

MPS Workshop

Till Schallau (till.schallau@tu-dortmund.de)

14. September 2020

The following exercise are designed to give you an in-depth understanding for the creation of languages using the MPS language workbench.

1 Setup - Console

1. Download the latest version of MPS for your operating system from <https://www.jetbrains.com/de-de/mps/download/>
2. Clone the *mps-workshop* repository using `git@github.com:tillschallau/mps-workshop.git` or <https://github.com/tillschallau/mps-workshop.git>
3. Open the project in MPS with `File >> Open`
4. When prompted: `migrate` the project
5. Right-click on the language `de.tudo.cs.ls14.aqua.mps.workshop.java` `>> Rebuild Language` to make the language available for the sandbox solution
6. Right-click on the project `mps-workshop` `>> Rebuild project`

2 Setup - IDE

1. Download the latest version of MPS for your operating system from <https://www.jetbrains.com/de-de/mps/download/>
2. Open MPS
3. Select `Get from Version Control`
4. Enter <https://github.com/tillschallau/mps-workshop.git> into the `URL` field, select a directory and press `clone`
5. When prompted: `migrate` the project
6. Right-click on the language `de.tudo.cs.ls14.aqua.mps.workshop.java` `>> Rebuild Language` to make the language available for the sandbox solution
7. Right-click on the project `mps-workshop` `>> Rebuild project`

3 MPS Hands-On

These exercises are designed to give you a feeling of how the projectional editor of MPS is working. For the following tasks, work on the `↔ master` branch. Navigate to the model `sandbox` under the solution `de.tudo.cs.ls14.aqua.mps.workshop.java` and create a new `JavalnMPSTest` node as shown in Figure 1.

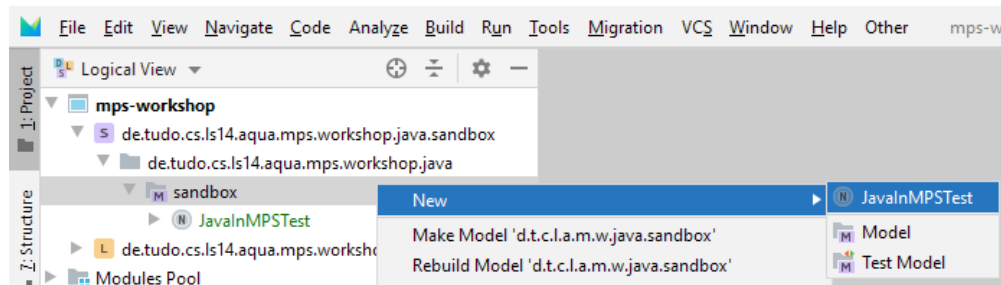


Figure 1: Navigation menu for creation of a new `JavalnMPSTest` node

Exercise 3.1. Implement the code shown in Listing 1 in your `JavalnMPSTest` node. Notice the change in the predefined code fragment while editing the name of the method. After completion rebuild the model by right-clicking `sandbox` `Rebuild model` . When there are no more syntax and compilation errors, run the code by right-clicking onto `JavalnMPSTest` `Run 'Node de.tudo.cs.ls14...'` .

Note: You can check the generated code by right-clicking onto the editor and select `Preview Generated Text` .









```
1 public static void readLine(){
2     String name = "";
3     try{
4         System.out.println("Hello! Please enter your name."
5                             ↪ );
6         BufferedReader br = new BufferedReader(new
7                             ↪ InputStreamReader(System.in));
8         name = br.readLine();
9         System.out.println("Hello " + name);
10    } catch(Exception e){
11        throw e;
12    }
13 }
```

Listing 1: Java code to read a line from the console and print the result

Class	::=	Function*	BExp	::=	BExp && BExp
					BExp BExp
Function	::=	ID = AExp			AExp == AExp
		ID = BExp			AExp != AExp
					AExp ≥ AExp
AExp	::=	AExp + AExp			AExp > AExp
		AExp − AExp			AExp ≤ AExp
		AExp * AExp			AExp < AExp
		AExp / AExp			True
		INT			False
		ID //variable			ID //variable






Table 1: BNF of arithmetic and boolean expressions

Exercise 3.2. Be creative! Add a new  `JavaInMPSTest` node and add your own code.


Note: If your desired package does not exist in your current context, you can add *Accessories Models* by right-clicking the language  `de.tudo.cs.ls14.aqua.mps.workshop.java`  `Module Properties`. Navigate to  `Runtime`  `Accessories Models` and add the required models using the  button, e.g.  `java.time@java_stub`. You have to rebuild the language  `de.tudo.cs.ls14.aqua.mps.workshop.java`  `Rebuild Language` to apply your change.

4 Expressions


This section is designed to show you the most important core aspects of MPS. Each sub-exercise can be pulled from their respective branch, such that each task can be done based on a shared code base.

Exercise 4.1. Create a new language named `de.tudo.cs.ls14.aqua.mps.workshop.expressions` by right-clicking onto  `mps-workshop`  `New`  `Language`. Check the `Create Sandbox Solution` in the process. You have to build the language  `de.tudo.cs.ls14.aqua.mps.workshop.expressions`  `Rebuild Language` to remove errors.

Exercise 4.2. Translate the BNF of Table 1 into concepts. You may use abstract concepts to simplify the task and add internal structure. Give each concept a meaningful **name** and **short description**.

For the next exercise you can checkout the branch  `expression-0-concepts` to work with the shared solution, or work with your own concepts.

Exercise 4.3. Add editors for your concepts, such that the order of operations is clearly visible.

Note: For some visual modifications you may need the  `Inspector`. To show it, click on the respective button in the lower right corner of your IDE.

Intermediate solution: [↔ expression-1-editors](#)

Exercise 4.4. Add intentions for the **and**- and **or**-Expression which transforms a given node into the other and vice versa.

Intermediate solution: [↔ expression-2-intentions](#)

Exercise 4.5. Add transformations that automatically append a node into the AST after typing a specific keyword. Write transformations for the **and**- and **or**-Expression for both directions (left and right).

Intermediate solution: [↔ expression-3-transformations](#)

Exercise 4.6. Add checks to ensure that each function name is unique. Add checks to ensure that each variable is unique for each function.

Intermediate solution: [↔ expression-4-checks](#)

Exercise 4.7. Add generators for at least one of **AExp** or **BExp**. Your code construct for the expression **x >= 2** should look like the following Listing 2.

The final solution with all steps can be found under [↔ expression-5-generations](#)

```

1 package de.tudo.cs.ls14.aqua.mps.workshop.expressions.
    ↪ sandbox;
2
3 /*Generated by MPS */
4
5 import java.io.BufferedReader;
6 import java.io.InputStreamReader;
7
8 public class ExpressionTest {
9     public static void main(String[] args) {
10         test();
11     }
12     public static void test() {
13         BufferedReader reader = new BufferedReader(new
            ↪ InputStreamReader(System.in));
14         int x = -1;
15
16         try {
17             System.out.println("Please enter value for int-
                ↪ variable " + "x");
18             String line = reader.readLine();
19             x = Integer.parseInt(line);
20         } catch (Exception e) {
21             }
22         boolean z = (x >= 2);
23         System.out.println("The result of your function
            ↪ test is: " + z);
24     }
25 }

```

Listing 2: Generated Java code that parses the variables and returns the result of the defined expression