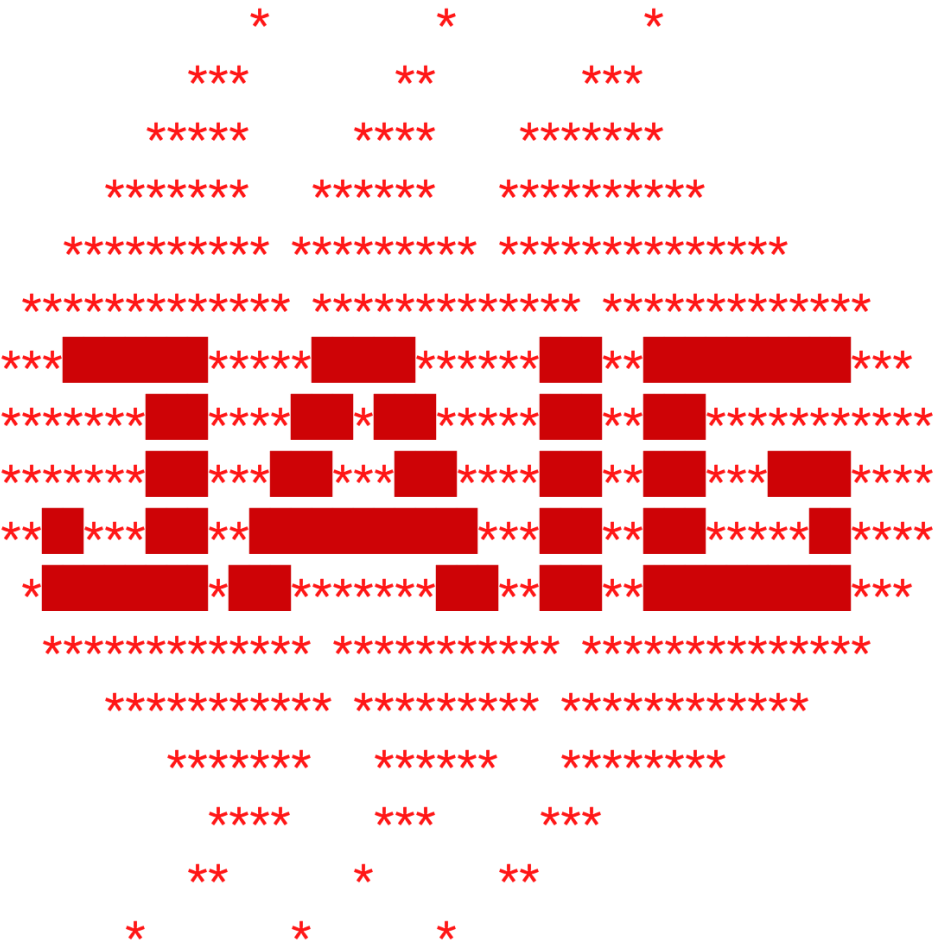
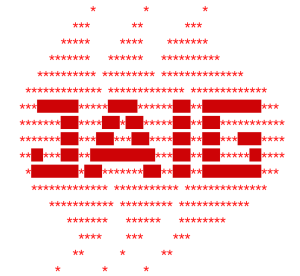


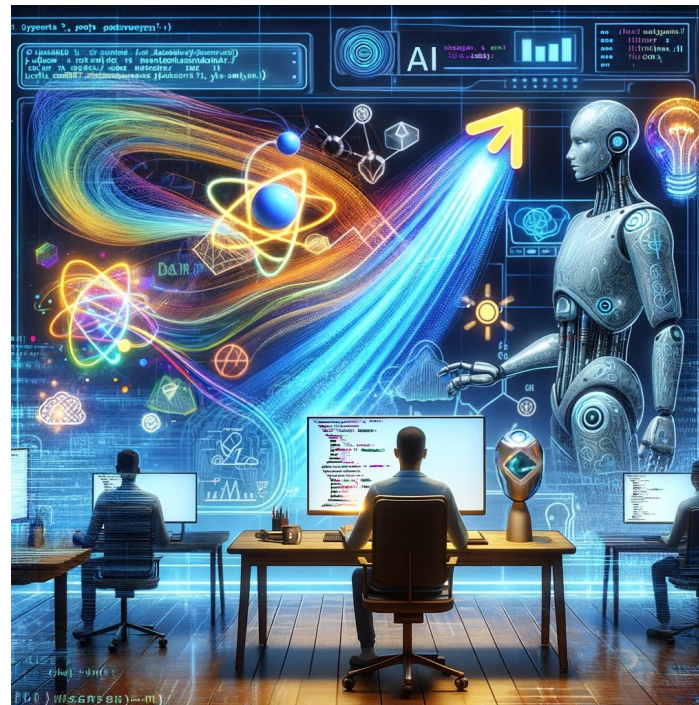
# Java AI-powered Code Generator



# Why we need JAIG?



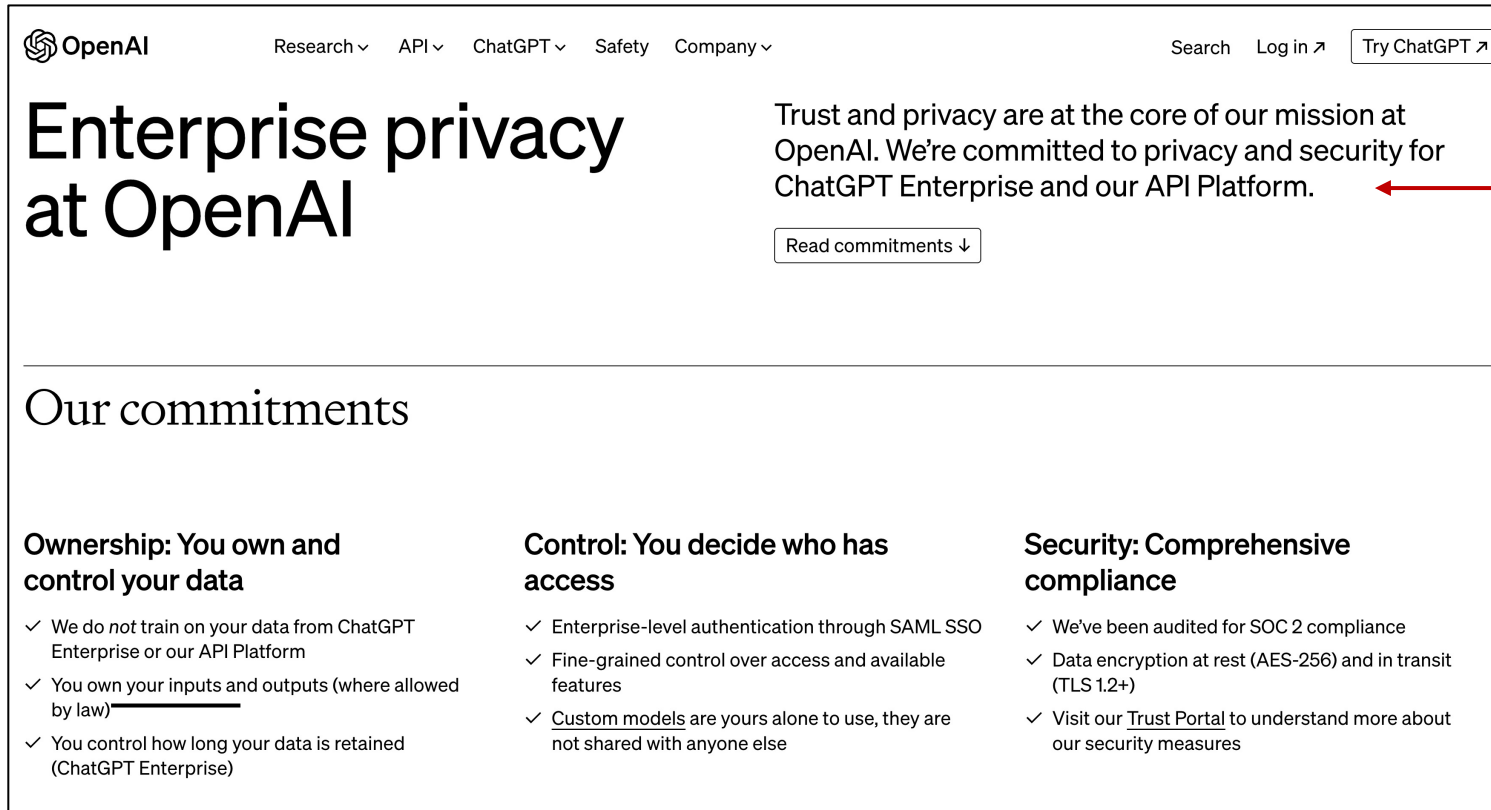
**JAIG** will **improve** the **efficiency** of senior developers (2-3x times)  
and **replace** junior/middle developers with **code generation**



# Privacy for your data and generated code

Chat GPT                      => insecure  
Open AI **API**                => **secure**

<https://openai.com/enterprise-privacy>



The screenshot shows the OpenAI website's enterprise privacy page. At the top, there's a navigation bar with links for Research, API, ChatGPT, Safety, and Company. A search bar and links for Log in and Try ChatGPT are also present. The main heading is "Enterprise privacy at OpenAI". Below this, a paragraph states: "Trust and privacy are at the core of our mission at OpenAI. We're committed to privacy and security for ChatGPT Enterprise and our API Platform." A button labeled "Read commitments" is positioned below this text. The section "Our commitments" follows, divided into three columns: Ownership, Control, and Security, each with a list of bullet points detailing their privacy and security measures.

OpenAI    Research ▾    API ▾    ChatGPT ▾    Safety    Company ▾    Search    Log in ↗    Try ChatGPT ↗

## Enterprise privacy at OpenAI

Trust and privacy are at the core of our mission at OpenAI. We're committed to privacy and security for ChatGPT Enterprise and our API Platform.

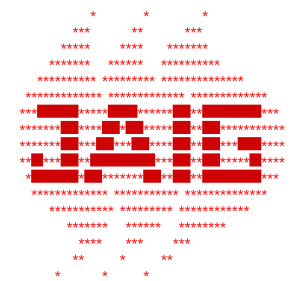
[Read commitments ↓](#)

### Our commitments

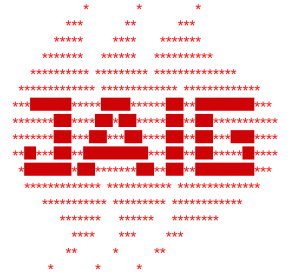
<b>Ownership: You own and control your data</b> <ul style="list-style-type: none"><li>✓ We do <i>not</i> train on your data from ChatGPT Enterprise or our API Platform</li><li>✓ You own your <u>inputs and outputs</u> (where allowed by law)</li><li>✓ You control how long your data is retained (ChatGPT Enterprise)</li></ul>	<b>Control: You decide who has access</b> <ul style="list-style-type: none"><li>✓ Enterprise-level authentication through SAML SSO</li><li>✓ Fine-grained control over access and available features</li><li>✓ <u>Custom models</u> are yours alone to use, they are not shared with anyone else</li></ul>	<b>Security: Comprehensive compliance</b> <ul style="list-style-type: none"><li>✓ We've been audited for SOC 2 compliance</li><li>✓ Data encryption at rest (AES-256) and in transit (TLS 1.2+)</li><li>✓ Visit our <a href="#">Trust Portal</a> to understand more about our security measures</li></ul>
---	--	---

Using API  
is secure!

Using  
Chat GPT  
is NOT



# Generating code in IDE



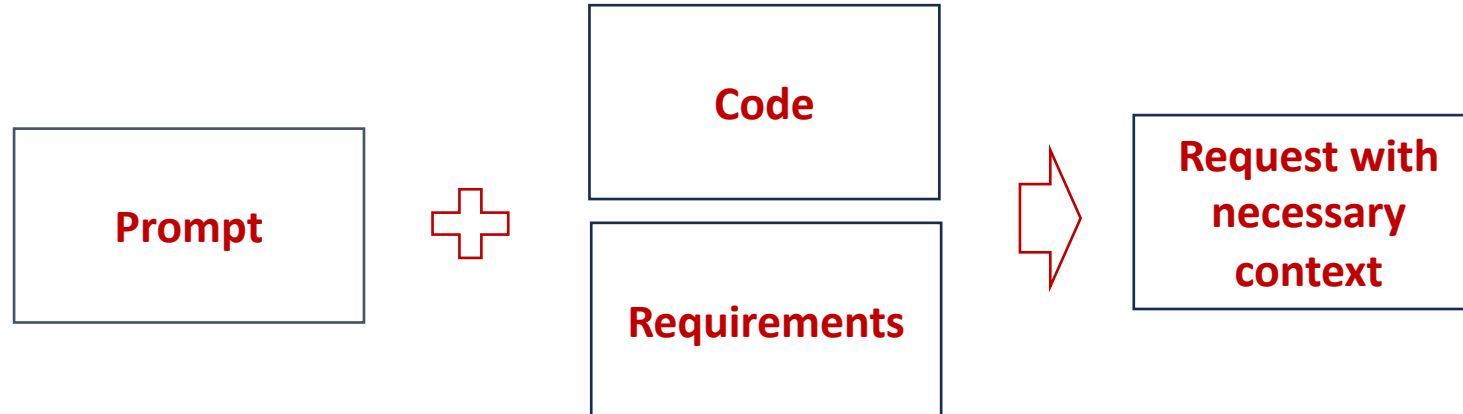
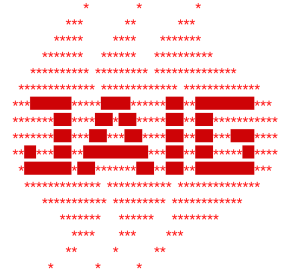
**Prompt**



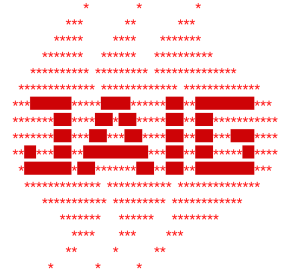
**Response**

# Feeding the necessary context to LLM

When you use ChatGPT, you need to copy-paste the necessary context manually.  
With JAIG, add the path to the necessary files.



# Feeding the necessary context to LLM



When you use ChatGPT, you need to copy-paste the necessary context manually.  
With JAIG, add the path to the necessary files.

prompt.txt

```
/src/java/main/App.java  
/docs/requirements.txt
```

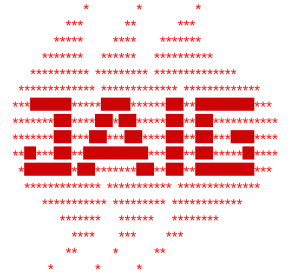
```
Update App to satisfy the requirements
```

← **Files to be included in the prompts**

← **Prompt**

As a result, request.txt is generated with the contents of App.java and requirements.txt included.

# Feeding the necessary context to LLM



When you use ChatGPT, you need to copy-paste the necessary context manually.  
With JAIG, add the path to the necessary files.

prompt.txt

```
/src/main/**  
/docs/**
```

```
Update App to satisfy the requirements
```

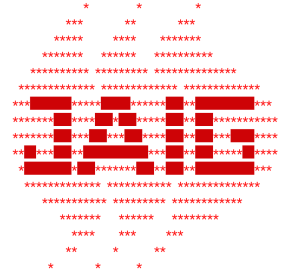
**Include all source files**  
**Include all documents**

**Prompt**

**We can easily include many files to the request if necessary with wildcards.**

Caution: including more context makes the response less precise and more expensive.

# Feeding the necessary context to LLM



When you use ChatGPT, you need to copy-paste the necessary context manually.  
With JAIG, add the path to the necessary files.

prompt.txt

```
/src/main/**
```

```
../01_uml/uml-response.txt
```

```
Update App to satisfy the requirements
```

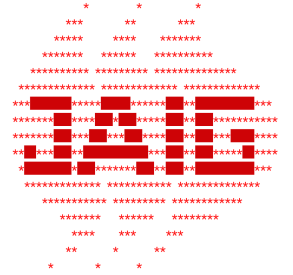
← **Relative path**

← **Prompt**

**We can include the relative paths**



# Automatic parsing of the generated code



If generated code includes package names, it is parsed automatically and placed to the folders.

To make it working, ask LLM to add package names. Example:

***Use `app.jtutor.shapes` as a package name before each class.***

```
package app.jtutor.shapes;

public class Shape { ... }

package app.jtutor.shapes;

public class Line extends Shape { ... }

package app.jtutor.shapes;

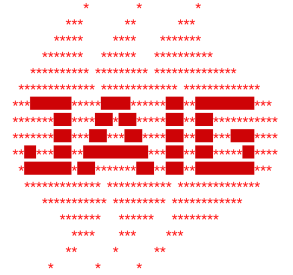
public class Rectangle extends Shape { ... }
```

**When package name is specified,  
JAIG is able to automatically  
split the response and put it to  
the folders**

response

# Automatic parsing of the generated code

If generated code includes package names, it is parsed automatically and placed to the folders.



```
singleton.txt  JAIG.yaml  code.txt  prompt.txt
1  ../01_uml/uml-response.txt
2
3  Create Java application from the UML model.
4  Use inheritance.
5
6  Use app.jtutor.shapes as a package name before each class.
7
8  Add getters and setters.
9
10 Implement method draw() with simple printout (print all properties).
11
```

# Merge code and GPT response

We can have  
**2 types of changes:**

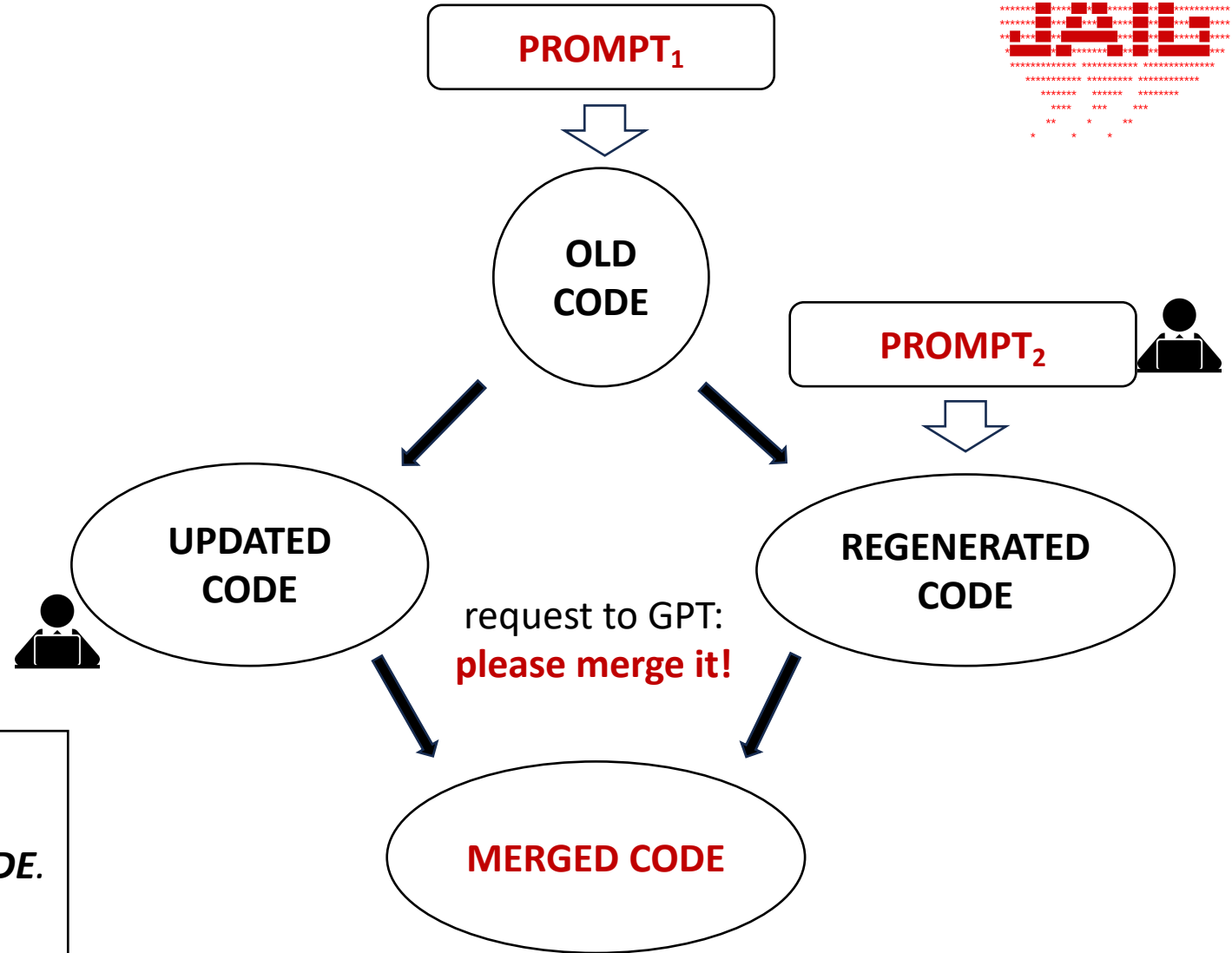
- In the **code**
- In the **prompt**

If you have **both**, you need to  
**merge** them

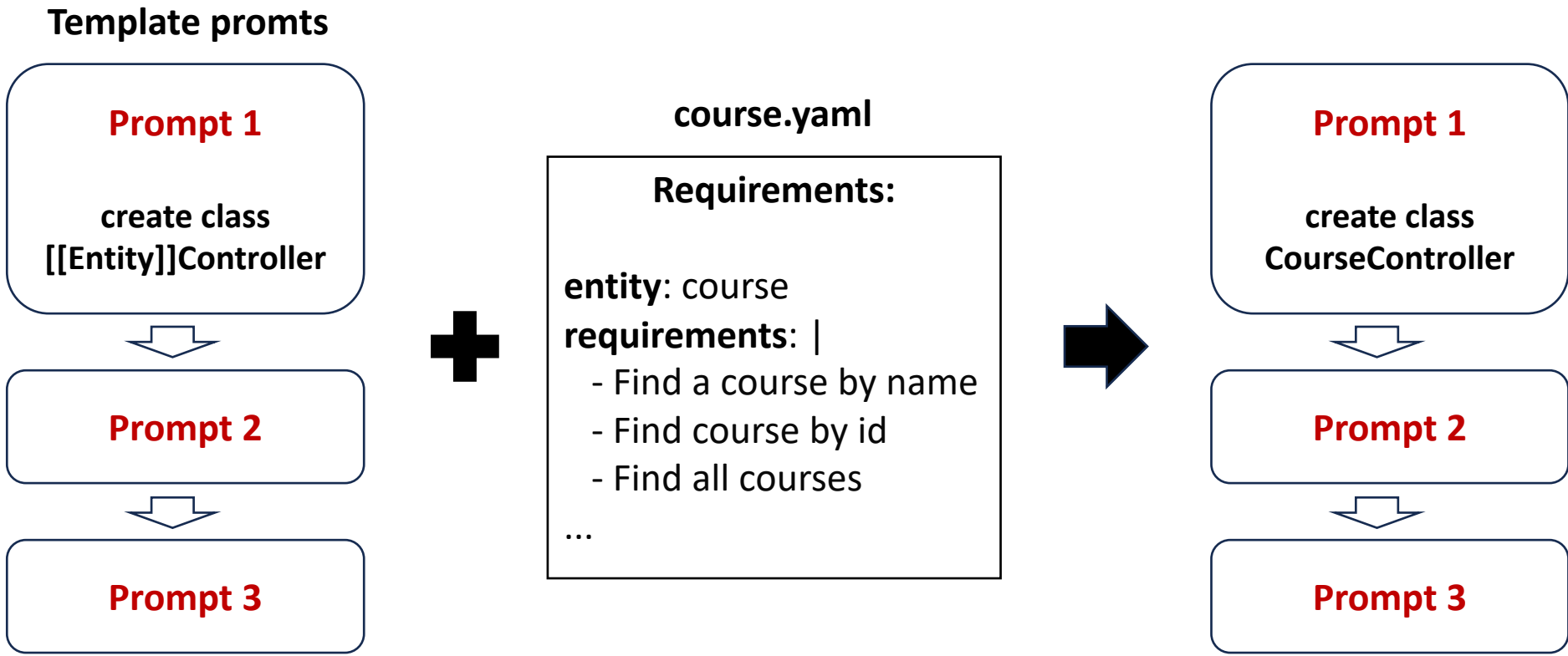
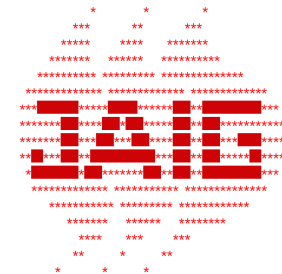
**Merging allows to separately  
change code and requirements.**

**merge prompt** be like:

*Dear Chat GPT,  
I had **OLD CODE**.  
Then programmer changed it to **UPDATED CODE**.  
And you changed it to **REGENERATED CODE**.  
Can you please **merge all** these changes?*



# JAIG Templates



entity: course

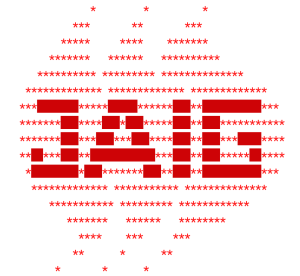
Entity: Course

entities: courses

Entities: Courses

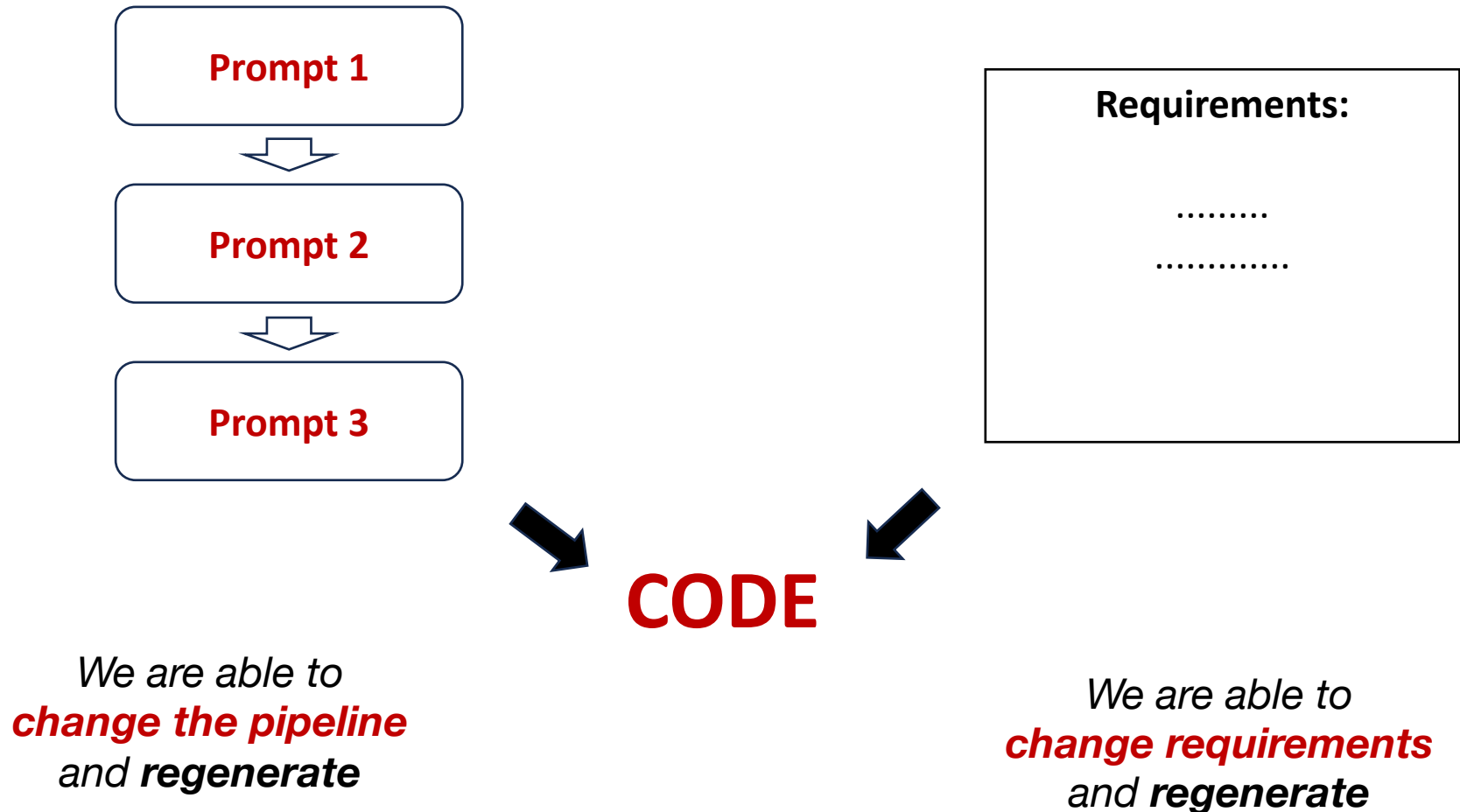
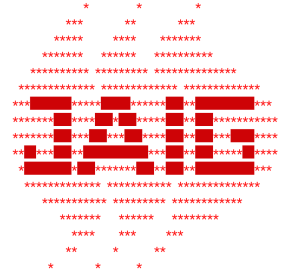
# Major JAIG features:

- **Privacy** for your data and generated code
- Feeding the **necessary context** to LLM
- Automatic **parsing** of generated code
- Automatic code **merge**
- Prompt **Templates**
- Prompt **Pipelines**
- Code **Refactoring** support

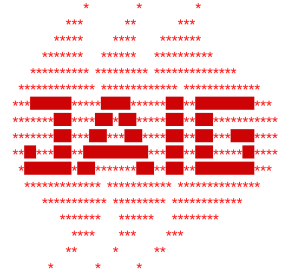


# Development cycle with JAIG:

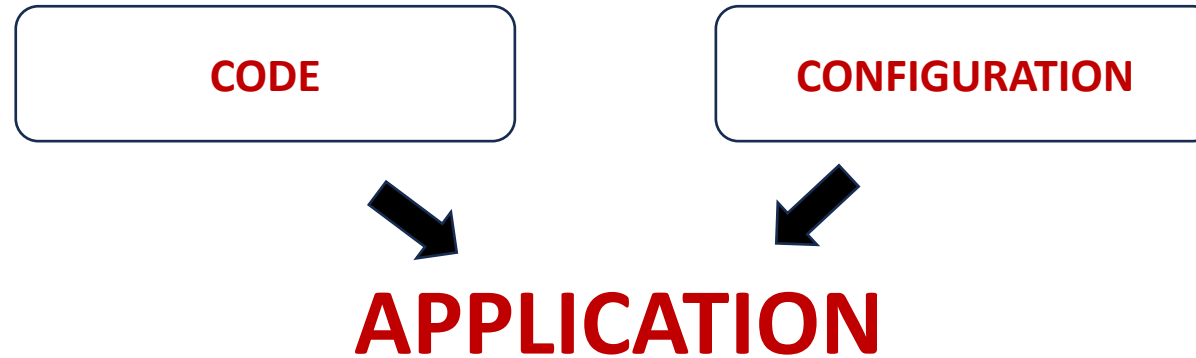
- Create a universal **prompts pipeline** of application generation
- Apply any **business domain** to generate a standardized code



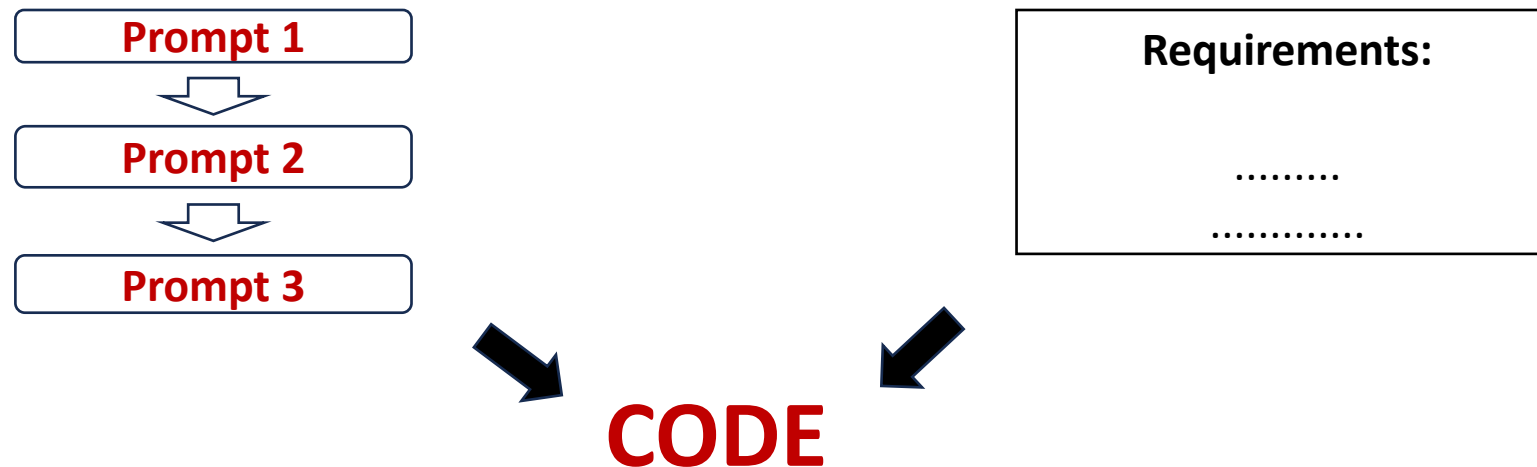
# Spring vs JAIG



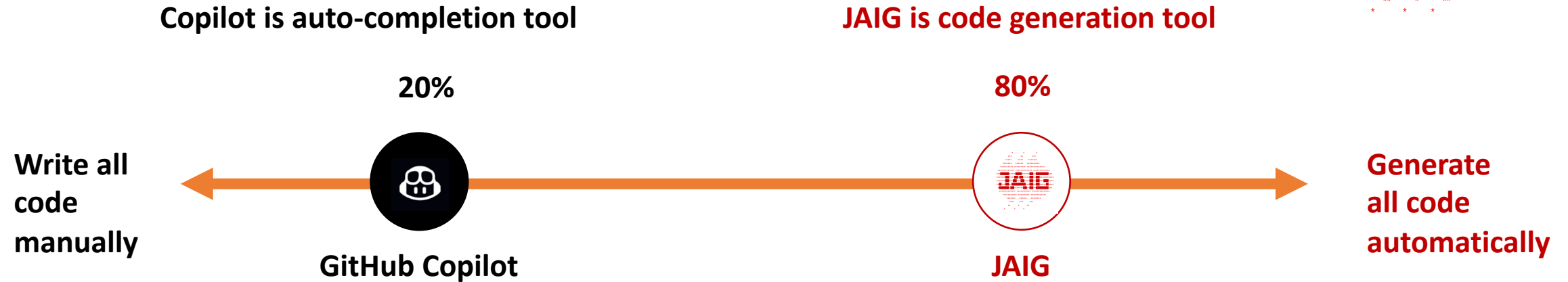
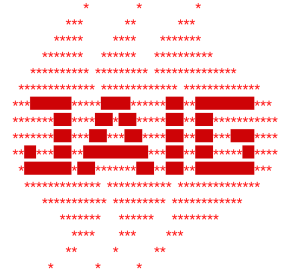
Spring separates **code** from **configuration**



JAIG separates **prompt pipeline** to implement code (specific for your project) from **requirements**



# Microsoft Copilot and JAIG



**JAIG** will **improve** the **development efficiency** of senior developers (2-3x times)  
and **replace** junior/middle developers with **code generation**