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. Třída `Engine` představuje extrémně jednoduchý grafický engine, který při každém taktu voláním `MovingObject.moveYourself()` vyzve všechny evidované objekty, aby změnily svoji polohu. Dopište kód třídy `Bumblebee`, která reprezentuje objekt vodorovně kroužící v zadané vzdálenosti okolo zadaného středu. Jednu otáčku proveďte za 360 taktů.

2. Instance třídy `Guard` se nejprve stokrát pohnou o vektor `direction` a poté stokrát opačně. Dopište jejich kód, při implementaci využijte návrhový vzor stav.

3. Do `Engine.mainLoop` dopište detekci blízkosti dvou objektů: pokud se dva objekty přiblíží na vzdálenost menší, než `proximityRadius`, zavolejte na nich `movedClose`, pokud se naopak vzdálí, zavolejte na nich `movedAway`. Obě metody volejte pouze při změně stavu, tzn. nevolejte je, pokud dva objekty byly v předchozím taktu blízko/daleko a v tomto taktu stále zůstávají blízko/daleko.

4. Napište implementaci dekorátoru `SpeedDampener`, který sníží rychlost dekorovaného objektu tím, že mu přepošle jen každé druhé volání `moveYourself`.

5. Dopište metodu `Engine.iterator(double maxDistance)`, která vrátí všechny evidované objekty, jejichž vzdálenost od bodu $(0, 0, 0)$ není větší než `maxDistance`. Nesmíte použít iterátor již existující kolekce, můžete předpokládat, že během iterování se objekty nepohnou. Metodu `remove` implementovat nemusíte.

6. Do iterátoru z předchozího příkladu (a třídy `Engine`) dopište detekci pohnutí — pokud se objekty od vytvoření iterátoru pohnuly, metody `hasNext` a `next` vyhodí výjimku.