BKPI B-LANGUAGE COMPILER

QUICK USER'S GUIDE

Dear user,

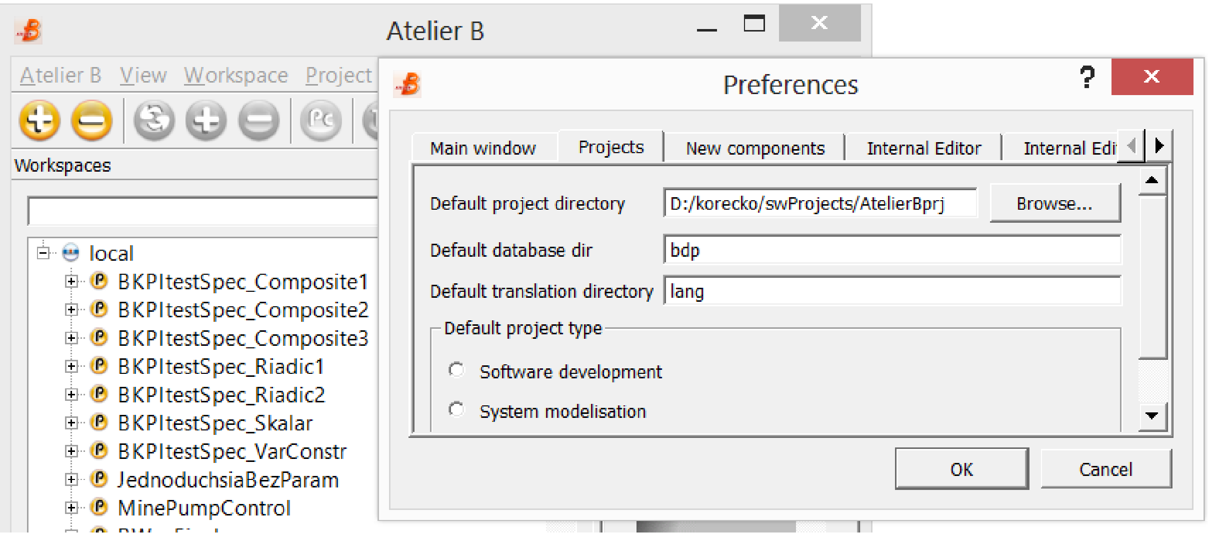
Thank you for showing interest in our compiler that translates implementation-level specifications in B-language to Java and C#. It also provides additional functionality, but this guide only shows how to compile implementations in B-language. The BKPI compiler is based on the work by Bruno Tatibouet, namely on his compiler jBTools. It still uses some of his basic machines.

The BKPI compiler can be found in the `BKPICompiler` folder, which contains the following important files and folders:

- **BKPIcompiler.jar** – executable file of the compiler (in the form of Java archive).
- **compiler_lib** – a folder with libraries necessary for the compiler to run.
- **BasicMachines** – a folder with basic B-machines and their implementations in Java and C#. They are in more detail described in section Basic Machines.

PROJECTS DIRECTORY

Before showing how to compile your implementations we have to clarify the term "projects directory". The projects directory is a folder which contains your B-Method projects. It is expected that each project occupies a separate subfolder of the projects directory and this subfolder contains specification files of the project (machines, refinements and implementations). If you use Atelier B tool for your B-Method projects then this folder should usually be the same as Atelier B's "Default projects directory". For example, if my folder with Atelier B projects is `D:\korecko\swProjects\AtelierBprj`:

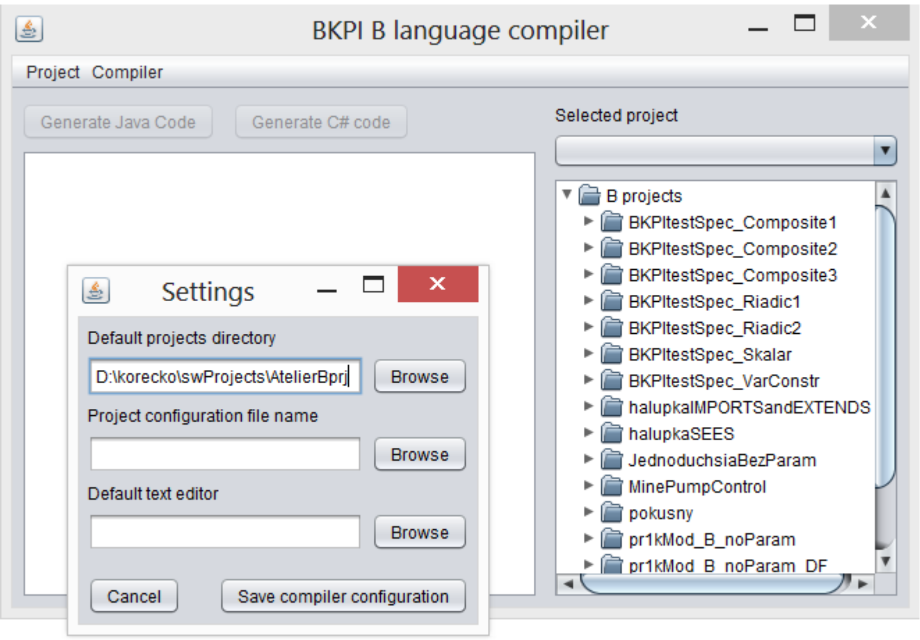


Default project directory: D:/korecko/swProjects/AtelierBprj

Default database dir: bdp

Default translation directory: lang

then this folder will also be the (default) projects directory in my BKPI compiler:



Default projects directory: D:\korecko\swProjects\AtelierBprj

Project configuration file name:

Default text editor:

```
B projects
  BKPItestSpec_Composite1
  BKPItestSpec_Composite2
  BKPItestSpec_Composite3
  BKPItestSpec_Riadic1
  BKPItestSpec_Riadic2
  BKPItestSpec_Skalar
  BKPItestSpec_VarConstr
  halupkaIMPORTSandEXTENDS
  halupkaSEES
  JednoduchsiaBezParam
  MinePumpControl
  pokusny
  pr1kMod_B_noParam
  pr1kMod_B_noParam_DF
```

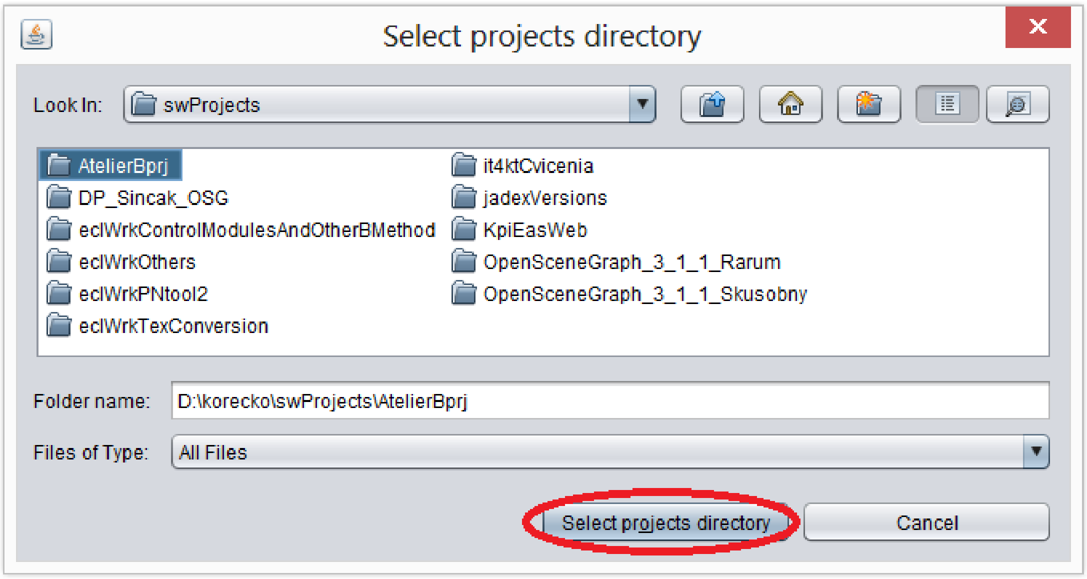
RUNNING BKPI B-COMPILER FOR THE FIRST TIME

Please, follow these steps to run the compiler and set up the (default) projects directory:

1. Go to the `BKPICompiler` folder and run `BKPIcompiler.jar`.

2. Choose *Compiler/Change projects directory* from the main menu.

3. Select the desired folder and click the *Select projects directory* button.



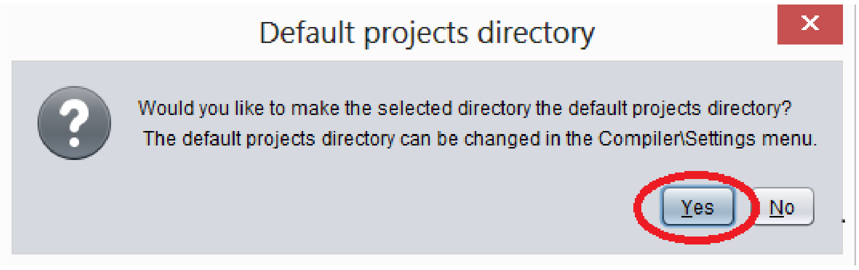
Look In: swProjects

```
AtelierBprj                           it4ktCvicenia
DP_Sincak_OSG                         jadexVersions
eclWrkControlModulesAndOtherBMethod   KpiEasWeb
eclWrkOthers                          OpenSceneGraph_3_1_1_Rarum
eclWrkPNtool2                         OpenSceneGraph_3_1_1_Skusobny
eclWrkTexConversion
```

Folder name: D:\korecko\swProjects\AtelierBprj

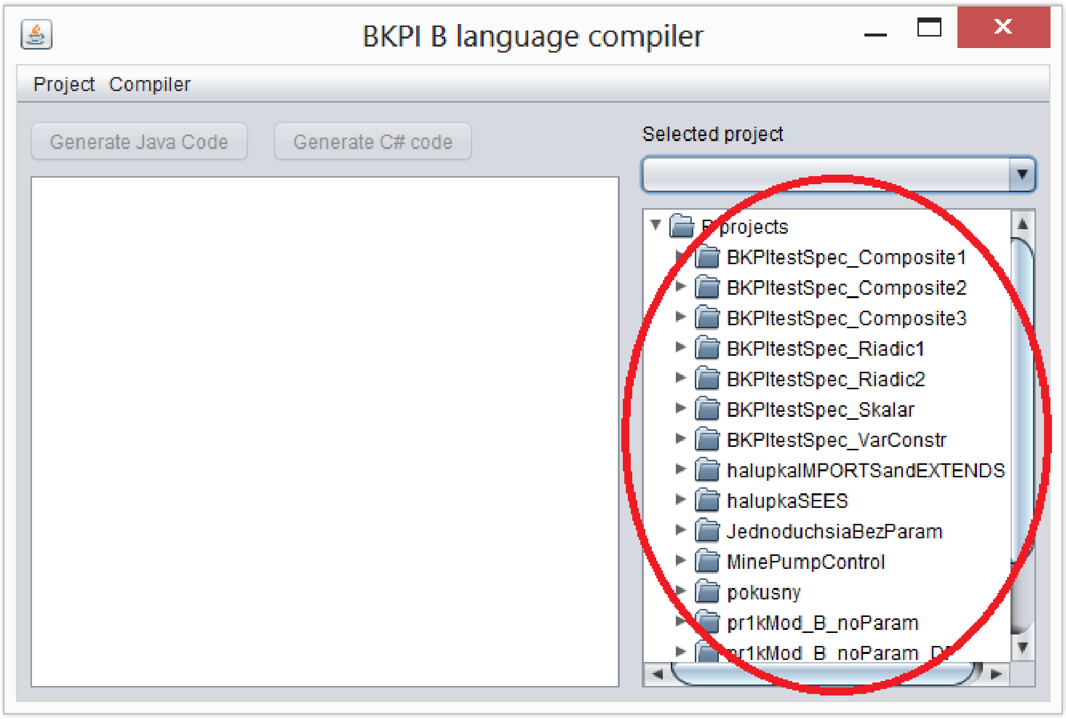
Files of Type: All Files

4. In the dialog that pops up click *Yes*.



Would you like to make the selected directory the default projects directory?
The default projects directory can be changed in the Compiler\Settings menu.

5. ***Projects list***, similar to that in Atelier B, should appear in the right part of the compiler window.



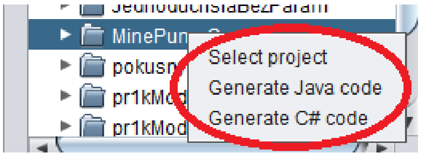
To change the project directory, use the same menu item. To set up the default project directory you can also use the setting dialog available from the *Compiler/Settings* menu.

COMPILING A B-METHOD PROJECT

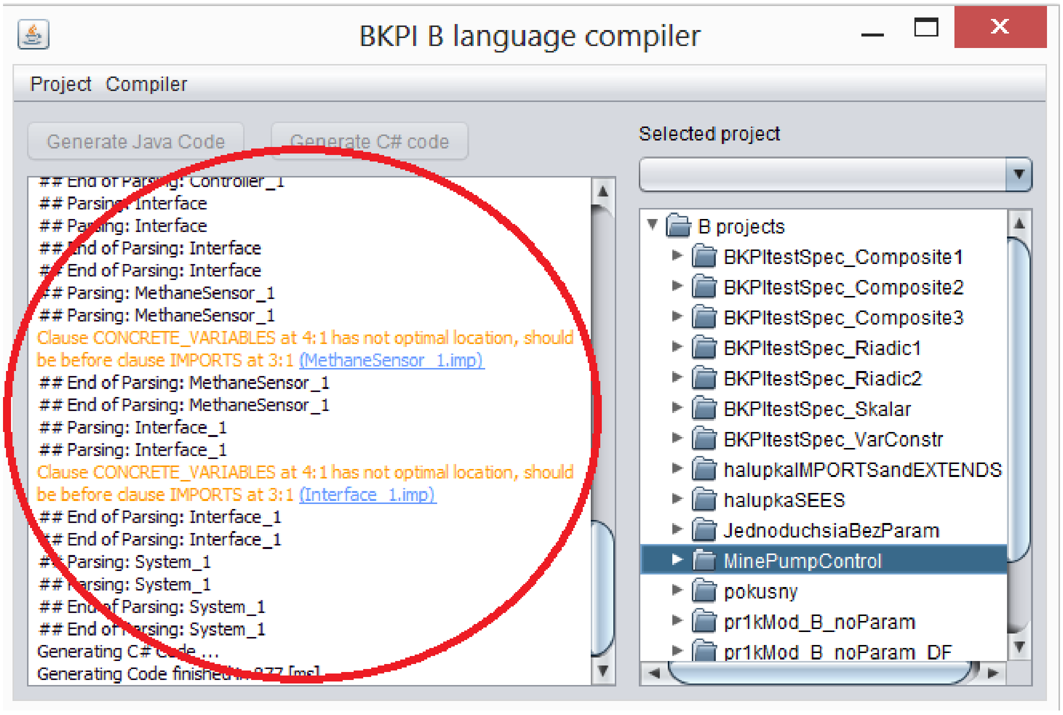
1. Run the BKPI compiler and make sure that a proper projects directory is set.

2. Go to the projects list in the right part of the compiler window.

3. Right click on the project you wish to compile. A context menu appears:



4. Select *Generate Java code* from the context menu if you wish to translate your implementation to Java or *Generate C# code* if you wish to translate it to C#. A compilation report appears in the left part of the compiler window



```
## End of Parsing: Controller_1
## Parsing: Interface
## Parsing: Interface
## End of Parsing: Interface
## End of Parsing: Interface
## Parsing: MethaneSensor_1
## Parsing: MethaneSensor_1
Clause CONCRETE_VARIABLES at 4:1 has not optimal location, should
be before clause IMPORTS at 3:1 (MethaneSensor_1.imp)
## End of Parsing: MethaneSensor_1
## End of Parsing: MethaneSensor_1
## Parsing: Interface_1
## Parsing: Interface_1
Clause CONCRETE_VARIABLES at 4:1 has not optimal location, should
be before clause IMPORTS at 3:1 (Interface_1.imp)
## End of Parsing: Interface_1
## End of Parsing: Interface_1
## Parsing: System_1
## Parsing: System_1
## End of Parsing: System_1
## End of Parsing: System_1
Generating C# Code ...
Generating Code finished in 977 [ms]
```

and a source code is generated to the java or c# subfolder of the project folder.

BASIC MACHINES

Application program interface for software developed by B-Method is provided in the form of so-called *basic machines*. These machines don't have implementations, just a source code in selected programming languages. The BKPI compiler contains the following basic machines:

- `BASIC_IO` – operations for console input and output.
- `BT_IO` – additional input and output operations.
- `BT_SET` – sets implementation.
- `BT_SEQUENCE` – sequences implementation.

The machines with the prefix "BT" have been originally defined by Bruno Tatibouet.

The `BasicMachines` folder contains 3 subfolders:

- `MCH` – basic machines in B-language. If you need to use a basic machine in your B-Method project, then the corresponding .mch file from this folder have to be added to your project.
- `Java` – source code (subfolder `src`) and compiled versions (subfolder `bin`) of the basic machines in Java. These files should be used in your Java projects that use code generated from B-language. Instead of using separate files you can also include `BKpiBasic.jar` library.

- `C#` - source code of the basic machines in C#. These are used in C# projects that use code generated form B-language.

You can also notice that the `Java` and `C#` folders also contain additional classes `ArrayUtils` and `Interval`. Methods of these classes are often used in generated code, so you should add them in your Java or C# project even if no basic machines are used.