# 1. Dop. Post. Programování DU1

Vypracovali: Štěpán Balcar, Filip Řepka

## Třída Token

U této třídy byl změněn způsob ukládání flagů. Považujeme za velmi nepraktické uchovávat dvoustavovou informaci jako jeden bit integeru. Flagy byly převedeny do podoby booleanu a k hromadné manipulaci s nimi byly přidány nové metody. Výsledkem je přehlednější kód, jednodušší manipulace s daty bez bitových posunů. Nevýhodou je horší paměťová složitost. Integer sice zabírá 4 byty, ale boolean v Javě celý jeden byte. Tudíž deset booleanů bude zabírat 10 bytů. Kdybychom chtěli zabránit nárůstu paměti Tokenu a zároveň nechtěli přijít o jednoduché rozhraní, mohli bychom využít původní způsob uchování dat pomocí bitů v integeru s tím, že třída Token by tvořila fasádu pro práci s flagy. Pro každý flag by vznikla nová funkce pro nastavení a přečtení hodnoty flagu. To nám ale při zanedbatelně velkém zýšení paměťové náročnosti jednoho Tokenu nepřipadalo nutné.

Proměnné uvnitř třídy byly přesunuty, tak aby jejich pořadí vypovídalo o účelu proměné. Token „prev„ a „next„ byly jako implementační proměnné přesunuty až na konec.

Konstruktor třídy neobsahuje třetí parametr int flags, ale instanci třídy TokenFlagModel, která slouží jako struktura pro předávání flagů. Flagy změnou reprezentace jde nyní efektivně předat pouze jako skupina argumenů narozdíl od předchozího jednoho integeru. Samotná třída token se neodkazuje na žádný TokenFlagModel. Argumenty jsou předány a překopírovány do proměnných hodnotového typu. Dělá se to z toho důvodu, aby se nemusel zaobalovat jednoduchý způsob přistupování k datům v Tokenu.

Nově přibyly dvě metody setTokenFlagModel a getTokenFlagModel, které slouží pro jednoduchý export a import hodnot flagů zabalených ve třídě TokenFlagModel. Import spočívá v překopírování hodnot flagů z modelu do Tokenu a u exportu dochází opět ke kopírování hodnotových proměnných z Tokenu do nově vytvořeného modelu.

Funkce clone byla změněna, protože nový Token musí být volán nově s argumentem TokenFlagModel.

## Třída TokenFlagModel

Tato třída byla nově vytvořena jako datový model pro flagy. Třída obsahuje jen bezparametrický konstruktor, který vrátí instanci třídy se všemi flagy nastavenými na hodnotu false. Dále obsahuje proceduru pro vytváření průniku dvou modelů. Tato funkce dostane dva modely a vrátí logické AND jednotlivých flagů.

## Třída IndentLvl

Tato třída slouží jako jeden record ze spojového seznamu reprezentující jednotlivé úrovně odsazení. Tato třída byla původně vnitřní třídou ve třídě Indent. Pro přehlednost a jednoduchost byla třída vyextrahována ven. Proměnná třídy „klass„ byla přejmenována na type byl u ní změněn formát ze stringu na enum.

## Třída Indent

Do třídy byla přidána konstanta INDENT-SIZE, a byly změněny i funkce ve třídě.

## Podtřída IndentContext

Třída slouží jako datová struktura popisující aktuální stav odsazení. Číselné hodnoty ve funkcích byly nahrazeny konstantami, byly zrušeny nevhodně navržené deklarace proměnných, které se používají pouze na pár po sobě jdoucích řádcích programu. Přejmenované byly také proměnné, které nejednoznačně vypovídaly o svém účelu. Byly rozepsány spletité logické výrazy z jejichž podvýsledků uložených v pomocných proměnných lze mnohem snadněji pochopit směr kterým se ubíhají větve programu.

## Funkce skipWhitespaceAndComments

Funkce obsahovala příziž velkou podmínku v kulatých závorkách forcyklu. Příkaz byl přeformulován a rozepsán.

## Funkce skipUntil

V této funkce byl opět rozepsán forcyklus skládající se z několika podmínek do čitelnější podoby while cyklu s omezujícími podmínkami

## Fukce changeColUntilEOL

Ve funkci byla zrušena deklarace proměnné. Proměnná byla zadeklarována až přo prvním použití.

## Funkce ensureBlankLineAfter

Ve funkci byl použit jiný způsob s manipulací s flagy a byl odstraněn příkaz while s následujícím switchem. Zdrojový kód šlo upravit do daleko jednodušší a čitelnější podoby bez použití těchto příkazů.

## Funkce indentLine

Tato funkce obsahovala velké množství vnořených podmíněných skoků. Funkce v nové podobě slouží jen jako rozstřel pro volání dalších procedur. Toto oddělení prospělo přehlednosti kódu. Oddělování bylo komplikovanější, protože deklarace proměnných byla provedena hned na začátku funkce a využívalo se vyditelnosti

těchto proměnných pro předávání výsledků. V důsledku oddělení problémů vznikly nové funkce creatEmptyString, indentLineExtension, indentLineShorening, indentLineCreating.

## Funkce indentComments

Funkce byla přeformulována z bloku několika vložených forcyklů a podmíněných příkazů pouze na průchod toků, se začátkem daným na vstupu a zavoláním procedury indentComment na každý jednotlivý token.

Funkce volá další dvě funkce isCommentStandAlone a firstWhiteTokenAtThisLine, které pomáhají s procházením spojového seznamu a specifikují pozici komentáře. Funkce také nově využívá enum jako seznamu rezervovaných slov.

## Funkce skipUntil

Ve funkci jsme změnili příkaz while na forcyklus. Tuto úpravu jsme provedli i v následujících funkcích, kde podmínka forcyklu nekonkrétně vystihovala počet iterací.

## Ostatní

V celém zdrojovém kódu byl sjednocený styl definování proměnných a indentace.