B-OOP: Úloha č. 4

Vytvorte novú triedu s názvom **Task4**, ktorá obsahuje nasledovné premenné a funkcie:

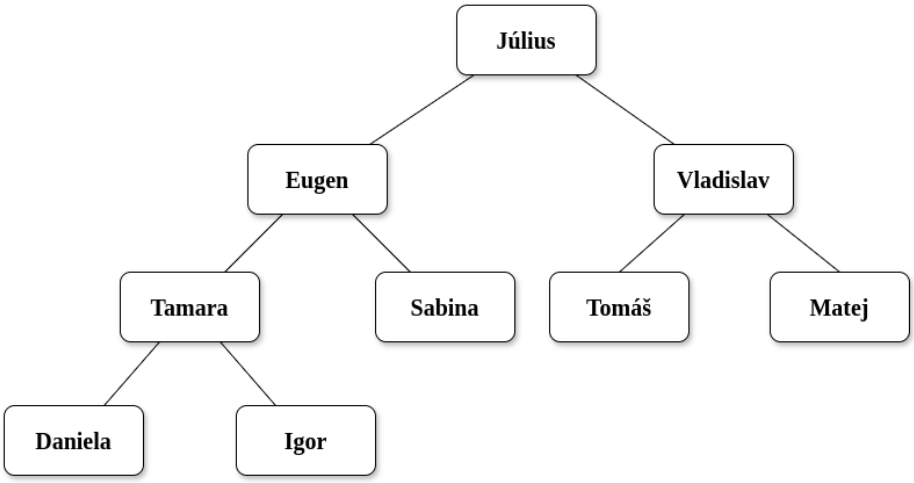
- metódu **main** s korektnou signatúrou
- statickú metódu **change**
  - bez návratovej hodnoty
  - so vstupom **int[] arr**
  - metóda zmení druhý prvok poľa na hodnotu 42. V prípade, že pole nie je dosť dlhé, funkcia neurobí nič.
  - v metóde **main** vytvorte pole **int[] a = 0,1,2,3;** a vypíšte jeho obsah do konzoly. Zavolajte metódu **change** s poľom **a** a potom vypíšte jeho obsah. Zamyslite sa, čo sa udialo s poľom **a** a prečo.
- druhú statickú metódu **change**
  - bez návratovej hodnoty
  - so vstupom **String s**
  - metóda bude obsahovať iba jeden riadok kódu: **s = "hello";**
  - v metóde **main** vytvorte reťazec **str** s hodnotou **"jello"**. Zavolajte funkciu **change** s parametrom **str** a potom ho vo funkcii **main** vypíšte. Čo sa stalo a prečo?
  - Všimnite si, že teraz sú vo vašej triede dve funkcie s rovnakým názvom. Ide o príklad polymorfizmu.

Vytvorte enum **Color**. Pridajte do neho možnosti **RED**, **GREEN**, **BLUE**.
Vytvorte triedu **ColorPicker**. Do tejto triedy pridajte metódu **main** s korektnou signatúrou. Ďalej pridajte:

- statickú metódu **printColor**
  - bez návratovej hodnoty
  - so vstupom **Color c**
  - táto metóda vypíše "Red", ak je c rovné **Color.RED**, "Green", ak je c rovné **Color.GREEN**, "Blue", ak je c rovné **Color.BLUE**
- statickú metódu **pickRandomColor**

- s návratovou hodnotou typu **Color**
- bez vstupov
- táto metóda vygeneruje náhodnú farbu a vráti ju

Ďalej vytvorte triedu **BinaryTree**. Táto trieda bude reprezentovať binárny strom. Binárny strom je graf, v ktorom má každý uzol nanajvýš 2 deti. Pre lepšiu predstavu sa môžete pozrieť na Obr. 1. Tento strom je možné reprezentovať obyčajným poľom. Za týmto účelom očíslujeme jednotlivé uzly podľa obrázku 2. Vytvoríme pole **String[] tree = new String[9];** Koreň stromu má hodnotu Július a číslo 0. Preto do poľa **tree** na index 0 uložíme hodnotu **"Július"**. Ľavé dieťa koreňa je uzol s hodnotou **"Eugen"** a číslom 1. Pravé dieťa je očíslované číslom 2. Ľavé dieťa uzla s číslom 1 je uzol s číslom 3 a pravé dieťa je uzol s číslom 4, atď. Výsledné pole bude mať hodnoty **{"Július", "Eugen", "Vladislav", "Tamara", "Sabina", "Tomáš", "Matej", "Daniela", "Igor"}**

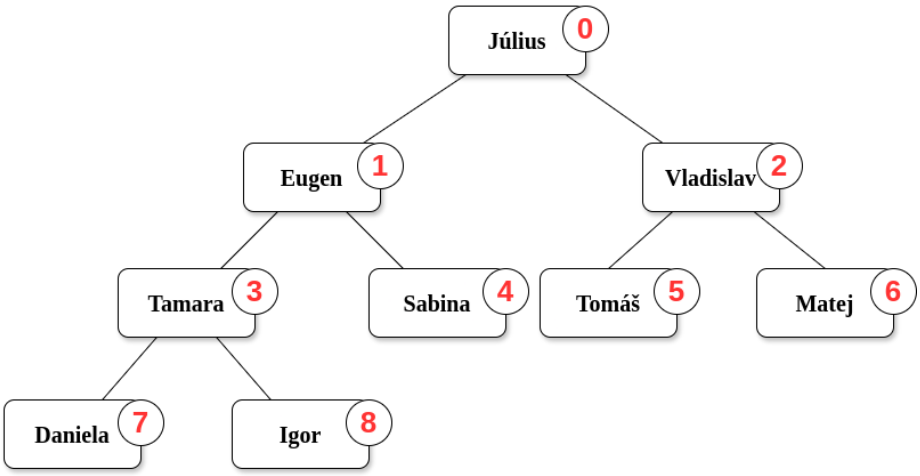


Obr. 1: Binárny strom

Všimnite si, že je možné veľmi jednoducho spočítať indexy detí uzla na základe jeho indexu. Ľavé dieťa i-teho uzlu je na indexe $2i + 1$ a pravé dieťa zasa na indexe $2i + 2$.

V triede **BinaryTree** vytvorte:

- metódu **main** s korektnou signatúrou
  - V tejto metóde vytvorte pole **tree**, ktoré reprezentuje strom z obrázkov 1, resp. 2.
- statickú metódu **getLeftChild**



Obr. 2: Binárny strom s očíslovaním

  - s návratovou hodnotou **Integer**
  - so vstupmi **String[] tree** a **int index**
  - Metóda vráti index ľavého dieťaťa uzla s indexom **index**. Ak také dieťa neexistuje, vráti null.

- statickú metódu **getRightChild**
  - s návratovou hodnotou **Integer**
  - so vstupmi **String[] tree** a **int index**
  - Metóda vráti index pravého dieťaťa uzla s indexom **index**. Ak také dieťa neexistuje, vráti null.

- Vytvorte statickú metódu **inOrder**
  - bez návratovej hodnoty
  - so vstupmi **String[] tree** a indexom **int index**
  - táto metóda rekurzívne prehľadá strom a vytlačí obsah uzlu. Pri prehľadávaní "in order" funkcia najprv zavolá samú seba s parametrami **tree** a indexom ľavého dieťaťa. Potom vytlačí hodnotu uzla na indexe a potom zavolá rekurzívne samú seba so vstupmi **tree** a indexom pravého dieťaťa. Ak je aktuálny index indexom uzlu bez detí, iba sa vytlačí jeho hodnota. Pre uvedený príklad stromu očakávame výstup v poradí **"Daniela", "Tamara", "Igor", "Eugen", "Sabina", "Július", "Tomáš", "Vladislav", "Matej"**

Do AIS odovzdajte zdrojové súbory (s príponou .java).