# Installation und Einrichtung

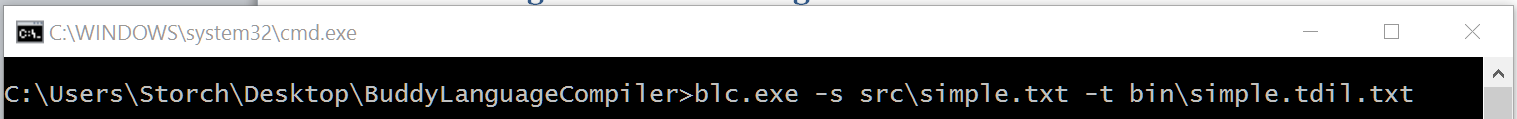
Zur Installation und Einrichtung des Programms reicht es, das ZIP-Paket irgendwo auf dem System zu entpacken. Damit das Programm ausführbar ist, wird mind. .NET 4.0 auf dem System benötigt.

# Programmausführung

Das Programm wird über die Konsole gestartet. Zur Ausführung benötigt es zwei Parameter:

* Verweis auf die zu übersetzende Datei (Quelle)
* Verweis auf den Ort und Namen der Zieldatei (Ziel)

Ausgehend vom entpackten Verzeichnis (siehe „Installation und Einrichtung“) kann das Programm beispielsweise wie folgt gestartet werden:



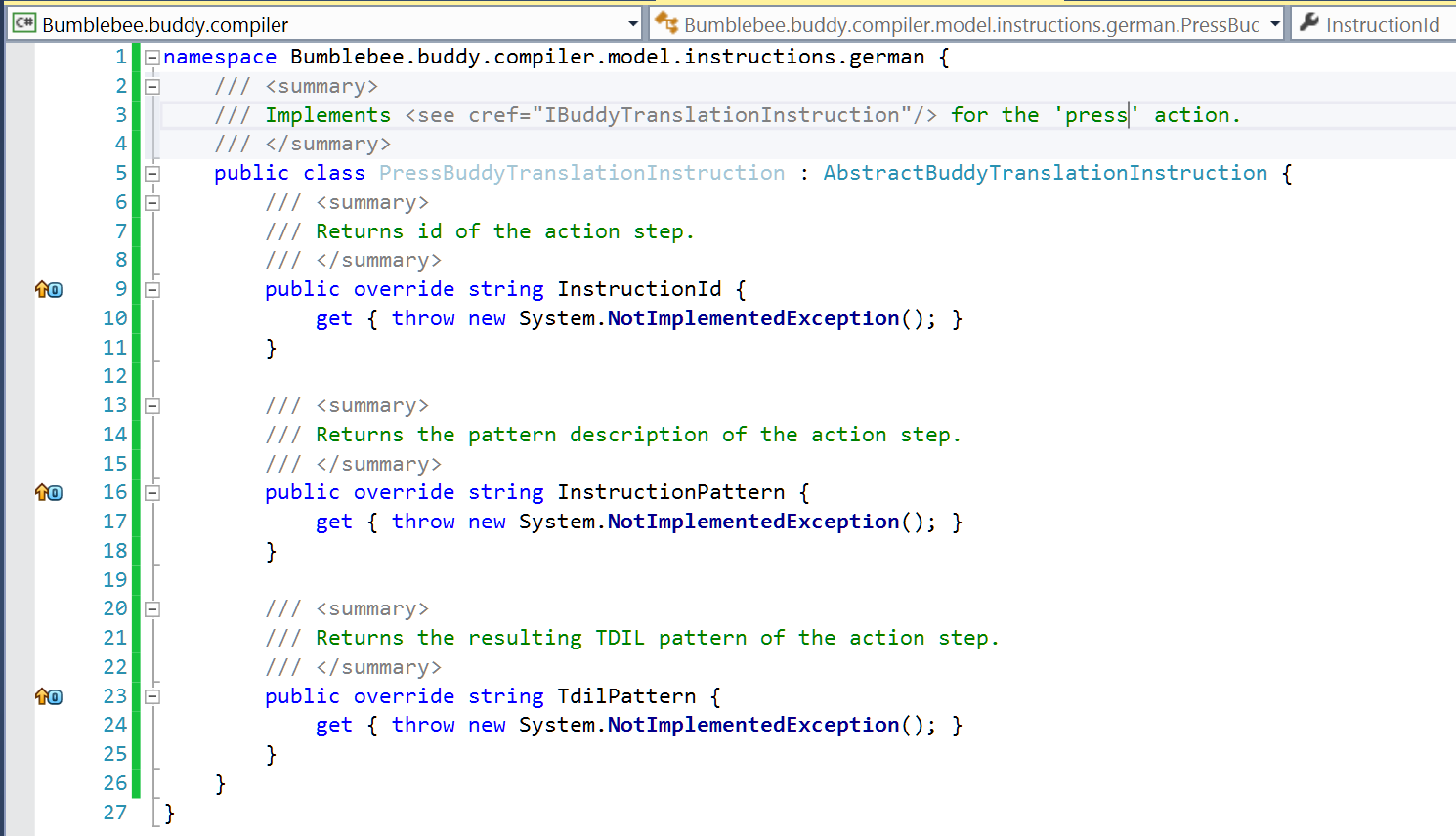
Von Haus aus stehen die folgenden Stilmuster zur Verfügung:

# Eigene Musterübersetzung definieren

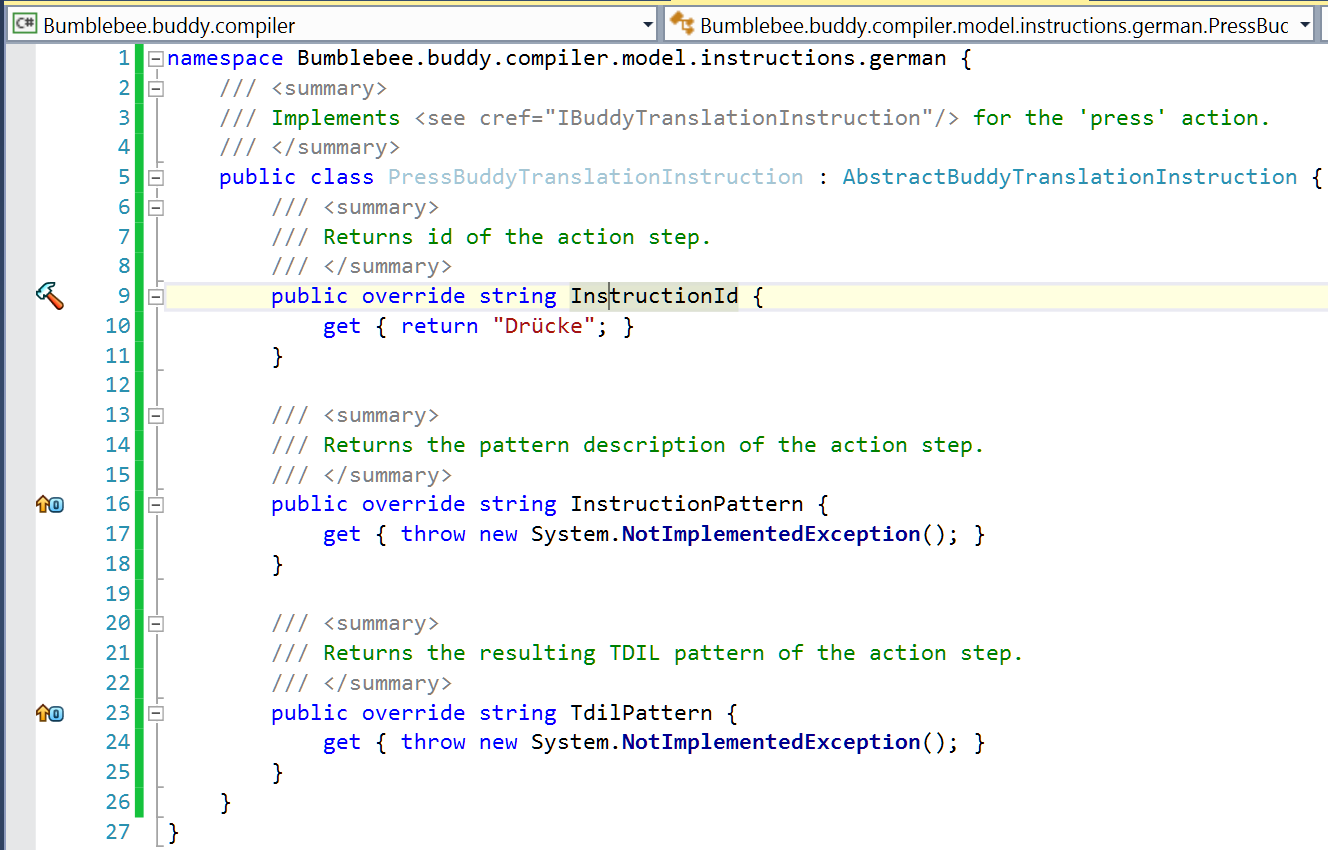
Soll das Programm um neue bzw. eigene Übersetzungsmuster erweitert werden, so ist dies einfach zu realisieren. Allerdings muss dafür der Quellcode erweitert werden. Dies geschieht in mehreren Schritten.

## Klasse zur Musterbeschreibung implementieren

Die Klassen zur Musterbeschreibung befinden sich im Projekt „Bumblebee.buddy.compiler“ unter dem Namespace „model/instructions/german“. Für eine neue Musterbeschreibung legen wir dort eine neue Klasse ein. In unserem Beispiel soll die Klasse „PressBuddyTranslationInstruction.cs“ heißen. Die Klasse erbt von der abstrakten Basisklasse „AbstractBuddyTranslationInstruction“ um Coderedundanzen zu vermeiden. Der Klassenrumpf sieht damit zunächst wie folgt aus:



Die neue Musterbeschreibung soll Formulierungen zum Beschreiben von Tastaturanweisungen (z. Bsp. „Drücke die Taste Tab zweimal.“) übersetzen. Daher lautet die eindeutige „InstructionId“ „Drücke“. Dies können wir somit in die Klasse einsetzen. Die Implementierung der Klasse sieht nun wie folgt aus:

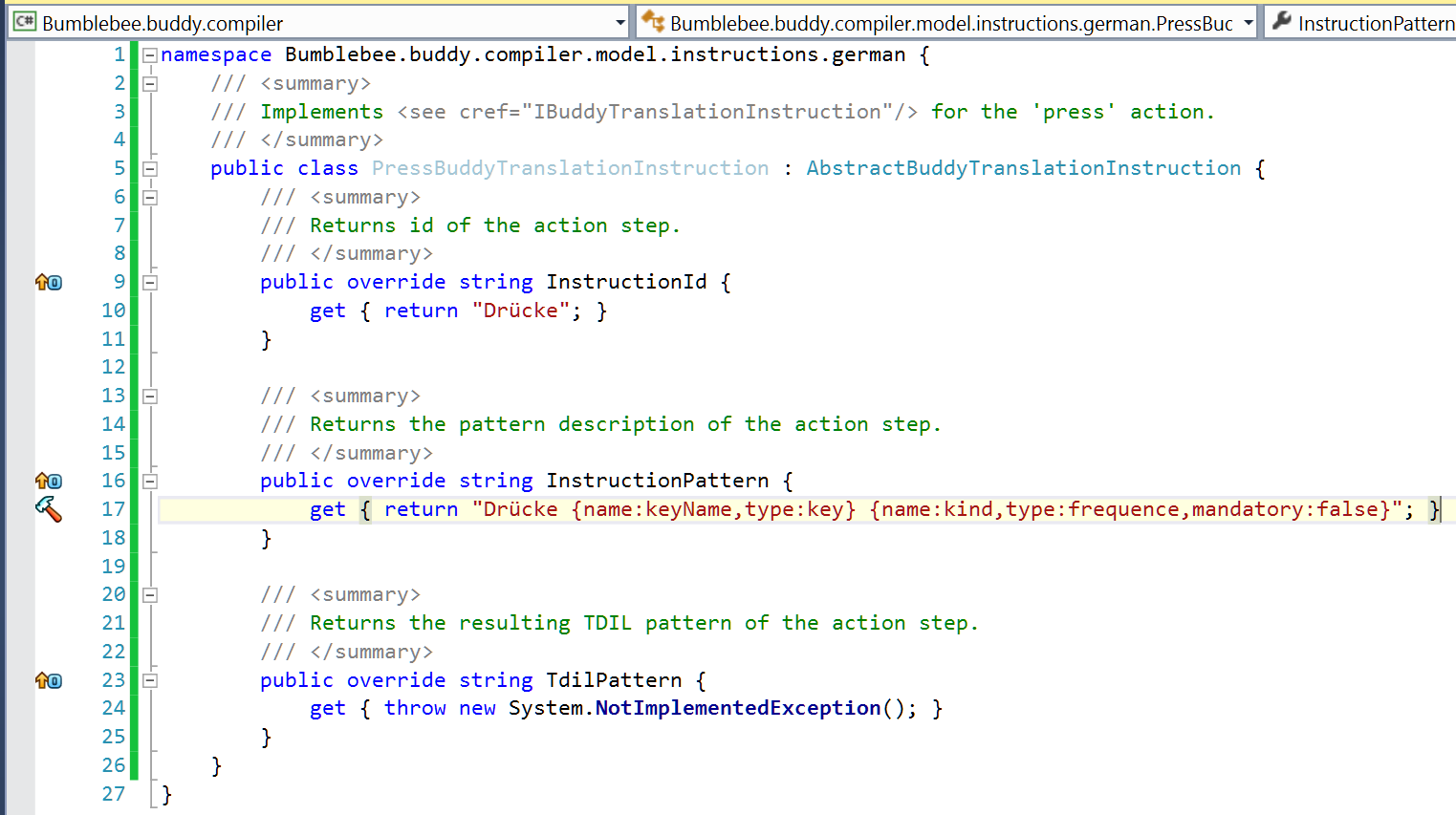


Als nächstes muss nun das Muster der natürlichen Sprache für diese Anweisung beschrieben werden. Dies wird in der Eigenschaft „InstructionPattern“ hinterlegt. Das Muster für die Anweisung gestaltet sich wie folgt:

“Drücke {name:keyName,type:key} {name:kind,type:frequence,mandatory:false}”

Ausdrücke in geschweiften Klammern stehen für Parameter. Demnach hat dieses Muster zwei Parameter. Der erste steht für die Taste, die gedrückt werden kann. Der zweite steht für die Frequenz (bspw. einmal, zweimal, usw.).

Jede Parameterbeschreibung hat einen festen Aufbau. Dieser besteht aus einer Sequenz von Parametereigenschaften, die per Key-Value-Muster beschrieben werden und kommasepariert sind. Dabei befindet sich der Name der Eigenschaft jeweils vor dem Doppelpunkt und der Wert dafür dahinter. Jeder Parameter muss mind. die Eigenschaft „name“ und „type“ deklarieren. Demnach hat der erste Parameter in unserem Musten den Namen „keyName“ und ist vom Typ „key“. Der zweite Parameter hat den Namen „kind“ und ist vom Typ „frequence“. Zusätzlich ist der zweite Parameter nicht verpflichtend. Dies wird anhand der Eigenschaft „mandatory“ festgelegt. Mit diesem Muster sieht der Klassenrumpf nun wie folgt aus:

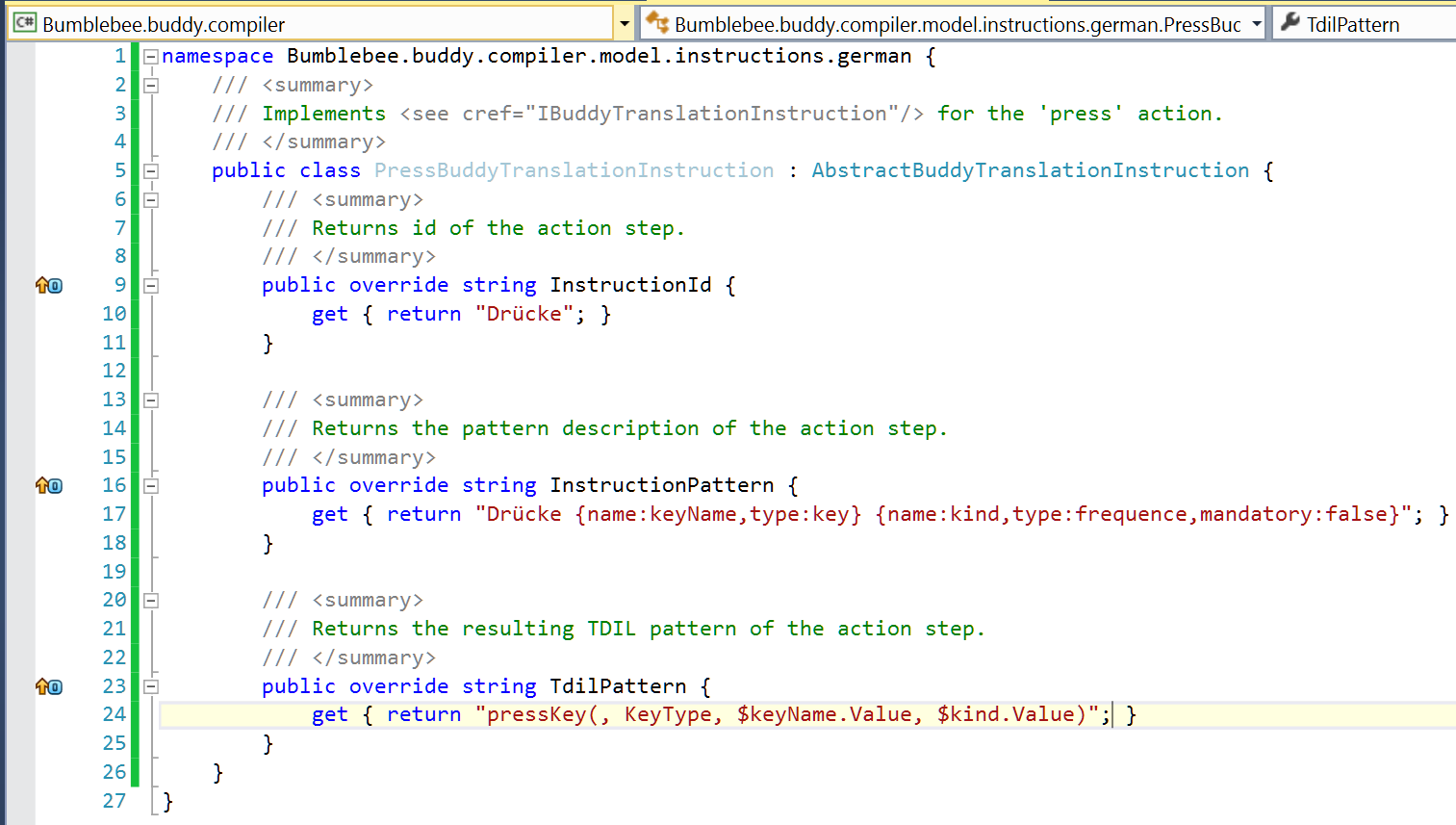


Nun muss das korrespondierende TDIL-Muster definiert werden. Dies erfolgt mittels der Eigenschaft „TdilPattern“. Jede IL-Anweisung muss das sog. AAPD-Muster (Action-Alias-Property-Datavector) erfüllen. Die „Action“ beschreibt die Aktion, die durch den Testtreiber (TD) ausgeführt werden soll. Der Wert „Alias“ bezeichnet das UI-Element, das Ziel der Aktion wird. Wenn dieses nicht gegeben ist, so ist die Anwendung an sich das Ziel der Aktion. Die Eigenschaft „Property“ bezeichnet die Eigenschaft des UI-Elements (bspw. Text, Visibility, usw.), die durch die Aktion verändert werden soll. Diese kann leer sein, wenn bspw. kein UI-Element referenziert ist (alias ist leer). Die Eigenschaft „Datavector“ liefert die für jede Aktion notwendigen Daten. Dieser ist abhängig von der jeweiligen Aktion und ist niemals leer! Weitere Informationen zur Grammatik und Aufbau der Intermediate Language (IL) befinden sich in den Dokumenten „EBNF\_TDIL31.pdf“ und „TDIL10.pdf“.

Um das AAPD-Muster für unsere Anweisung zu erfüllen, muss der Musterrumpf somit wie folgt aussehen:

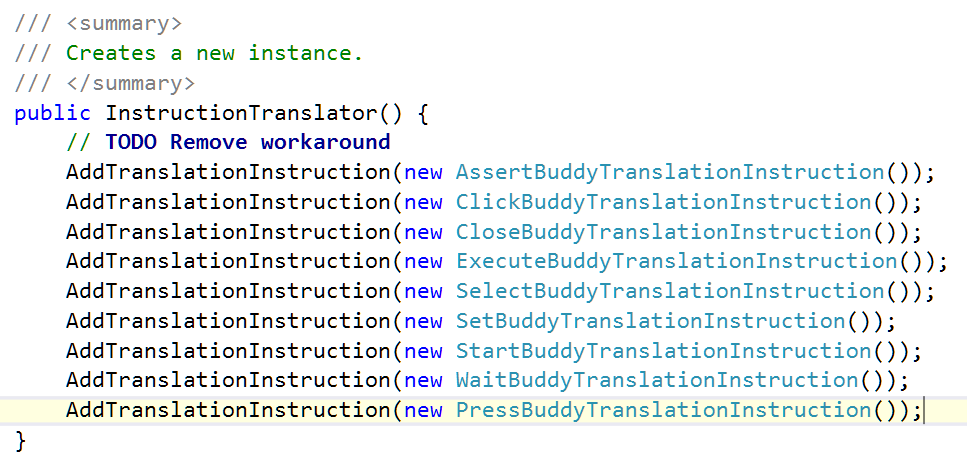
"pressKey(, KeyType, $keyName.Value, $kind.Value)"

Die „Action“ für das TDIL-Muster lautet somit “pressKey”. Ein „Alias“ ist nicht gegeben und somit leer. Die „Property“ für diese IL-Anweisung lautet „KeyType“. Der Datenvektor besteht aus zwei Werten, die sich aus dem Muster für die natürlich sprachliche Anweisung (s.o. „InstructionPattern“) ergeben. Das Dollarzeichen „$“ signalisiert den Verweis auf einen Parameter. Danach folgt der Name des Parameters. Dieser muss in der Musterbeschreibung für die natürlich sprachliche Anweisung vorkommen (siehe Parametereigenschaft „name“). Mittels der „Punktnotation“ lassen sich Eigenschaften eines Parameters abrufen. In dem Fall reicht es allerdings, einfach nur den Wert des Parameters zu verwenden. Daher lautet der Parameterverweis hier schlichtweg „$keyName.Value“ bzw. „$kind.Value“. Durch die Einsetzung des IL-Musters ist das Übersetzungsmuster nun vollständig. Die fertige Implementierung sieht nun wie folgt aus:



## Registrierung des neuen Übersetzungsmusters

Nachdem die Implementierung für das neue Übersetzungsmuster fertiggestellt ist, muss es noch registriert werden. Dies erfolgt in der Klasse „InstructionTranslater.cs“, welche sich direkt im Wurzelverzeichnis des Projekts „Bumblebee.buddy.compiler“ befindet. Im Konstruktor der Klasse werden alle bereits vorhandenen Übersetzungsmuster registriert. Die neue muss dieser Stelle lediglich ergänzt werden. Damit sieht der erweiterte Konstruktor nun wie folgt aus:

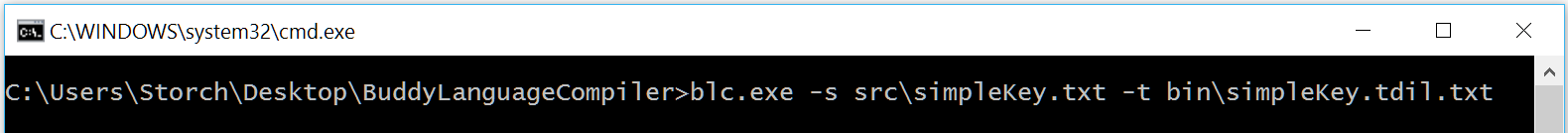


## Testen des neuen Übersetzungsmusters

Abschließend soll das neue Übersetzungsmuster getestet werden. Dies lässt sich mithilfe eines einfachen Testfalls realisieren. Wir verwenden dafür den folgenden Testfall:

|  |
| --- |
| Anwendung: HelloWorld  Anwendungsfall: Tastendruck  Szenario: Einfacher Tastendruck  Schritte: Drücke STRG+SHIFT+S doppelt. Schliesse Anwendung. |

Dafür erstellen wir im Installationsverzeichnis (siehe „Installation und Einrichtung“) im Ordner „src“ eine neue Textdatei namens „simpleKey.txt“ und kopieren den zuvor beschriebenen Testfall hinein. Anschließend führen wir den Buddy Language Compiler wie folgt aus:



Das Ergebnis (siehe simpleKey.tdil.txt) sollte in etwa so aussehen:

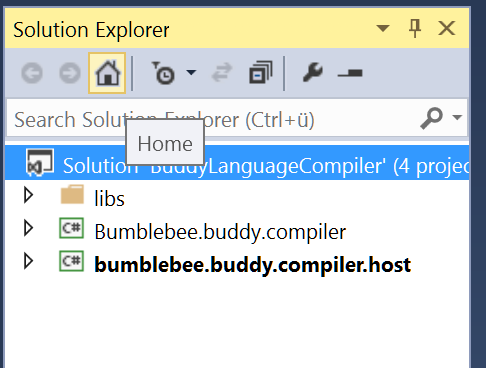
|  |
| --- |
| // Compiler generated file  // Buddy Compiler version 0.1  // Generated on 10.10.2016 09:24:40  Unit HelloWorld.Tastendruck.Einfacher\_Tastendruck  Main:  processHandle1 = start(,, "{HelloWorld}")  pressKey(, KeyType, "STRG+SHIFT+S", Double)  close(\_Application,, Default)  End  End |

# Einrichtung der Entwicklungsumgebung

Um Änderungen am Quellcode vornehmen, muss zuvor die Entwicklungsumgebung eingerichtet werden. Als Entwicklungsumgebung empfiehlt sich eine aktuelle Visual Studio Version – zum Beispiel „Visual Studio Community 2015“. In dem Fall wird .NET 4.6 benötigt.

Zunächst muss das ZIP-Paket mit den Quellen „BuddyLanguageCompiler-src.zip“ irgendwo auf dem System entpackt werden. Anschließend muss die dort enthaltene Solution-Datei „BuddyLanguageCompiler.sln“ in Visual Studio geöffnet werden. Sie befindet sich im Unterordner „Bumblebee“.

Nach dem Öffnen der Solution-Datei sollte der Solution-Explorer von Visual Studio in etwa wie folgt aussehen:



Sollte der Eintrag „bumblebee.buddy.compiler.host“ nicht fett gedruckt erscheinen, so lässt sich das wie folgt beheben. Führen Sie einen Rechtsklick auf den Eintrag durch und wählen Sie den Menüpunkt „Set as StartUp Project“. Damit ist dieses Projekt als Startprojekt, welches die Main-Methode beinhaltet, gekennzeichnet.