Během zkoušky aktivně komunikujte se zkoušejícími, nebojte se zeptat. Nad zadáním přemýšlejte, často je těžší zjistit *co* řešit než *jak*. Kód pište čistě a průběžně ho vylepšujte. Nesnažte se vyřešit všechny příklady naráz — celé zadání nejspíš nestihne nikdo, hodnotit budeme především podle kvality vašich myšlenek a vašeho kódu.

1. Vyrobte strom reprezentující tento výrok:

$$(\forall x \in \{1,2,4\})((\exists y \in \{1,2,4\})(x = 4 \vee x < y)).$$

2. Do rozhraní `Formula` a `Term` dopište podporu pro návrhový vzor návštěvník/visitor. Napište návštěvníka třídy `Term`, který daný term zaserializuje do řetězce.

3. Do rozhraní `Formula` dopište metody `Boolean isTautology()` a `Boolean isContradiction()`. Tyto metody by měly formuli označit jako tautologii nebo kontradikci minimálně v následujících případech:

   - $true$ je tautologie,
   - $false$ je kontradikce,
   - pokud je f tautologie, pak $\neg f$ je kontradikce,
   - pokud je f kontradikce, pak $\neg f$ je tautologie,
   - pokud je f kontradikce, pak i $f \wedge g$ a $g \wedge f$ je kontradikce ,
   - pokud je f tautologie, pak i $f \vee g$ a $g \vee f$ jsou tautologie,
   - pokud je f tautologie, pak i $(\forall x \in s)(f)$ je tautologie,
   - pokud je s prázdná, pak $(\forall x \in s)(f)$ je tautologie,
   - pokud je f kontradikce, pak i $(\exists x \in s)(f)$ je kontradikce a
   - pokud je s prázdná, pak $(\exists x \in s)(f)$ je kontradikce.

4. Vyrobte tovární metody, které při konstrukci detekují tautologie/kontradikce a případně redukují konstruované formule na instance tříd `True`/`False`.

5. Napište metodu `Boolean Formula.isClosed()`, která rozhodne, zda je daná formule uzavřená (tzn. neobsahuje volné proměnné) nebo ne. Tip: chytře využijte návrhový vzor strom/kompozit/composite.

6. Smažte třídu `Addition` a zbylý kód vhodně upravte tak, aby byl použitelný i s termy jiného typu než jen `Integer`.