PV178 - DOMÁCÍ ÚKOL #2

VÁŠ ÚKOL

1. Přidejte události do třídy `Robot` i do rozhraní `IRobot`.
   a. Přidejte událost pro chyby robota, které je potřeba zobrazit uživateli ve formě textové zprávy (narazil jsem do zdi, není co zvednou).
   b. Přidejte událost pro sledování činnosti robota tak, aby při obsluze události bylo možné rozlišit, o jakou akci šlo (krok, vlevo vbok, položil značku nebo zvednul značku).
   c. U obou událostí půjde určit robota, který událost vyvolal.
   d. Při vytváření rozhraní aplikujte konvence pro události v C#/.NET (obě).
   e. Ze tříd robotů odstraňte všechny vazby na třídu `Console`.
   f. Implementuje rozhraní `ITracer` ve třídě, která bude zapisovat jak zprávy, tak chování robota. (HINT: Ve třídě `Console` je statická vlastnost `Out`, která stojí za prozkoumání.)
2. Naprogramuje třídy pro reprezentaci struktury programu v jazyce Karel a jednoduchý překladač pro jazyk EasyKarel.
   a. Implementuje rozhraní `IProgram`, `IStatement` a `ICondition` ve třídách reprezentujících strukturu program tak, abyste byli schopni program v jazyce Karel spustit.
   b. Pro primitivní instrukce vytvořte a použijte jedinou společnou třídu implementující `IStatement`, s pomocí delegáta `Action<T>`,
   c. Implementujte `IStatement` pro vykonání seznamu jiných příkazů.
   d. Implementujte rozhraní `IStatement`, které na základě výsledku rozhraní `ICondition` vykoná jeden ze dvou příkazů.
   e. Implementuje rozhraní `ICondition`, aby robot mohl provést rozhodnutí.
   f. Implementujte rozhraní `IParser` tak, aby jeho metoda vracela objekt představující strukturu správně vytvořeného programu v jazyce Karel.
   g. Parser vždy předpokládá správný vstup a při chybě vyhazuje odpovídající výjimku.
3. Ve třídě `Program` vytvořte metodu, která:
   a. načte ze zadaného souboru program v jazyce EasyKarel,
   b. tento program vykoná s hloupým robotem, ve městě 10x10 políček a
   c. chování robota bude zapisovat pomocí objektu implementujícího `ITracer`, který metoda dostane jako parametr.
   d. V případě chyby vypište hlášení o chybě na konzoli (v pěkném formátu a s informací o chybě).
4. Přepište metody `ToString`, aby vracely smysluplné informace.
5. Doplňte dokumentaci svých tříd.
6. Vlastní i dodané třídy můžete rozšiřovat a měnit, ale jen tak, aby to neodporovalo smyslu úlohy. Změny rozhraní, pokud nejsou vyžadovány zadáním, vám musí dovolit cvičící.
7. Použijte své znalosti OOP k dosažení robustního, stabilního a krátkého kódu.

LEXIKÁLNÍ ANALYZÁTOR

Rozhraním `IScanner` implementované třídou `Scanner` představuje lexikální analyzátor, který zpracuje vstup programu, rozdělí ho na jednotlivá slova jazyka a zařadí do kategorií. Při tom také odstraní z programu všechny komentáře. Výstupem třídy je posloupnost objektů (Token), kde každý z nich odpovídá slovu z jazyka EasyKarel:

- Klíčová slova jsou `PROGRAM`, `AS`, `BEGIN`, `END`, `IF`, `THEN`, `ELSE`, `NOT`.
- Identifikátory - názvy programu, instrukcí a testů.
- Jako poslední se vždy vrací token označující konec souboru.

Vaším úkolem je z objektů vrácených třídou `Scanner` vytvořit strom objektů, představujících strukturu programu.

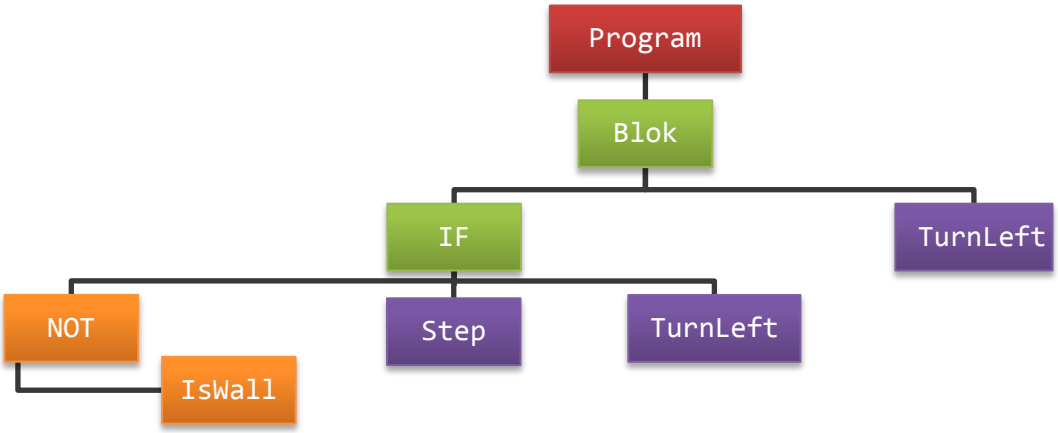
JAZYK EASYKAREL

Program v jazyce EasyKarel je posloupnost příkazů, které robot vykomá. Struktura programu je pevně daná a zapsaný program vypadá nějak takto:

```
PROGRAM Demo AS                    ' Hlavička programu

BEGIN                              ' Tělo programu
  IF NOT IsWall THEN
    Step
  ELSE
    TurnLeft
  TurnLeft
END
```

Tento program pak lze objekty reprezentovat následovně (červeně je třída implementující `IProgram`, oranžově jsou podmínky (`ICondition`), a zelená i fialová odpovídají příkazům (`IStatement`)):



Program začíná klíčovým slovem `PROGRAM`, za kterým následuje název programu, klíčové slovo `AS`, po něm příkaz představující tělo programu. Příkaz (statement) může být instrukce robota, blok příkazů nebo podmíněný příkaz.

Identifikátor může představovat **instrukci pro robota**. Robot rozumí 4 instrukcím těchto názvů:

- `Step` (krok vpřed),
- `TurnLeft` (otočení o 90 ° vlevo),
- `PickSign` (zvednout značku),
- `PutSign` (polož značku).

Blok příkazů ohraničený klíčovými slovy `BEGIN` a `END`, mezi kterými je libovolné množství dalších příkazů.

Podmíněný příkaz

```
IF <podmínka> THEN <příkaz1>
IF <podmínka> THEN <příkaz1> ELSE <příkaz2>.
```

Za klíčovým slovem `IF` následuje podmínka, `THEN` příkaz vykonaný pokud je podmínka splněna, a příkaz za `ELSE` se vykoná pokud podmínka splněna není. Robot umí vyhodnotit následující podmínky i jejich negaci (pokud je před testem `NOT`):

- `IsWall` (přímo před robotem je zeď),
- `IsNorth` (robot je otočen na sever),
- `IsSign` (pod robotem je značka).

Při parsování jazyka EasyKarel lze vždy podle aktuálního tokenu jednoznačně rozhodnout, jak by měl parser pokračovat. Parsování také může skončit také jednou z následujících výjimek:

- `UnexpectedEndOfFileException` – parser předčasně narazil na konec souboru,
- `UnexpectedTokenException` – parser očekával jiný token, než který byl na řadě,
- `UnreachableCodeException` – za tělem programu je kód navíc.
- `UnknownInstructionException` – parser narazil na neznámou instrukci.
- `UnknownTestException` – parser narazil na neznámý test.

Definice jazyka EasyKarel v BNF-DP vypadá takto:

```
      <program> ::= PROGRAM <identifikator> AS <prikaz>
       <prikaz> ::= <blok> | <if> | <instrukce>
    <instrukce> ::= <identifikator>
         <blok> ::= BEGIN (<prikaz>)* END
           <if> ::= IF <podminka> THEN <prikaz> |
                    IF <podminka> THEN <prikaz> ELSE <prikaz>
     <podminka> ::= <identifikator> | NOT <identifikator>
<identifikator> ::= Posloupnost písmen a číslic bez mezer, začínající písmenem
```

Zápis **`(<prikaz>)*`** znamená, že se `<prikaz>` může opakovat 0 až libovolně krát. Znaky `* ( )` v tomto případě nepatří do jazyka EasyKarel, ale k BNF-DP.