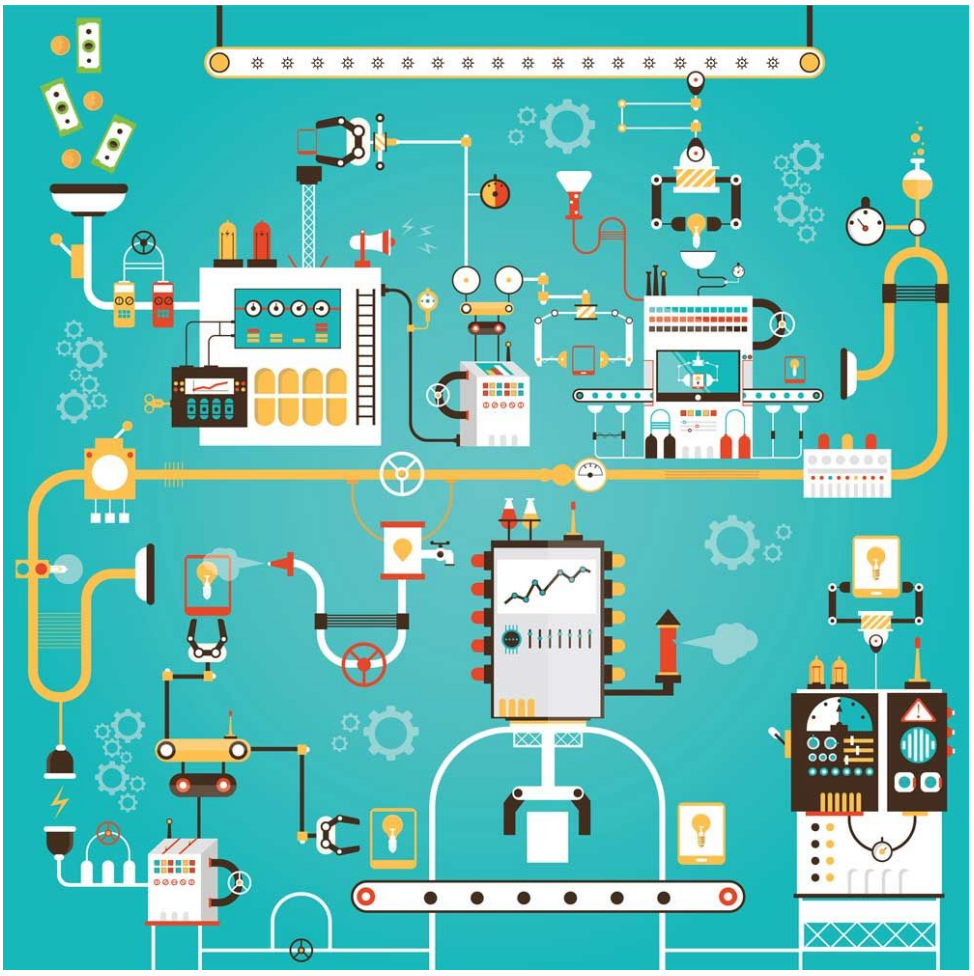

Projekt: Smart Factory

Abstrakt

Vytvořit aplikaci pro virtuální simulaci inteligentní továrny, kde simulujeme chod výroby - na výrobních linkách s pomocí strojů a lidí vyrábíme produkty. Používáme jednotlivé stroje a vyhodnocujeme jejich využití, spotřebu a kvalitu výsledných výrobků. Součástí výrobního procesu jsou nejen stroje a lidé, ale i kolaborativní roboty. Základní časová jednotka je jeden takt (trvá jednu hodinu). Stavy továrny se mění (a vyhodnocují) po těchto taktech.

Funkční požadavky

F1. Hlavní entity se kterými pracujeme je továrna, linka (s prioritou), stroj, člověk a výrobek, materiál plus libovolné další entity. Stroje, lidé i výrobky mohou být různého druhu.

F2. Produkty se vyrábějí v sériích po několika stech kusů, jestliže se mění série nekompatibilních výrobků, tak je potřeba výrobní linky přeskládat. Každý výrobek má definovanou sekvenci zařízení, robotů, lidí, které je potřeba za sebou uspořádat na linku.

F3. Stroje a roboty mají svoji spotřebu; lidé, roboty, stroje i materiál stojí náklady.

F4. Komunikace mezi stroji, roboty a lidmi probíhá pomocí eventů. Event může dostat 1 až N entit (člověk, stroj, robot), které jsou na daný druh eventu zaregistrované. Eventy je potřeba odbavit.
F5. Jednotlivá zařízení mají API na sběr dat o tomto zařízení. O zařízeních sbíráme data jako je spotřeba elektřiny, oleje, materiálu a funkčnost (opotřebení roste s časem).
F6. Stroje a roboty se po určité době rozbijí. Po rozbití vygenerují event (alert) s prioritou podle důležitosti linky, který odbaví člověk - opravář.
F7. Opravářů je omezený počet. Oprava trvá několik taktů. Při začátku opravy a konci opravy je generován event (bude se hodit pro požadavek F10 :-). Vznikají situace, kdy nejsou dostupní žádní opraváři - pak se čeká dokud se některý z nich neuvolní. Po uvolnění opravář nastupuje na 1. nejprioritnější, 2. nejstarší defekt.
F8. Návštěva ředitele a inspektora. Realizujeme návštěvu továrny, kdy ředitel prochází továrnou přesně podle stromové hierarchie entit *továrna ->* linka -> *(stroj|robot|člověk nebo výrobek)* a inspektor prochází podle míry opotřebení. Ředitel i inspektor mají na sobě definované akce, které provedou s daným typem entity. Zapište sekvenci procházení a názvy provedených akcí do logu.
F9. Za továrnu je nutné vygenerovat následující reporty za **libovolné časové období**:
    - *FactoryConfigurationReport*: veškerá konfigurační data továrny zachovávající hierarchii - továrna ->* linka -> *(stroj|robot|člověk nebo výrobek).
    - *EventReport*: jaké za jednotlivé období vznikly eventy, kde je grupujeme 1) podle typu eventu, (2) podle typu zdroje eventu a (3) podle toho, kdo je odbavil.
    - *ConsumptionReport*: Kolik jednotlivé zařízení, roboty spotřebovaly elektřiny, oleje, materiálu. Včetně finančního vyčíslení. V reportu musí být i summární spotřeby za nadřazané entity (linka|továrna)
    - *OuttagesReport*: Nejdelší výpadek, nejkratší výpadek, průměrná doba výpadku, průměrná doba čekání na opraváře a typy zdrojů výpadků setříděné podle délky výpadku.
F10. Vraťte stavy strojů v zadaném taktu (jiném než posledním :-)). Stav zrekonstruujte z počátečního stavu a sekvence eventů, které byly na stroji provedeny.

Nefunkční požadavky

- Není požadována autentizace ani autorizace
- Aplikace může běžet pouze v jedné JVM
- Aplikaci pište tak, aby byly dobře schované metody a proměnné, které nemají být dostupné ostatním třídám. Vygenerovný javadoc by měl mít co nejméně public metod a proměnných.
- Reporty jsou generovány do textového souboru
- Konfigurace továrny může být nahrávána přímo z třídy nebo externího souboru (preferován je json)

Vhodné design patterny

- State machine
- Iterator
- Factory/Factory method

- Decorator/Composite
- Singleton
- Visitor/Observer/Listener
- Chain of responsibility
- Partially persistent data structure
- Object Pool
- Lazy Initialization

Požadované výstupy

- Design ve formě class diagramů a stručného popisu jak chcete úlohu realizovat
- Veřejné API - Javadoc vygenerovaný pro funkce, kterými uživatel pracuje s vaším software
- Alespoň 20% tříd veřejného API pokryto unit testy
- Dvě různé konfigurace továrny, alespoň 30 entit.

Hodnocení

- Čistota softwarového designu aplikace a veřejného API (40%)
- Množství naimplementovaných funkčních požadavků (50%)
- Vlastní inovace - nový funkční požadavek, který jste přidali nebo zajímavý softwarový design (10%)

Logistika

- Úloha se vypracovává ve dvou studentech (ze stejného cvičení), přičemž je požadováno, aby každá osoba napsala přibližně stejně komplexní část aplikace a rozuměla i částem, které nepsala (bude kontrolováno v gitu). Semestrální úloha může být znovu otevřena u zkoušky. Skupiny po jednom a třech jsou výjimkou.
- Odevzdání je do konce semestru. Ideálně před vánoci, maximálně konec semestru.
- Minimálně tři týdny před finálním odevzdáním je potřeba cvičícímu ukázat design vaší aplikace. Pokud by byl design špatný, došlo k odchýlení od zadání nebo naopak budete chtít poradit, tak bude konzultace provedena na cvičení, případně mimo cvičení po dohodě s cvičícím.