Prompt Templates:

We want to respond in a certain format.
Instead of prompt strings, Langchain4J uses parameterized prompt strings.

Applications for templates:
Teachers creating lesson plans or curricula
Automated customer service or support
Document summarization for a given audience
Blog posts (generate a basic blog for a website where only certain information changes)
Marketing content

```
String myTemplate = "Please explain {{topic}} to a {{student_type}} using a clear, succinct paragraph";
PromptTemplate promptTemplate = PromptTemplate.from(myTemplate);
```

Dynamically we assign values to PromptTemplate

Advantages of PromptTemplates:
Development efficiency because we use predefined prompt structures
Consistent structure reduces typos and errors
Scalability improves as prompt templates keep structure clear and consistent with low complexity
Testability, you can test prompt templates more easily by changing the parameter values. It makes testing easier
Prompt templates can reduce bias with standardization, which is a repeating deviation from an expected response from an LLM

Disadvantages of PromptTemplates:
Prompt injection (bad actors can inject additional nefarious text into prompt)
Systems breach: details of systems infrastructure in prompt and this information can be revealed to wrong audience
Privacy breach: prompt may contain private information
Adversarial attack: confuses the LLMs to cause harm
Additional processing: more complexity, context size issues. String management can be costly. Especially in chatbot conversation, we got to monitor the growing size of the prompt.

Tips for prompt templates
Keep prompt templates concise and clear
Use parameters only for dynamic content
Store prompt templates separately in a template repo for maintenance purposes
Version templates for maintainability
Monitor prompt templates for regular refinement
Encrypt prompt strings before sending into any network public or private