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, hodnotit budeme především podle kvality vašich myšlenek a vašeho kódu.

Regulární výraz má jednu z následujících podob:

- prázdný výraz, reprezentující řetězec nulové délky,
- řetězec, reprezentující sám sebe,
- dva regulární výrazy napsané za sebou ($r_1 r_2$), reprezentující všechny řetězce, které vzniknou spojením retězců reprezentovaných prvním výrazem a řetězců reprezentovaných druhým výrazem,
- dva regulární výrazy oddělené svislítkem ($r_1|r_2$), reprezentující všechny řetězce reprezentovaných prvním nebo druhým výrazem,
- regulární výraz následovaný hvězdičkou ($r*$), reprezentující všechny řetězce, které se dají složit z libovolného počtu řetězců reprezentovaných regulárním výrazem r,
- regulární výraz následovaný plusem ($r+$), reprezentující všechny řetězce, které se dají složit z nenulového počtu řetězců reprezentovaných regulárním výrazem r,
- regulární výraz následovaný otazníkem ($r?$), reprezentující prázdný řetězec a všechny řetězce, které jsou reprezentované regulárním výrazem r a
- regulární výraz následovaný nenulovým přirozeným číslem v složených závorkách ($r\{n\}$), reprezentující všechny řetězce, které se dají složit z n řetězců reprezentovaných regulárním výrazem r.

1. Naimplementujte metodu `Regexp.desugar`, která z daného regulárního výrazu vyrobí regulární výraz se stejným významem, ale bez výskytu instancí tříd `Plus`, `QuestionMark` a `Braces`. Při nahrazování podvýrazů postupujte podle následujících příkladů:
   (a) podvýraz `r+` jde nahradit podvýrazem `r(r*)`,
   (b) podvýraz `r?` jde nahradit podvýrazem `(|r)` (prázdný řetězec nebo `r`) a
   (c) podvýraz `r{3}` jde nahradit podvýrazem `rrr`.
2. Dopište stavitele `RegexpBuilder`. Zařiďte, aby dvě různá volání metod stavících ten samý regulární výraz (např. `RegexpBuilder.literal("OMO").or(RegexpBuilder.literal("omo")).build()`) vracela ukazatel na tu samou instanci. Sémantickou ekvivalenci výrazů (např. ekvivalenci výrazů `rrr` a `r{3}`) řešit nemusíte.
3. Do rozhraní `Regexp` dopište podporu návrhového vzoru návštěvník (visitor) a napište návštěvníka, který vrátí textovou podobu daného regulárního výrazu včetně korektního uzávorkování.

4. Dopište metodu `Regexp.equalsTo`, při její implementaci použijte dvojité volání (double dispatch). Sémantickou ekvivalenci výrazů řešit nemusíte.

5. Upravte třídy `Plus`, `QuestionMark` a `Braces` tak, aby fungovaly jako adaptéry podle návodu naznačeného v příkladu 1. Zachovejte signatury jejich konstruktorů.

6. Naimplementujte metodu `Regexp.iterator`. Vrácený iterátor by měl postupně vrátit všechny řetězce odpovídající danému regulárnímu výrazu. Duplicity generované způsobené samotným regulárním výrazem (např. `(OMO|OMO)`) řešit nemusíte.