Ukážkové príklady ku XML modelu návrhového vzoru

Tento dokument bol vytvorený ako príloha ku bakalárskej práci Filip Gschwandtnera s názvom **Java a návrhové vzory**. Jeho úlohou je pomocou ukážkových príkladov umožniť lepšie pochopiť súborový XML model návrhového vzoru, ktorý je popísaný v už spomínanej bakalárskej práci.

Príklad č.1:
Prvý príklad bude ukazovať ako sa dá zadefinovať rozhranie(interface).

```
1 .<?xml version="1.0" encoding="UTF-8"?>
2 .<design_pattern name="PatternName">
3 ..<clientobjects>
4 ...<interface name="interfaceName" parent="ParentName">
5 ....<interfaceVariable name="InterfaceName1"/>
6 ....<interfaceVariable name="InterfaceName2"/>
7 ....<interfaceVariable name="InterfaceName3"/>
8 ....<interfaceVariable name="InterfaceName4"/>
9 ....<interfaceVariable name="InterfaceName5"/>
10....<method name="InterfaceMethodName" is_constructor="false">
11.....<return_type>void</return_type>
12.....<list_of_parameters>
13.......<parameter type="ParamType1">ParameterName1</parameter>
14.......<parameter type="ParamType2">ParameterName2</parameter>
15.....</list_of_parameters>
16....</method>
17...</interface>
18..</clientobjects>
19..<nonclientobjects>
20..</nonclientobjects>
21.</design_pattern>
```

Riadok `1` je len úvodným riadkom každého XML súbora. Riadok `2` definuje návrhový vzor s menom `PatternName`. Riadky `3` až `18` definujú objekty, s ktorými pomyselný klient návrhového vzoru pracuje. Riadky `19` a `20` definujú zvyšné objekty návrhového vzoru (v tomto prípade žiadne). Na riadkoch `4` až `17` sa definuje rozhranie(interface) s menom `interfaceName` a rodičovským rozhraním `ParentName`. Rozhranie v modeli môže mať maximálne 1 rodiča. Ak chceme, aby rozhranie nemalo definovaného rodiča, tak na miesto mena rodiča napíšeme `null`. Riadky `5` až `9` definujú 5 interfacových premenných s menami `InterfaceName1`, ..., `InterfaceName5`. Tieto premenné sú ponímané ako `public`, `static` a `final`. Riadky `10` až `16` definujú metódu pre interface. Metóda nie je konštruktor, má názov `InterfaceMethodName` (riadok `10`) a typ návratovej hodnoty je `void` (riadok `11`). Typ návratovej hodnoty môže byť ešte primitívum (`int`, `boolean`,...). Na riadkoch `12` až `15` sa definujú parametre metódy (v tom poradí ako majú byť v metóde uvedené). Riadky `13` a `14` definujú 2 parametre. Prvý parameter má meno `parameterName1` a je typu `ParamType1`. Druhý parameter je definovaný podobne.

V skutočnosti je model metódy väčší avšak pre interface je takýto osekaný. Plný model metódy demonštrujeme na ďalšom príklade.

Príklad č.2:
Druhý príklad bude ukazovať plný model metódy nadefinovanom vo vnútri triedy.

```
1 .<?xml version="1.0" encoding="UTF-8"?>
2 .<design_pattern name="Unknown">
3 ..<clientobjects>
4 ...<object name="MethodSample" parent="ParentName" abstract="false">
5 ....<template>this will be ignored
6 .....int templateoutside;
7 ......int templateoutside2;
8 ....</template>
9 ....<method name="MethodName" is_constructor="false">
10.....<return_type>void</return_type>
11.....<list_of_parameters>
12......<parameter type="ParamType1">ParameterName1</parameter>
13......<parameter type="ParamType2">ParameterName2</parameter>
14.....</list_of_parameters>
15.....<demands>
16......<template>
17.......int templateinsidemethod1;
18......</template>
19......<call_method use_reference="ReferenceName" method_name="MethodName"
                     is_return_call="false" is_list_command="false">
20.......<list_of_parameters_for_call>
21........<use_reference>ReferenceName</use_reference>
22........<use_parameter>ParName1</use_parameter>
23........<this/>
24.......</list_of_parameters_for_call>
25......</call_method>
26......<template>
27.......int templateinsidemethod2;
28......</template>
29......<call_method use_reference="ParameterName" method_name="MethodName"
                     is_return_call="false" is_list_command="false">
30.......<list_of_parameters_for_call>
31........<use_reference>ReferenceName</use_reference>
32........<use_parameter>ParName1</use_parameter>
33........<this/>
34.......</list_of_parameters_for_call>
35......</call_method>
36......<call_method use_reference="super" method_name="MethodName"
                     is_return_call="false" is_list_command="false">
37.......<list_of_parameters_for_call>
38.......</list_of_parameters_for_call>
```

```
39......</call_method>
40......<call_method use_reference="null" method_name="MethodName"
                     is_return_call="false" is_list_command="false">
41.......<list_of_parameters_for_call>
42.......</list_of_parameters_for_call>
43......</call_method>
44......<call_method use_reference="new" method_name="ObjectName"
                     is_return_call="true" is_list_command="false">
45.......<list_of_parameters_for_call>
46.......</list_of_parameters_for_call>
47......</call_method>
48......<added_behaviour/>
49.....</demands>
50....</method>
51...</object>
52..</clientobjects>
53..<nonclientobjects>
54..</nonclientobjects>
55.</design_pattern>
```

Riadky 1-3, 52-55 boli už vysvetlené v predchádzajúcom príklade. Na riadkoch 4 až 51 definujem neabstraktnú triedu s menom `MethodSample` a predkom `ParentName` (informácie riadku 4). Parameter `label` tagu `object` je nepovinný. Ako prvé , pri výpise triedy do templatu, bude vypísané vo vnútri triedy text nadefinovaný v tagu `template` na riadkoch 5 až 8. Text `"This will be ignored"` bude ignorovaný, pretože leží na riadku, kde je začiatok tagu `template`. Text `"int templateoutside;"` bude odsadený v template ako aj premenné (keby tam niake boli). Text `"int templateoutside2;"` bude odsadený o jeden odsek(tab) doprava oproti predošlému riadku.
Metóda zadefinovaná na riadkoch 5 až 50 je podobná definovaniu metódy v predošlom príklade. Rozdiel nastáva len v tagu `demands` (riadky 15-49), ktorý predstavuje požiadavky na vnútro metódy. Požiadavky na metódu treba definovať v takom poradí, v ako ich chceme, aby boli zisťované pri hladaní v návrhového vzoru či vypísane do templatu tejto triedy. Na riadkoch 16-18,26-28 sú znova templaty, ktoré ale pre zmenu sa vypíšu do vnútra metódy. Na riadkoch 19 až 25 je ukážková požiadavka na zavolanie inej metódy. Riadok 19 hovorí, že táto metóda sa volá `MethodName` a volá sa pomocou referencie `ReferenceName` (t.j. `ReferenceName.MethodName(...)`). Hovorí tiež, že sa návratová hodnota volanej metódy nemá brať ako návratová hodnota celej metódy (t.j. nie je to prípad `return ReferenceName.MethodName(...)`). Spomínaný riadok hovorí tiež o prípade, keď uvedená referencia je zoznam. Vtedy sa má daná metóda zavolať pre každý prvok zoznamu. Ak by parameter `is_list_command` bol `true`, tak by sa metóda chovala ako príkaz na zoznam samotný. Riadky 20 až 24 definujú parametre aké sa majú použiť na zavolanie metódy. Môžu to byť referencia (riadok 21), parameter metódy,v ktorej chceme zavolať metódu,(riadok 22) alebo trieda, ktorú práve definujem (t.j. `this`), (riadok 23).
Bloky riadkov 29-35, 36-39, 40-43, 44-47 predstavujú pár alternatív využitia

tagu `call_method`. Prvý zo spomínaných blokov volá metódu pomocou parametra metódy, ktorú definujem, namiesto referencie. Druhý blok volá metódu predka triedy. Tretí blok volá metódu triedy, ktorú práve definujem. Štvrtý blok vytvára inštanciu triedy `ObjectElement` a nastavuje ju ako návratovú hodnotu celej metódy, ktorú definujem.
Riadok `48` definuje ďalší typ požiadavky.

Príklad č.3:
Tretí príklad bude ukazovať definovanom referencií, high level listov, bližšie nešpecifikovaných premenných(uvar) vo vnútri triedy.

```
1 .<?xml version="1.0" encoding="UTF-8"?>
2 .<design_pattern name="AllFeatures">
3 ..<clientobjects>
4 ...<object name="HighLevelListSample" parent="ParentName"
             abstract="true">
5 ....<is_high_level_list_to>ListElementType</is_high_level_list_to>
6 ....<unspecified_variable name="VariableName1">
7 .....<object_with_access>ObjectName1</object_with_access>
8 .....<object_with_access>ObjectName2</object_with_access>
9 .....<client_access/>
10....</unspecified_variable>
11....<reference name="ReferenceName1" reffers_to="ObjektName1"
                 is_list="false" assign_from="null">
12.....<object_with_access>ChangeObjectName1</object_with_access>
13.....<object_with_access>ChangeObjectName2</object_with_access>
14.....<client_access/>
15....</reference>
16...</object>
17..</clientobjects>
18..<nonclientobjects>
19..</nonclientobjects>
20.</design_pattern>
```

Časti tohto príkladu mimo riadkov `5` až `15` boli už v predošlých príkladoch vysvetlené. Riadok `5` definuje vysoko úrovňový list pre triedu `ListElementType`. Riadky `6` až `10` definujú premennú s menom `VariableName1`, ku ktorej majú prístup triedy `ObjectName1`, `ObjectName2` a ešte klient. Na riadkoch `11` až `15` sa definuje referencia (s menom `ReferenceName1`) na triedu `ObjectName1`. Referencia nie je zoznam a tiež nemá nastaveného nikoho špeciálneho, kto by ju inicializoval. Riadky `12` až `14` hovoria o podobnej prístupnosti referencie ako v prípade premennej `VariableName1`.