Test č. 1, úloha č. 2

Stiahnite si priložený ZIP archív obsahujúci kostru niekoľkých tried v balíku `trees`, ktoré reprezentujú uzly stromov *booleovských výrazov* resp. *zložených výrokov* – ide o obdobu aritmetických stromov, kde listy namiesto číselných konštánt alebo premenných zodpovedajú výrokovým premenným a vnútorné uzly namiesto aritmetických operácií zodpovedajú logickým operáciám $\neg, \vee, \wedge$ negácie, disjunkcie a konjunkcie.

Priložený archív obsahuje balík `trees` a v ňom nasledujúce triedy:

- Hotovú *abstraktnú* triedu `Node`, ktorej potomkami sú všetky triedy reprezentujúce uzly stromov booleovských výrazov.
- Hotovú *abstraktnú* triedu `BinaryOperatorNode`, od ktorej dedia triedy reprezentujúce vnútorné uzly zodpovedajúce binárnym logickým operáciám. Trieda poskytuje konštruktor umožňujúci vytvoriť takýto uzol s danými dvoma synmi a metódy `getLeft` a `getRight`, pomocou ktorých možno k týmto dvom synom pristupovať.
- Hotovú *abstraktnú* triedu `UnaryOperatorNode`, od ktorej dedia triedy reprezentujúce vnútorné uzly zodpovedajúce unárnym logickým operáciám. Trieda poskytuje konštruktor umožňujúci vytvoriť takýto uzol s daným synom a metódu `getChild`, pomocou ktorej možno k tomuto synovi pristupovať.
- Kostru triedy `Variable` rozširujúcej triedu `Node` a reprezentujúcej listy stromu zodpovedajúce výrokovým premenným. Každá výroková premenná je určená svojim názvom typu `String`, ku ktorému možno pristupovať pomocou metódy `toString`.
- Kostru triedy `Not` rozširujúcej triedu `UnaryOperatorNode` a reprezentujúcej uzly zodpovedajúce logickej negácii.
- Kostru triedy `Or` rozširujúcej triedu `BinaryOperatorNode` a reprezentujúcej uzly zodpovedajúce logickej disjunkcii.
- Kostru triedy `And` rozširujúcej triedu `BinaryOperatorNode` a reprezentujúcej uzly zodpovedajúce logickej konjunkcii.

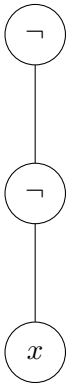
Cieľom tejto úlohy je doprogramovať metódy umožňujúce k stromu zloženého logického výroku zostrojiť strom zodpovedajúci jeho negácii. V abstraktnej triede `Node` je deklarovaná abstraktná metóda `public abstract Node negate();` volanie `node.negate()` tejto metódy pre uzol `node` by malo vrátiť koreň stromu reprezentujúceho výrok, ktorý je negáciou výroku reprezentovaného stromom s koreňom `node`.
Vašou úlohou je doprogramovať telo metódy `negate()` prekrývajúcej túto abstraktnú metódu triedy `Node` v triedach `And`, `Or`, `Not` a `Variable`. Výroky pritom negujte *výhradne nasledujúcim spôsobom*:

- Ak je x výroková premenná, je negáciou výroku x výrok $\neg x$. Pre strom obsahujúci jediný uzol typu `Variable` teda strom jeho negácie pozostáva z koreňa typu `Not`, ktorého jediným synom je pôvodný koreň typu `Variable`.
- Pre ľubovoľný výrok φ je negáciou výroku $\neg\varphi$ výrok φ. Pre strom s koreňom zodpovedajúcim unárnej operácii $\neg$ je teda stromom jeho negácie podstrom zakorenený v jedinom synovi koreňa.
- Pre ľubovoľné dva výroky φ, ψ je negáciou výroku $\varphi \vee \psi$ výrok $\overline{\varphi} \wedge \overline{\psi}$, kde $\overline{\varphi}$ je negáciou φ a $\overline{\psi}$ je negáciou ψ. Ak je teda koreňom stromu disjunkcia, bude koreňom príslušného negovaného stromu konjunkcia, pričom obidva podstromy zakorenené v synoch koreňa negovaného stromu získame rekurzívnym volaním negácie na synov koreňa pôvodného stromu.
- Pre ľubovoľné dva výroky φ, ψ je negáciou výroku $\varphi \wedge \psi$ výrok $\overline{\varphi} \vee \overline{\psi}$, kde $\overline{\varphi}$ je negáciou φ a $\overline{\psi}$ je negáciou ψ. Ak je teda koreňom stromu konjunkcia, bude koreňom príslušného negovaného stromu disjunkcia, pričom obidva podstromy zakorenené v synoch koreňa negovaného stromu získame rekurzívnym volaním negácie na synov koreňa pôvodného stromu.

Okrem správnosti sa budú hodnotiť aj niektoré ďalšie aspekty odovzdaného riešenia – predovšetkým dodržiavanie elementárnych zásad objektovo orientovaného programovania a rešpektovanie konvencií jazyka Java.

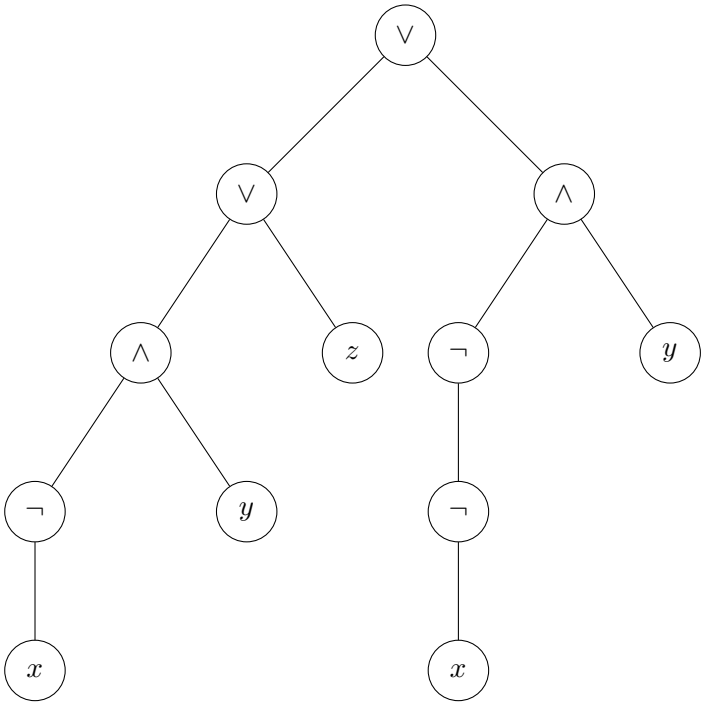
Na testovač odovzdávajte ZIP archív obsahujúci priečinok `trees` a v ňom súbory so zdrojovými kódmi všetkých tried tohto balíka (vrátane tých, ktoré sa oproti kostre nezmenili).

Príklad 1. Strom výroku $\neg\neg x$.



Strom negácie $\neg x$.

Príklad 2. Strom výroku $((\neg x \wedge y) \vee z) \vee (\neg\neg x \wedge y)$.



Strom negácie $((x \vee \neg y) \wedge \neg z) \wedge (\neg x \vee \neg y)$.