Automated Software Engineering Exam 2 (January 16, 2025) Group A

Task 1: Short questions 15 points

1.1 We are measuring the performance of a newly developed data structure. We developed the data structure in Java, and we are using JMH for the measurements. We want to evaluate the runtime of a special constructor. We notice that after the first run, the measured runtime drops to almost 0 (an extremely low value of 0.6 nanoseconds, to be precise). How can you explain this situation and how can we fix it?

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

1.2 We evaluated the performance of our program on five increasingly difficult problem sizes, and we want to visualize the results using boxplots. Draw an example **boxplot** with five measurement problem sizes where the median runtime and the standard deviation both grow with the problem size.

1.3 Create an example **control flow diagram** and corresponding **test inputs** where the condition coverage is 100% but the MC/DC coverage is not.

Task 2: Graph modeling 25 points

a) Write a **Refinery metamodel** based on the following specification:

We want to model a system for managing ancient text analysis using AI-powered tools. In this domain, we track ancient manuscripts and their digital analysis process. Each manuscript consists of 1 to 50 text regions and is written in exactly one ancient script. Ancient scripts may be logographic, syllabic, or alphabetic. Each text region undergoes analysis through multiple AI models, producing interpretation results. An AI model specializes in either character recognition or semantic analysis. Character recognition models may recognize multiple ancient scripts, while semantic analysis models analyze exactly one ancient script. Interpretation results can be either validated or disputed by human experts, who must be affiliated with at least one research institution.

Only provide Refinery code and do NOT write Java code or draw an UML class diagram.

Use the following concept names: `affiliations, AIModel, AlphabeticScript, analyzes, AncientScript, CharacterRecognizer, disputedBy, HumanExpert, InterpretationResult, interpretedRegion, LogographicScript, Manuscript, recognizes, ResearchInstitution, results, SemanticAnalyzer, SyllabicScript, TextRegion, textRegions, validatedBy, writtenIn`

b) Draw a **graph model** based on the following data:

Hieroglyphic and Demotic are logographic scripts from ancient Egypt. The Papyrus of Ani is written in hieroglyphic and contains three text regions. Researchers have involved two AI models: HieroNet (a character recognizer that can recognize both hieroglyphic and Demotic) and EgyptBERT (a semantic analyzer specialized in hieroglyphic texts). The first text region has been analyzed by both models, producing two interpretation results. Dr. Sarah Jones from Oxford University validated the character recognition result, while Prof. Ahmed Hassan from Cairo University disputed the semantic analysis result.

Only provide a graph model and do NOT write Java or Refinery code.

c) According to a domain expert, the following **constraint** holds: *an AI model analyzing a text region must be able to handle the corresponding script.* However, the expert suspects that this cannot be enforced by the current version of the metamodel. To confirm, draw a **graph model** that conforms to the metamodel but violates the constraint.

d) The domain expert also noted that some manuscripts contain text regions written in different scripts. **Recommend** a way to update the metamodel for representing such manuscripts.

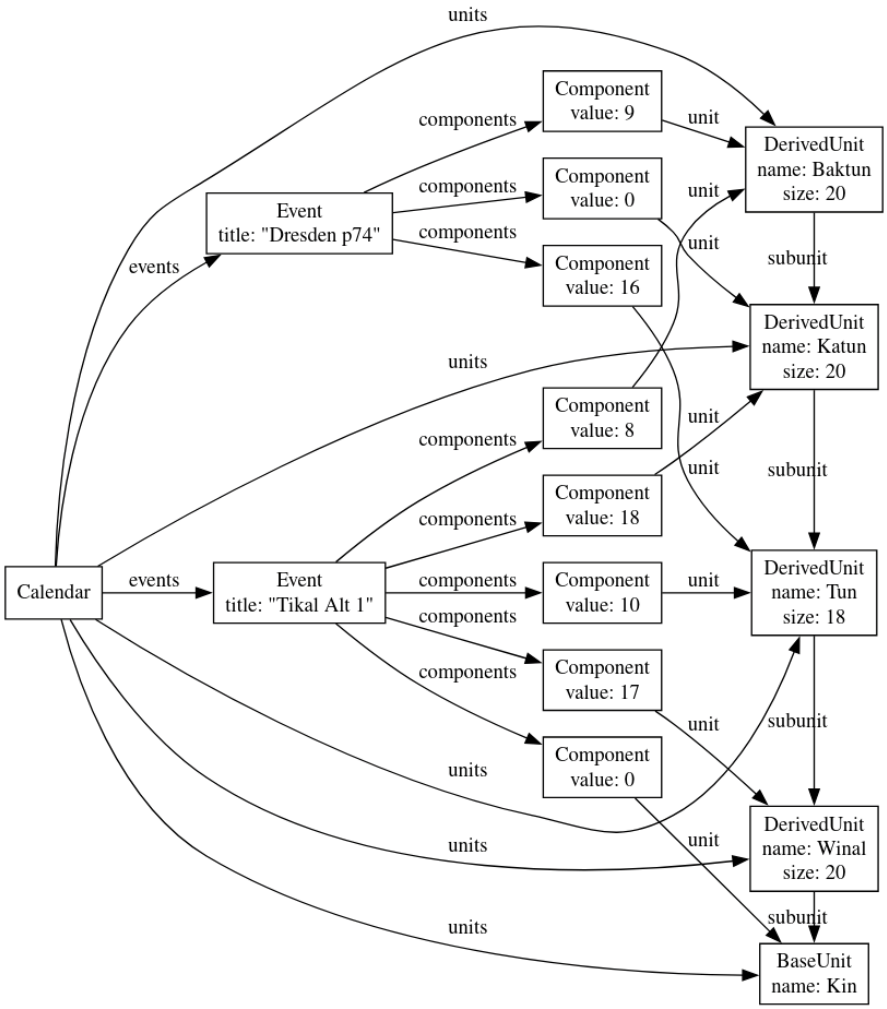
Task 3: Textual modeling — 30 points

We would like to create a **textual domain-specific modeling language** to describe the time units and events in the Mayan calendar. The example below shows the desired **concrete syntax** (textual description) **and abstract syntax** (instance graph model) for the language:

```
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) Create a **Langium grammar** to parse this language! The following declarations are available to you:

```
grammar Chronology

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

Provide the rest of the grammar.

b) Create a **Jinja2 template** to generate a unit converter C library. The input of the template is the `Calendar` object parsed by the grammar you created in part a). An example C library is shown below:

```
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); }
```

To help you in reading the example, we emphasized text coming directly from instance model in **bold**. Make sure to output a conversion function for each derived unit. Do not output a function for any base unit.

In the instance model, the `type` attribute contains the type of each object (e.g., use `x.type == "BaseUnit"` to check if `x` is of type `BaseUnit`). Cross-references are encoded as strings equal to the `name` of the referenced object.