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.

1. Dopište implementaci třídy `Type` a vytvořte objekty reprezentující následující kus kódu:

   ```
   interface A { A f(); }
   interface B extends A {}
   interface C extends A { C f(); }
   ```

2. Vyřešte zpřístupnění vnitřní reprezentace ve vaší implementaci třídy `Type`.

3. Napište iterátor, který postupně vrátí všechny nadtypy (i tranzitivní) daného typu. Pokud je nějaký nadtyp dosažitelný přes několik různých cest, můžete (ale nemusíte) jej vrátit víckrát. Metodu `remove` neimplementujte; implementace, která nejprve zkopíruje všechny nadtypy do jedné kolekce a poté vrátí její iterátor není přípustná. Tip: použijte frontu.

4. Napište adaptér třídy `Type` pojmenovaný `TypeAdapter`, který má metody `getAdapterSuperTypes` a `getMethods` a který v metodě `getMethods` vrací i metody zděděné z nadtypů. Překrytí metod detekujte podle jména, přetěžování metod neuvažujte.

5. Třídu `Type` změňte na abstraktní, vytvořte dvě její potřídy `Interface` a `Class` a upravte metodu pro přidávání nadtypů tak, abyste zabránili vzniku v Javě nepřípustných struktur (tzn. aby třída byla nadtyp rozhraní nebo aby třída měla jako nadtyp dvě a více jiných tříd). Pro zjišťování typu přidávaného nadtypu použijte dvojitý dispatch.

6. Napište metodu `Type Type.buildTypes(java.lang.Class c)`, která k danému class souboru vytvoří odpovídající strukturu typů. Při budování uvažujte jen typy a metody anotované anotací `@show`.