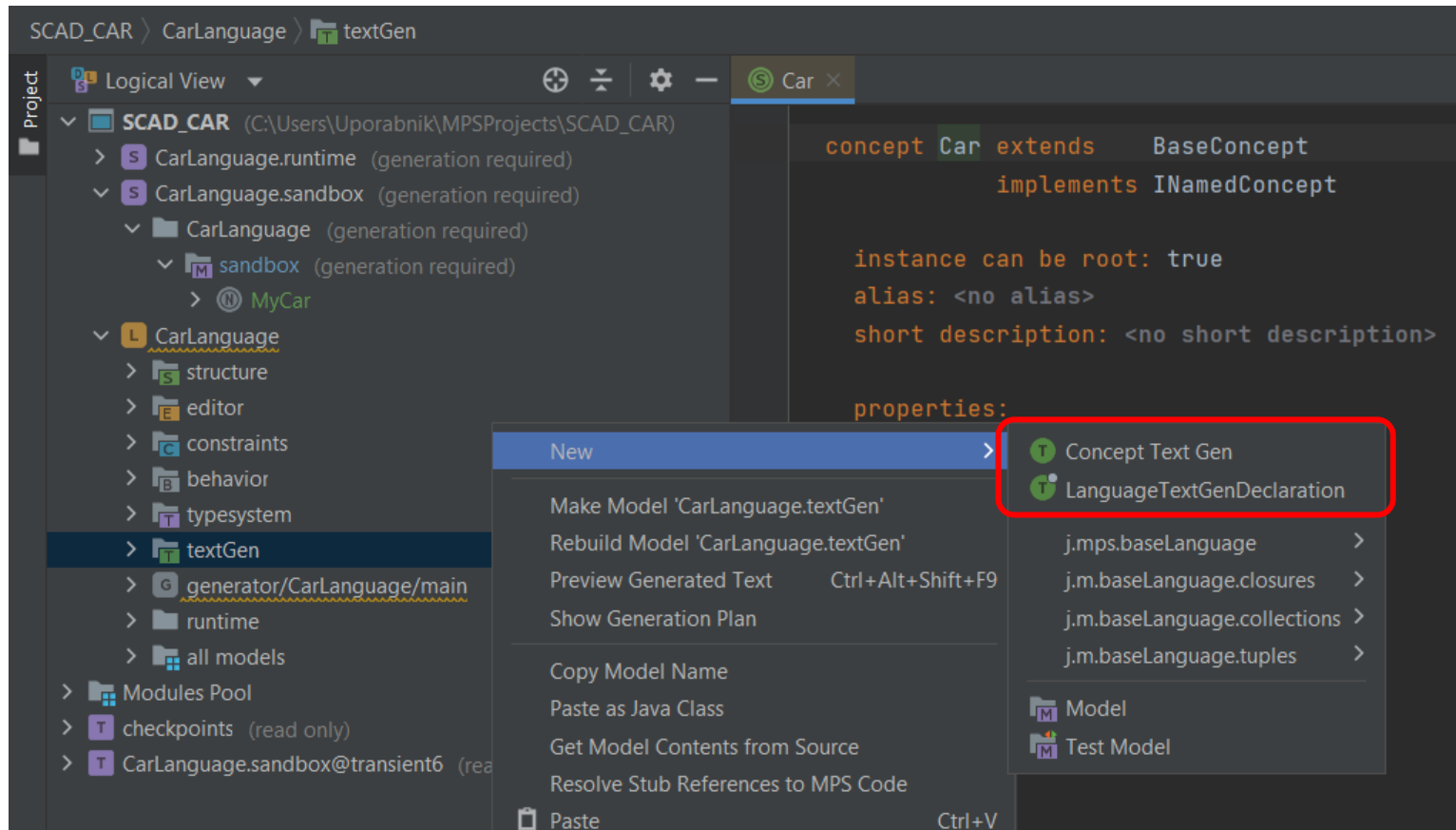
# TextGen - postavitve

- Obstaja mehanizem za nadzor postavitve (layout) izhodne datoteke: **text layout**
- **Samo** za korenske koncepte
- Definicija več **ločenih logičnih odsekov**
- V **ukazu append** **opcijsko** določimo v katero sekcijo dodamo besedilo
  - Uporabno za npr. include stavke, ipd.

```
ClassConcept_TextGen

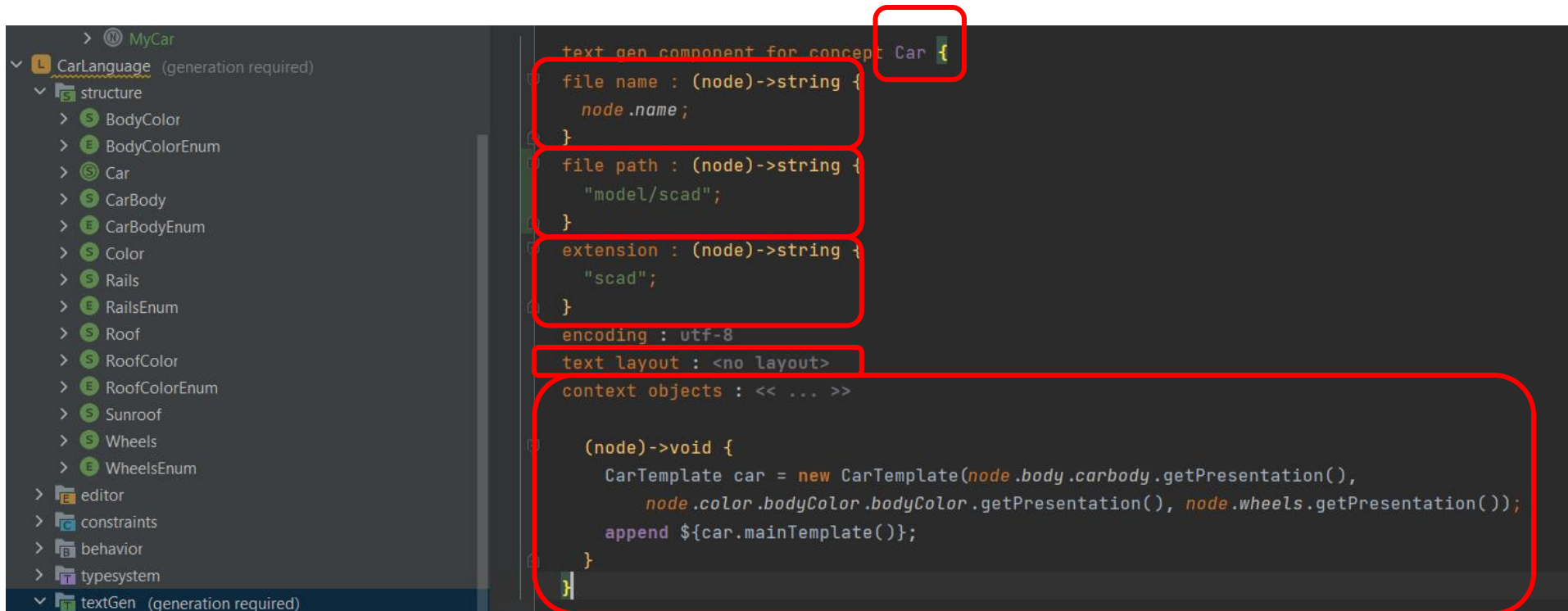
text gen component for concept ClassConcept {
  file name : <Node.name>
  extension : (node)->string {
    "java";
  }
  encoding : utf-8
  text layout : Initial text area BODY
    HEADER
    << ... >>
    IMPORTS
    << ... >>
    SEPARATOR
    << ... >>
    BODY
    << ... >>
```

# Primer CarLanguage



# Primer CarLanguage

- Konzept Car

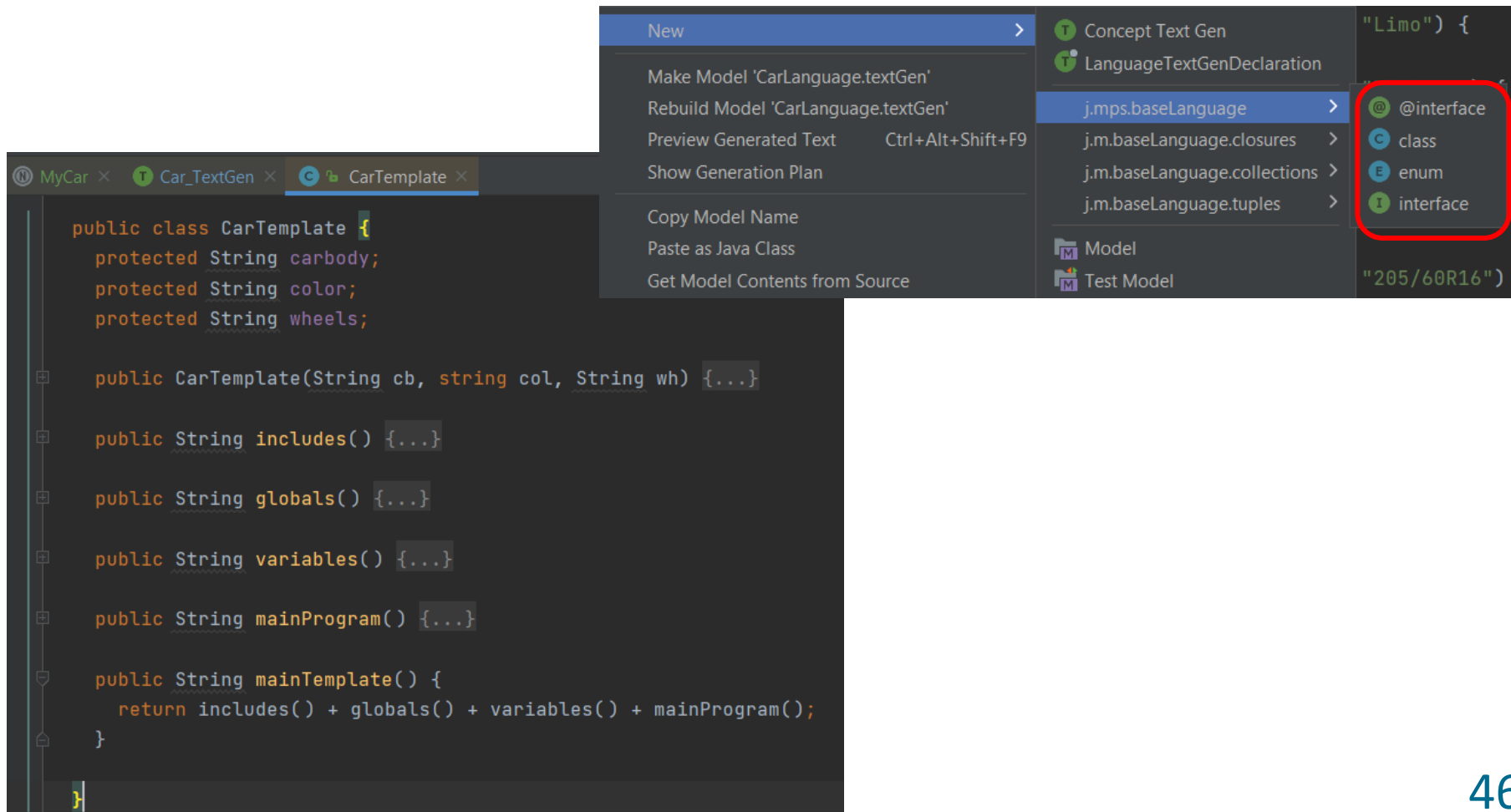


The screenshot displays the CarLanguage project in an IDE. The left sidebar shows the project structure, including a 'structure' folder with various components like BodyColor, Car, CarBody, Color, Rails, Roof, Sunroof, and Wheels. The main editor shows the code for a text generation component for the 'Car' concept. The code is as follows:

```
text gen component for concept Car {  
  file name : (node)->string {  
    node.name;  
  }  
  file path : (node)->string {  
    "model/scad";  
  }  
  extension : (node)->string {  
    "scad";  
  }  
  encoding : utf-8  
  text layout : <no layout>  
  context objects : << ... >>  
  
  (node)->void {  
    CarTemplate car = new CarTemplate(node.body.carbody.getPresentation(),  
                                     node.color.bodyColor.bodyColor.getPresentation(), node.wheels.getPresentation());  
    append ${car.mainTemplate()};  
  }  
}
```

# Primer CarLanguage

- Dodajanje razreda CarTemplate



```
public class CarTemplate {
    protected String carbody;
    protected String color;
    protected String wheels;

    public CarTemplate(String cb, String col, String wh) {...}

    public String includes() {...}

    public String globals() {...}

    public String variables() {...}

    public String mainProgram() {...}

    public String mainTemplate() {
        return includes() + globals() + variables() + mainProgram();
    }
}
```

New

- Make Model 'CarLanguage.textGen'
- Rebuild Model 'CarLanguage.textGen'
- Preview Generated Text Ctrl+Alt+Shift+F9
- Show Generation Plan
- Copy Model Name
- Paste as Java Class
- Get Model Contents from Source

Concept Text Gen

LanguageTextGenDeclaration

j.mps.baseLanguage

j.m.baseLanguage.closures

j.m.baseLanguage.collections

j.m.baseLanguage.tuples

Model

Test Model

"Limo") {

@interface

class

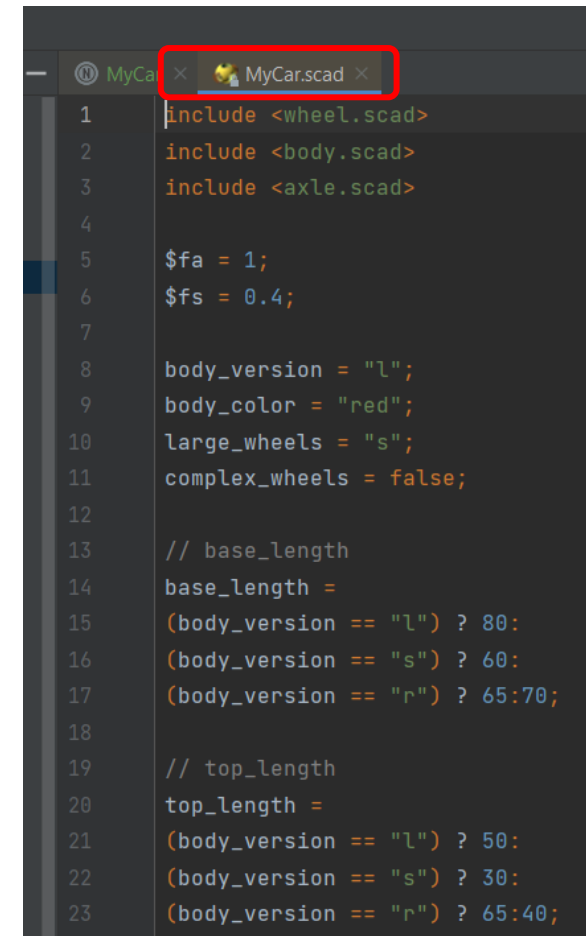
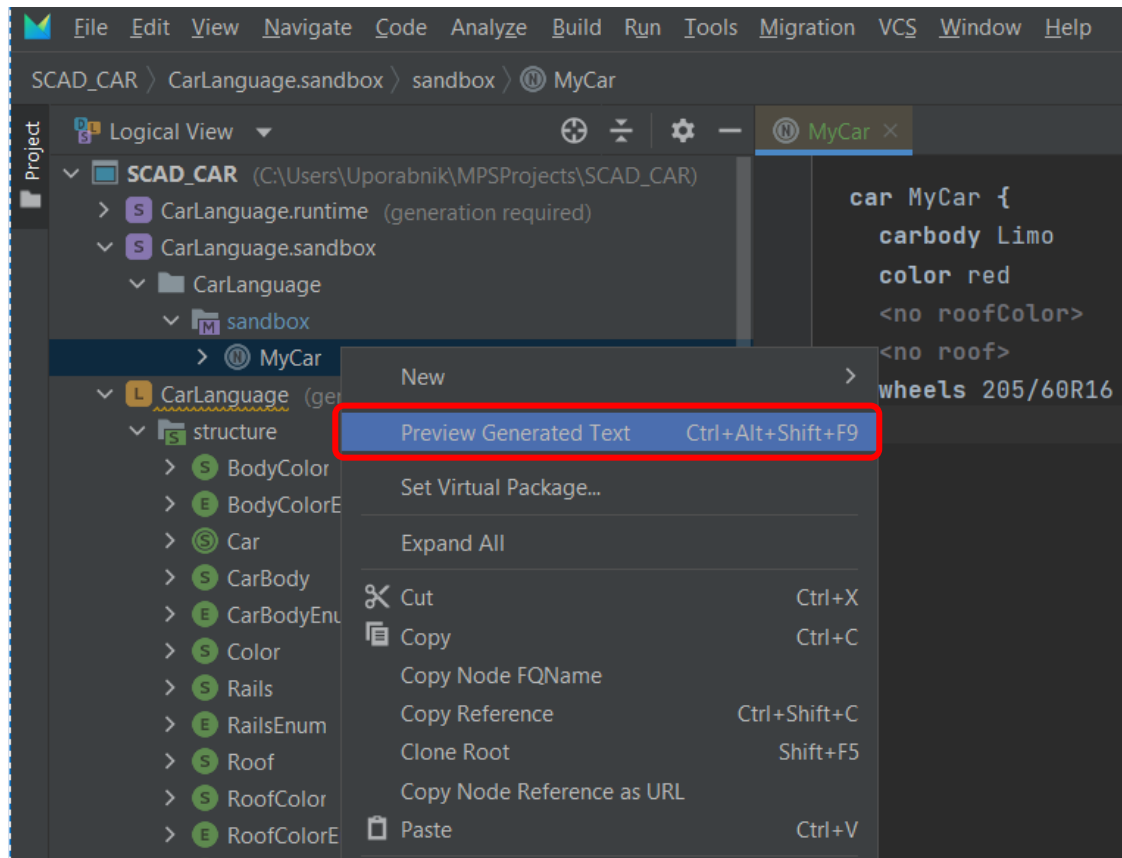
enum

interface

"205/60R16")

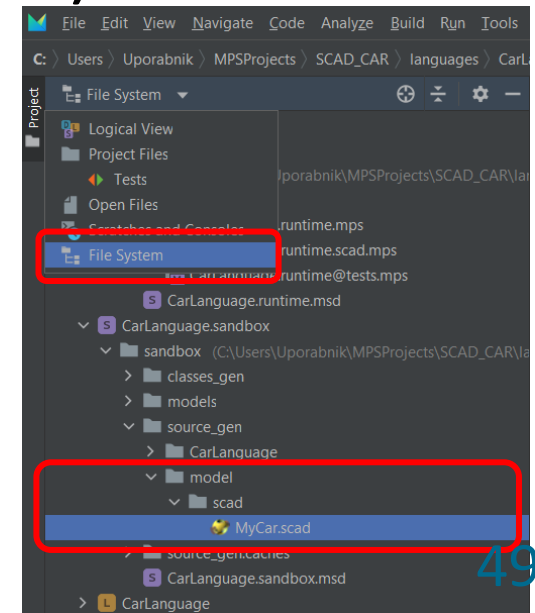
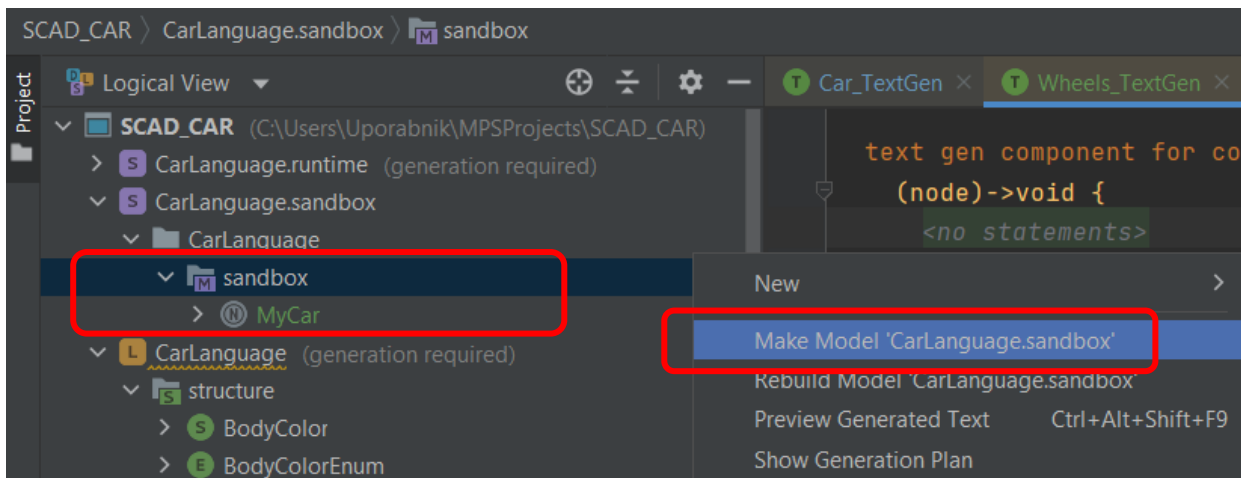
# Predogled kode

- Generiranje kode v peskovniku



# Generiranje kode

- Če želimo generirati kodo, moramo **zgraditi model** („make model“), ki smo ga definirali s TextGen
- Rešitev bo na voljo v mapi **source\_gen** (preklopite na pogled “File System“)





# Vaje

- Naloga 5.1. (obvezna)
  - Za jezik iz naloge 4.1, implementirajte translacijo v jezik Reakyll.
  - Za translacijo uporabite TextGen in orodje JetBrains MPS.
  - Za testiranje translacije uporabite programe iz naloge 1.1.
  - Za semantiko uporabite module iz naloge 1.2.
- Naloga je vredna **10 točk!**



# PRIPRAVE NA 1. KOLOKVIJ

<b>predmet</b>	<b>izvajalec</b>	<b>datum</b>	<b>prostor</b>	<b>ura</b>
Domensko specifični modelirni jeziki	Kosar	ponedeljek, 11. 11. 2024	E-105	14-16
Izbrani algoritmi	Strnad	torek, 19. 11. 2024	Beta+A-305	11-13
Spletne tehnologije	Lukač	četrtek, 21. 11. 2024	Alfa	13-15
Razvoj in upravljanje programskih sistemov	Kokol	četrtek, 14. 11. 2024	A-306	15-17
Računalniška multimedia	Žalik	ponedeljek, 18. 11. 2024	E-105	12-14
Računanje v oblaku	Zamuda	sreda, 20. 11. 2024	Alfa	15-17

# O kolokviju

- Na list papirja
- Esejski tip nalog
- Začnemo 14:00
- Dolžina pisanja: 45 min

# 1. kolokvij

- Izbira vprašanj iz vsakega predavanja
  - 1. predavanje – Uvod v DSMJ
  - 2. predavanje – Domenska analiza
  - 3. predavanje – Domensko načrtovanje
  - 4. predavanje – JetBrains MPS (struktura, urejevalnik)
  - 5. predavanje – Generiranje kode v MPS

# Priprave 1. kolokvij (1/10)

1) Naštejte 7 faz življenjskega cikla DSLjev ?

# Priprave 1. kolokvij (1/10)

1) Naštejte 7 faz življenjskega cikla DSLjev ?

- ✓ **Odločitev**
- ✓ **Domenska analiza**
- ✓ **Načrtovanje**
- ✓ **Implementacija**
- ✓ **Testiranje**
- ✓ **Namestitev**
- ✓ **Vzdrževanje**

# Priprave 1. kolokvij (2/10)

2) Kaj je namen domenske analize? (2)

# Priprave 1. kolokvij (2/10)

## 2) Kaj je namen domenske analize? (2)

**Obseg domene**

**Domenski model**





# Priprave 1. kolokvij (3/10)

Kaj je domena?

# Priprave 1. kolokvij (3/10)

Kaj je domena?

Domena je **znanje** iz specifičnega področja, ki ga opišemo z **nizom konceptov** in **terminologija**, ki ju **razumejo strokovnjaki na tem področju**.

# Priprave 1. kolokvij (4/10)

3) Vrste lastnosti v diagramu lastnosti (angl. feature diagram)? (4)

# Priprave 1. kolokvij (4/10)

3) Vrste lastnosti v diagramu lastnosti (angl. feature diagram)? (4)

- ✓ Obvezne lastnosti
- ✓ Neobvezne lastnosti
- ✓ Alternativne lastnosti
- ✓ Ali lastnosti

# Priprave 1. kolokvij (5/10)

Naštejte odločitvene vzorce pri razvoju DSLjev? (9)

# Priprave 1. kolokvij (5/10)

Naštejte odločitvene vzorce pri razvoju DSLjev? (9)

- ✓ **Notacija**
- ✓ **Avtomatizacija nalog**
- ✓ **Predstavitev strukture podatkov**
- ✓ **Sprehodi po strukturi podatkov**
- ✓ **Čelni del sistema**
- ✓ **Interakcija**
- ✓ **Razvoj GUI**
- ✓ **Vzorec AVOPT**
- ✓ **Produktna linija**

# Priprave 1. kolokvij (6/10)

V orodju MPS predstavitev koncepta naredimo z ...

# Priprave 1. kolokvij (6/10)

V orodju MPS predstavitev koncepta naredimo z **editor-jem**.



# Priprave 1. kolokvij (7/10)

Za naslednji jezik podan z EBNF zapišite dva RAZLIČNA primera programov (za eno osebo in za dve osebi):

PEOPLE ::= PERSON+  
PERSON ::= (PID SEX)? FIRST MIDDLE? LAST  
PID ::= number  
SEX ::= female | male  
FIRST ::= name  
MIDDLE ::= name  
LAST ::= name

# Priprave 1. kolokvij (7/10)

Za naslednji jezik podan z EBNF zapišite dva RAZLIČNA primera programov (za eno osebo in za dve osebi):

```
PEOPLE ::= PERSON+  
PERSON ::= (PID SEX)? FIRST MIDDLE? LAST  
PID ::= number  
SEX ::= female | male  
FIRST ::= name  
MIDDLE ::= name  
LAST ::= name
```

**Program a)**

**tomaz kosar**

**Program b)**

**1234 male janez janko novak**

**marija novak**

# Priprave 1. kolokvij (8/10)

Za program b) iz naloge 4) zapišite abstraktno sintaksno drevo.

PEOPLE ::= PERSON+

PERSON ::= (PID SEX)? FIRST MIDDLE? LAST

PID ::= number

SEX ::= female | male

FIRST ::= name

MIDDLE ::= name

LAST ::= name

**Program b)**

**1234 male janez janko novak**

**marija novak**

# Priprave 1. kolokvij (9/10)

Za spodnjo BNF gramatiko zapišite pretvorbo v EBNF (uporabite samo neterminala P in C):

$P ::= \text{begin } CS \text{ end}$

$CS ::= C S CS \mid \text{epsilon}$

$S ::= \#int \mid \text{epsilon}$

$C ::= \text{left} \mid \text{right} \mid \text{up} \mid \text{down}$

# Priprave 1. kolokvij (9/10)

Za spodnjo BNF gramatiko zapišite pretvorbo v EBNF (uporabite samo neterminala P in C):

$P ::= \text{begin CS end}$

$CS ::= C S CS \mid \text{epsilon}$

$S ::= \#int \mid \text{epsilon}$

$C ::= \text{left} \mid \text{right} \mid \text{up} \mid \text{down}$

**$P ::= \text{begin ( C \#int? )}^* \text{end}$**

**$C ::= \text{left} \mid \text{right} \mid \text{up} \mid \text{down}$**

# Priprave 1. kolokvij (10/10)

Kako se v TextGen imenuje metoda s katero dodamo tekst v generirano besedilo?



# Priprave 1. kolokvij (10/10)

Kako se v TextGen imenuje metoda s katero dodamo tekst v generirano besedilo?

append

# Naslednja predavanja

- Ponedeljek, 18.11.2024



# Vprašanja

