

1. feladat: Rövid kérdések

15 pont

1.1 Egy újonnan fejlesztett adatstruktúra teljesítményét mérjük. Az alkalmazást Java nyelven fejlesztettük, a teljesítményt JMH segítségével mérjük. Egy speciális konstruktor futás idejére vagyunk kíváncsiak. Észrevettük, hogy az első futás után az idő szinte nullára csökken (pontosabban 0,6 nanoszekundumra). Mi ennek a magyarázata, és hogyan lehet ezt javítani?

```
@Benchmark public void measureRuntime() { new MyDataStructure(12, false, 1); }
```

1.2 Programunk teljesítményét öt egyre nehezebb probléma méretén értékeljük ki. Az eredményeket egy boxplot segítségével szeretnénk megjeleníteni. Rajzoljon egy példát egy olyan **boxplotra**, ahol az öt mérési pont közül a medián futási idő és a szórás is növekszik a probléma méretével!

1.3 Mutasson egy példát egy **vezérlési folyam diagramra** és néhány **tesztbemenetre**, ahol a feltétel lefedettség 100% (condition coverage), de az MC/DC lefedettség nem!

2. feladat: Gráf alapú modellezés

25 pont

a) Készítsen egy **Refinery metamodellt** az alábbi specifikáció alapján:

Egy olyan informatikai rendszert szeretnénk modellezni, amely ókori szövegek elemzését kezeli mesterséges intelligencia által működtetett eszközökkel. Az ókori kéziratokat és azok digitális elemzési folyamatát követjük nyomon. Minden kézirat 1-50 szövegrészből áll, és pontosan egy ókori írásrendszerben íródott. Az ókori írásrendszerek lehetnek képirások, szótagírások vagy betűírások. Minden szövegrész MI-modelleken keresztül kerül elemzésre, amelyek értelmezési eredményeket állítanak elő. Egy MI-modell vagy karakterfelismerésre vagy szemantikai elemzésre specializálódik. A karakterfelismerő modellek több ókori írásrendszert is felismerhetnek, míg a szemantikai elemző modellek pontosan egy ókori írásrendszert elemeznek. Az értelmezési eredményeket emberi szakértők vagy jóváhagyhatják, vagy vitathatják. A szakértőknek legalább egy kutatóintézetrel kell affiliációban állniuk.

Kizárólag Refinery kódot írjon! NE írjon Java kódot vagy rajzoljon UML osztálydiagrammot.

Használja az alábbi fogalmakat: **affiliations** (affiliációi), **AIModel** (MI-modell), **AlphabeticScript** (betűírás), **analyzes** (elemzi), **AncientScript** (ókori írásrendszer), **CharacterRecognizer** (karakterfelismerő), **disputedBy** (vitatója), **HumanExpert** (emberi szakértő), **InterpretationResult** (elemzési eredmény), **interpretedRegion** (elemzett szövegrész), **LogographicScript** (képírás), **Manuscript** (kézirat), **recognizes** (felismeri), **ResearchInstitution** (kutatóintézet), **results** (eredményei), **SemanticAnalyzer** (szemantikai elemző), **SyllabicScript** (szótagírás), **TextRegion** (szövegrész), **textRegions** (szövegrészei), **validatedBy** (jóváhagyója), **writtenIn** (rajta íródott)

b) Rajzoljon **gráfmodellt** az alábbi adatok alapján:

A hieroglif és a démotikus írásrendszerek az ókori Egyiptom képirásai. Az Ani papiruszt hieroglif írással írták és három szövegrészt tartalmaz. A kutatók két MI-modellt alkalmaztak: a HieroNet-et (egy karakterfelismerőt, amely képes mind a hieroglif, mind a démotikus írást felismerni) és az EgyptBERT-et (egy hieroglif szövegekre specializálódott szemantikai elemzőt). Az első szövegrészt mindkét modell elemezte, két értelmezési eredményt létrehozva. Dr. Sarah Jones az Oxford Egyetemről jóváhagyta a karakterfelismerési eredményt, míg Prof. Ahmed Hassan a Kairói Egyetemről vitatta a szemantikai elemzés eredményét.

Kizárólag gráfmodellt rajzoljon! NE írjon Java vagy Refinery kódot.

c) Egy szakértő szerint a következő **jólformáltsági kényszer** érvényes: *egy szövegrészt egy MI-modell csak akkor elemezhet, ha képes kezelni a megfelelő írásrendszert.* A szakértő azonban gyanítja, hogy ezt a metamodell jelenlegi verziójával nem lehet kikényszeríteni. Ennek megerősítéshez rajzoljon egy olyan **gráfmodellt**, amely megfelel a metamodellnek, de megsérti a kényszert!

d) A szakértő szerint egy kézirat szövegrészei akár különböző írásrendszerekben is íródhatnak. **Javasoljon** egy módszert a metamodel módosítására, hogy az ilyen kéziratokat is kezelhessük.

3. feladat: Szöveges modellezés

30 pont

Egy **szöveges szakterületspecifikus nyelvet** szeretnénk készíteni a maja naptár időegységeinek és eseményeinek a leírásához. Az alábbi példa a kívánt **konkrét szintaxist** (szöveges leírást) és **absztrakt szintaxist** (példánygráfot) mutatja be a nyelvhez:

```
timeunit Kin;  
timeunit Winal = 20 Kin;  
timeunit Tun = 18 Winal;  
timeunit Katun = 20 Tun;  
timeunit Baktun = 20 Katun;
```

```
event "Dresden p74" (  
  9 Baktun,  
  0 Katun,  
  16 Tun  
);
```

```
event "Tikal Alt 1" (  
  8 Baktun,  
  18 Katun,  
  10 Tun,  
  17 Winal,  
  0 Kin  
);
```

a) Készítsen egy **Langium nyelvtant** a nyelv elemzésére! Az alábbi deklarációk már rendelkezésre állnak:

grammar Chronology

```
hidden terminal WS: /\s+/  
terminal ID: /[_a-zA-Z][\w_]*/;  
terminal INT: /\d+/  
terminal STRING: /"[^"]*"//;
```

Adja meg a nyelvtan hiányzó részét!

b) Készítsen **Jinja2 sablont** az időegységek közötti átváltást megvalósító C függvénykönyvtár generálásához! A sablon bemenete az a Calendar objektum, amelyet az a) részben létrehozott nyelvtani elemzővel olvastunk be. Egy példa C függvénykönyvtár az alábbiakban látható:

```
long convert_Winal(long value) { return 20 * value; }  
long convert_Tun(long value) { return 18 * convert_Winal(value); }  
long convert_Katun(long value) { return 20 * convert_Tun(value); }  
long convert_Baktun(long value) { return 20 * convert_Katun(value); }
```

A könnyebb olvashatóság kedvéért a közvetlenül a példánymodellből származó szövegeket **fékővérrel** emeltük ki. Ügyeljen arra, hogy minden származtatott időegységhez egy függvényt generáljon. Ne generáljon függvényt az alap időegységekhez.

Az példánymodellben a `type` attribútum tartalmazza az objektumok típusát (pl. a `x.type == "BaseUnit"` használatával ellenőrizheti, hogy az `x` `BaseUnit` típusú-e). A kereszthivatkozások sztringként vannak tárolva, amelyek értékei megegyeznek a hivatkozott objektum nevével.

