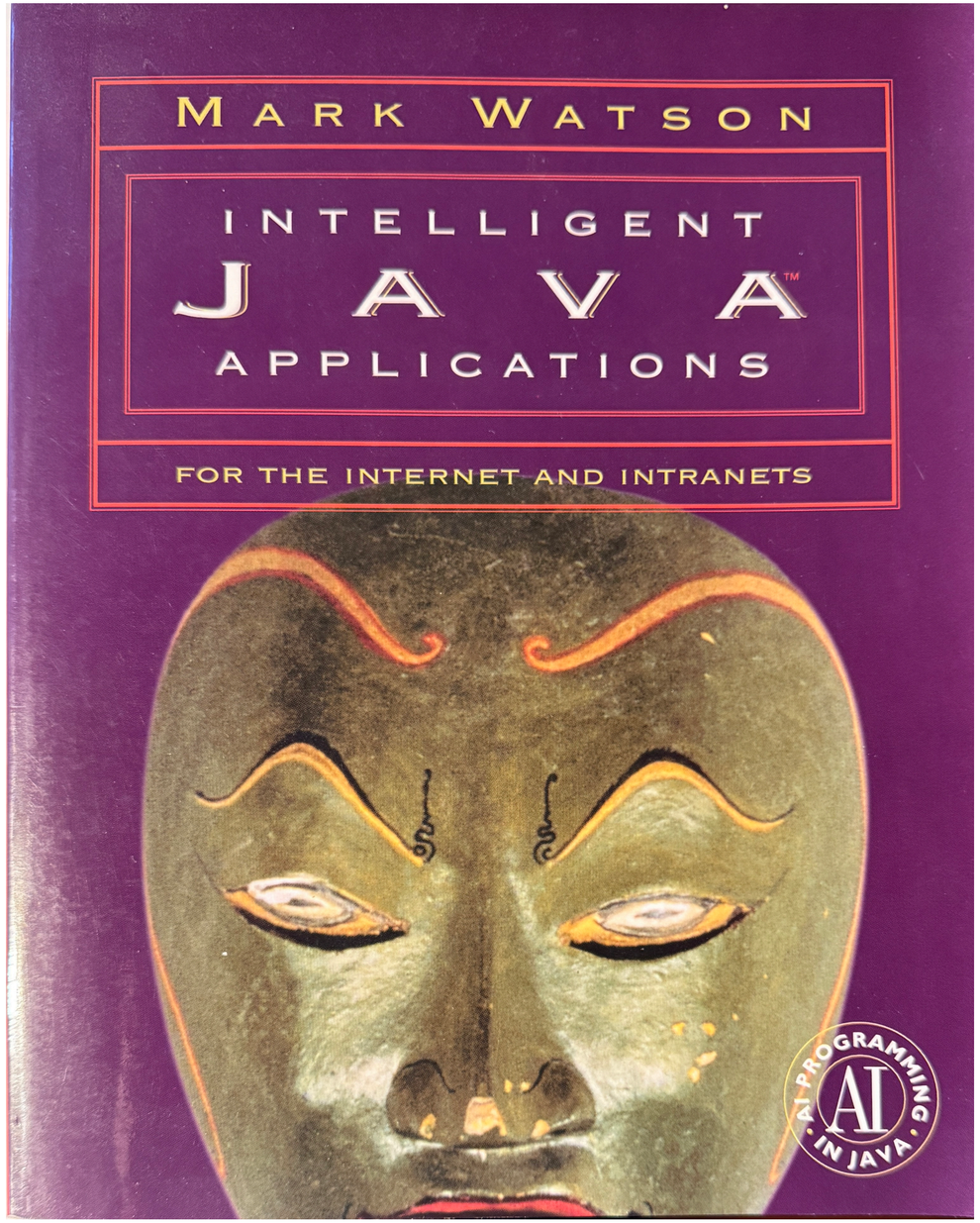
Pure Java LLM Patterns, Tricks and Hacks

🚀…for the Enterprise


MARK WATSON
INTELLIGENT
JAVA™
APPLICATIONS
FOR THE INTERNET AND INTRANETS
AI PROGRAMMING IN JAVA
AI

(1997)

airhacks.industries

airhacks.live

"It's not work if you like it"
…so I never worked. #java

youtube.com/

@bienadam

youtube.com/bienadam/shorts

airhacks.fm

#296 Exploring ONNX, Embedding Models, and Retrieval Augmented Generation (RAG) with Langchain4j

[episode link]

An airhacks.fm

Dmytro pre
for running
handling, e
longer text
generation
and data so
powered ap
embedding
embedding
outputs usi
specialized

Dmytro Liubars

#313 Revolutionizing AI with Java: From LLMs to Vector APIs

[episode link] Listen on Ap

An airhacks.fm c

Alfonso previ
projects for r
multiplication
qwen or gem
for Java Com
integrating w
optimizations
but limited w
improvement
in enterprise

#315 The AI Revolution in Java Development and Devoxx Genie

[episode link] Listen on Apple Podcasts LISTEN ON Spotify Listen on Google Podcasts [RSS]

An airhacks.fm conversation with Stephan Janssen (@Stephan007) about:

Stephan previously appeared on "#254 How JavaPolis and Devoxx Happened", discussion on the AI re
development of an AI-assisted photo sharing application, creation of the Devoxx Genie IntelliJ plugin f
advantages of Claude 3.5 from Anthropic, use of local AI models in development environments, integra
langchain4j and its adoption by Red Hat, development of Java-based AI tools like Lama3.java, jlama an
AI models in software development, challenges and opportunities for junior and senior developers in A
importance of understanding cloud services and cost structures when using AI, potential future of pro
generation, discussion on maintaining and improving AI-generated code, exciting developments in Ja
and tornadovm, potential for running AI models directly on Java without external dependencies, consi
and integration, the need for promoting Java's capabilities in AI development, potential for Visual Stud

airhacks.live

airhacks.TV

with the time machine, “100 episodes ago segment”

…any questions left?

airhacks.live

NEW https://discord.gg/airhacks

airhacks.live

NEW online, live virtual workshops

Continuous coding, explaining, interacting and sharing with Adam Bien

Live, Virtual Online Workshops, Summer 2025:

- LLM / GenAI Patterns, Architectures and Use Cases with Java, 10 July 2025
- Hardcore Serverless Java on AWS, 17 July 2025

Tickets are also available from: airhacks.eventbrite.com and meetup.com/airhacks

by Adam Bien

You don't like live, interactive virtual workshops? Checkout video courses: airhacks.io

Java + Clouds = ?

Total					
	Energy		Time		Mb
(c) C	1.00	(c) C	1.00	(c) Pascal	1.00
(c) Rust	1.03	(c) Rust	1.04	(c) Go	1.05
(c) C++	1.34	(c) C++	1.56	(c) C	1.17
(c) Ada	1.70	(c) Ada	1.85	(c) Fortran	1.24
(v) Java	1.98	(v) Java	1.89	(c) C++	1.34
(c) Pascal	2.14	(c) Chapel	2.14	(c) Ada	1.47
(c) Chapel	2.18	(c) Go	2.83	(c) Rust	1.54
(v) Lisp	2.27	(c) Pascal	3.02	(v) Lisp	1.92
(c) Ocaml	2.40	(c) Ocaml	3.09	(c) Haskell	2.45
(c) Fortran	2.52	(v) C#	3.14	(i) PHP	2.57
(c) Swift	2.79	(v) Lisp	3.40	(c) Swift	2.71
(c) Haskell	3.10	(c) Haskell	3.55	(i) Python	2.80
(v) C#	3.14	(c) Swift	4.20	(c) Ocaml	2.82
(c) Go	3.23	(c) Fortran	4.20	(v) C#	2.85
(i) Dart	3.83	(v) F#	6.30	(i) Hack	3.34
(v) F#	4.13	(i) JavaScript	6.52	(v) Racket	3.52
(i) JavaScript	4.45	(i) Dart	6.67	(i) Ruby	3.97
(v) Racket	7.91	(v) Racket	11.27	(c) Chapel	4.00
(i) TypeScript	21.50	(i) Hack	26.99	(v) F#	4.25
(i) Hack	24.02	(i) PHP	27.64	(i) JavaScript	4.59
(i) PHP	29.30	(v) Erlang	36.71	(i) TypeScript	4.69
(v) Erlang	42.23	(i) Jruby	43.44	(v) Java	6.01
(i) Lua	45.98	(i) TypeScript	46.20	(i) Perl	6.62
(i) Jruby	46.54	(i) Ruby	59.34	(i) Lua	6.72
(i) Ruby	69.91	(i) Perl	65.79	(v) Erlang	7.20
(i) Python	75.88	(i) Python	71.90	(i) Dart	8.64
(i) Perl	79.58	(i) Lua	82.91	(i) Jruby	19.84

GraalVM™

reduces RAM footprint

https://sites.google.com/view/energy-efficiency-languages/results?authuser=0

Total

	Energy
(c) C	1.00
(c) Rust	1.03
(c) C++	1.34
(c) Ada	1.70
(v) Java	1.98
(c) Pascal	2.14
(c) Chapel	2.18
(v) Lisp	2.27
(c) Ocaml	2.4
(c) Fortran	2
(c) Swift	9
(c) Haskell	0
(v) C#	
(c)	3.
(i) Dart	3.83
F#	2.13
ava	4.45
ack	7.91
(i) peScript	21.50
ack	24.02
(i) PHP	29.30
(v) Erlang	42.23
(i) Lua	45.98
(i) Jruby	46.54
(i) Ruby	6
(i) Python	.88
(i) Perl	9.5

	T
(c) C	1.
(c) Rust	1.0
(c) C++	1.56
Ada	5
Java	
(c) Chape	2
(c)	2.
(c) l	2
(c) Oc	3.09
C#	3.14
Lisp	3.40
askell	3.55
(c) Swift	4.20
(c) Fortran	4.20
(v) F#	6.30
(i) JavaScript	6.52
(i) Dart	6.67
(v) Racket	1.27
(i) Hack	9
(i) PHP	27
(v) Er g	36.71
(i) Jr y	43.44
T ipt	6.20
Ru	9.34
erl	65.79
(i) thon	71.90
(i)	82.91

	Mb
Pasca	1.00
o	1.05
(c)	1.17
(c) an	1.24
C++	1.34
(c) Ada	1.47
(c) Rust	1.54
(v) Lisp	1.92
(c) Haskell	2.45
(i) PHP	2.57
(c) Swift	
(i) Python	2.80
(c) Ocaml	2.82
(v)	85
ack	4
Ra	2
(i) y	3.97
(c) pel	4.00
(v)	4.25
(i) JavaScript	4.59
(i) TypeScript	4.69
(v) Java	6.01
(i) Perl	6.62
(i) Lua	6.72
(v) Erlang	7.20
(i) Dart	8.64
(i) Jruby	19.84

partially relevant

GraalVM™

reduces RAM footprint

https://sites.g e.com/ rgy-efficiency-languages/results?authuser=0

airhacks.live

serverless facts

The calculations below exclude free tier discounts.

Architecture

Arm

Number of requests

2

Unit

per second

Duration of each request (in ms)

Duration is calculated from the time your code begins executing until it returns or otherwise terminates.

100

Amount of memory allocated

Enter the amount between 128 MB and 10 GB

Value

2

Unit

GB

Amount of ephemeral storage allocated

Enter the amount between 512 MB and 10,240 MB. The first 512 MB are at no additional charge, you only pay for any additional storage that you configure for the function.

Value

512

Unit

MB

Show calculations

Total Upfront cost: 0.00 USD
Total Monthly cost: 15.07 USD

Show Details

Cancel

Save and view summary

Save and add servi

m-bien

The calculations below exclude free tier discounts.

Architecture

Arm

Number of requests

2

Unit

per second

Duration of each request (in ms)

Duration is calculated from the time your code begins executing until it returns or otherwise terminates.

10000

Amount of memory allocated

Enter the amount between 128 MB and 10 GB

Value

2

Unit

GB

Amount of ephemeral storage allocated

Enter the amount between 512 MB and 10,240 MB. The first 512 MB are at no additional charge, you only pay for any additional storage that you configure for the function.

Value

512

Unit

MB

▶ Show calculations

Total Upfront cost: 0.00 USD
Total Monthly cost: 1,402.66 USD

Show Details ▼

Cancel

Save and view summary

Save and add servi

dam-bien

The calculations below exclude free tier discounts.

Architecture

Arm

Number of requests

2

Unit

per second

Duration of each request (in ms)

Duration is calculated from the time your code begins executing until it returns or otherwise terminates.

10000

Amount of memory allocated

Enter the amount between 128 MB and 10 GB

Value

128

Unit

MB

Amount of ephemeral storage allocated

Enter the amount between 512 MB and 10,240 MB. The first 512 MB are at no additional charge, you only pay for any additional storage that you configure for the function.

Value

512

Unit

MB

Show calculations

Total Upfront cost: 0.00 USD
Total Monthly cost: 88.65 USD

Show Details

Cancel

Save and view summary

Save and add servi

am-bien

Who cares about “pure Java”?

Multithreading Support

One of the advantage of Deep Java Library (DJL) is Multi-threaded inference support. It can help to increase the throughput of your inference on multi-core CPUs and GPUs and reduce memory consumption compared to Python.

http://djl.ai/docs/development/inference_performance_optimization.html

Performance

Before DJL, running predictions with this model and such high-dimensional data used to take around 24 hours and had multiple memory issues. DJL reduced the prediction time on this model by **85%**, from around one day to a couple of hours. DJL worked out of the box without spending any time on engineering tasks, such as memory tuning. In contrast, prior to DJL, we spent more than two weeks in memory tuning.

More about DJL

Deep Java Library (DJL) is an open source library to build and deploy deep learning in Java. This project launched in December 2019 and is widely used among teams at Amazon. This effort was inspired by other DL frameworks, but was developed from the ground up to better suit Java development practices. DJL is framework agnostic, with support for Apache MXNet, PyTorch, TensorFlow 2.x (experimental), and fastText (experimental). Additionally, DJL offers a repository of pre-trained models in our ModelZoo that simplifies implementation and streamlines model sharing across projects.

https://aws.amazon.com/blogs/opensource/how-amazon-retail-systems-run-machine-learning-predictions-with-apache-spark-using-deep-java-library/

What is “Enterprise”?

enterprise LLMs / EAI

- Traceability / Explainability
- Cost Effectiveness
- Testability
- Security / Privacy
- Repeatability / Maintainability
- Robustness

additional challenges

- AI gateways
- Web Application Firewalls
- throttling / rate limiting
- legacy APIs / polling
- proprietary APIs
- authentication
- authorization

LLM / AI

remote LLM / AI

- HTTP API
- API-key authentication / RBAC
- JSON communication
- asynchronous streaming
- synchronous “RPC”
- …just another, unreliable and slow, microservice

unusual nature

- idempotent (mostly)
- stateless API
- stateful communication
- slow (high latency)
- low availability
- defined scalability
- robust
- IO bound

Anthropic Claude

```java
public interface Claude {

    enum Models{
        CLAUDE_35_SONNET("claude-3-5-sonnet-20240620"),
        CLAUDE_30_OPUS("claude-3-opus-20240229");
        private String modelName;
    }

    HttpClient client = HttpClient.newHttpClient();
    URI uri = URI.create("https://api.anthropic.com/v1/messages");
    String ANTHROPIC_VERSION = "2023-06-01";
    String API_KEY = "duke";
    Models currentModel = Models.CLAUDE_35_SONNET;


    public static JSONObject invoke(String system, String user, float temperature){
        var enclosedPrompt = messagePrompt(user);
        var payloadJSON = Claude.claudeMessage(enclosedPrompt,temperature,system);
        payloadJSON.put("model", currentModel.modelName());
        var payload = payloadJSON.toString();
        var answer = invoke(payload);
        return new JSONObject(answer);
    }

    static JSONArray messagePrompt(String user) {
        return new JSONArray()
                .put(message("user", user));

    }

    static JSONObject message(String role, String content) {
        return new JSONObject()
                .put("role", role)
                .put("content", content);
    }


    static JSONObject claudeMessage(JSONArray messages, float temperature, String system) {
        return new JSONObject()
                .put("max_tokens", MAX_TOKENS)
                .put("messages", messages)
                .put("temperature", temperature)
                .put("system", system);

    }

    static String invoke(String message) {
        var request = HttpRequest.newBuilder(uri)
                .POST(BodyPublishers.ofString(message))
                .header("x-api-key", API_KEY)
                .header("content-type", "application/json")
                .header("anthropic-version", ANTHROPIC_VERSION)
                .build();
        try {
            var response =  client.send(request, BodyHandlers.ofString());
            var body = response.body();
            if(response.statusCode() == 529){
                Log.error("claude is overloaded, please try again later " + body);
            }
            return body;
        } catch (IOException | InterruptedException e) {        }
    }
}
```

Claude via AWS Bedrock

```java
public interface Bedrock {
        String BEDROCK_VERSION = "bedrock-2023-05-31";
        enum Model {
                CLAUDE_V21("anthropic.claude-v2:1", Region.EU_CENTRAL_1),
                CLAUDE_V3_SONNET("anthropic.claude-3-sonnet-20240229-v1:0", Region.US_EAST_1);
        }

        Model model = Model.CLAUDE_V3_SONNET;
        AwsCredentials credentials = AwsBasicCredentials.create("chief","duke");

        static String getPrompt(String system, String user) {
                if (system == null || system.isEmpty()) {
                        return "\n\nHuman: %s\n\nAssistant:".formatted(user);
                } else {
                        return """
                                        %s
                                        \n\nHuman: %s\n\nAssistant:
                                        """.formatted(system, user);
                }
        }
        public static JSONObject invokeClaude(String user, float temperature) {
                return invokeClaude(null, user, temperature);
        }

        public static JSONObject invokeClaude(String system, String user, float temperature) {
                var client = bedrockRuntimeClient();
                var enclosedPrompt = Claude.messagePrompt(user);
                var payloadJSON = Claude.claudeMessage(enclosedPrompt,temperature,system);
                payloadJSON.put("anthropic_version", BEDROCK_VERSION);
                var payload = payloadJSON.toString();
                var request = InvokeModelRequest.builder()
                                .body(SdkBytes.fromUtf8String(payload))
                                .modelId(model.modelId())
                                .overrideConfiguration(AwsRequestOverrideConfiguration.builder()
                                                .apiCallTimeout(Duration.ofMinutes(5))
                                                .build())
                                .contentType("application/json")
                                .accept("application/json")
                                .build();
                var response = client.invokeModel(request);
                var message = response.body().asUtf8String();
                return new JSONObject(message);
        }

        private static BedrockRuntimeClient bedrockRuntimeClient() {
                var client = BedrockRuntimeClient
                                .builder()
                                .region(model.region())
                                .httpClientBuilder(UrlConnectionHttpClient
                                                .builder()
                                                .socketTimeout(Duration.ofMinutes(5))
                                                .connectionTimeout(Duration.ofMinutes(5)))
                                .overrideConfiguration(ClientOverrideConfiguration
                                                .builder()
                                                .apiCallTimeout(Duration.ofMinutes(5))
                                                .build())
                                .credentialsProvider(StaticCredentialsProvider.create(credentials))
                                .build();
                return client;
        }
}
```

quarkus / langchain4j

```
@RegisterAiService
public interface TriageService {

    @SystemMessage("""
            You are working for a bank. You are an AI processing reviews about financial products
            You will always answer with a JSON document, and only this JSON document.
            """)
    @UserMessage("""
            Your task is to process the review delimited by ---.
            Apply a sentiment analysis to the passed review to determine if it is positive or neg
            The review can be in any language. So, you will need to identify the language.

            For example:
            - "I love your bank, you are the best!", this is a 'POSITIVE' review
            - "J'adore votre banque", this is a 'POSITIVE' review
            - "I hate your bank, you are the worst!", this is a 'NEGATIVE' review

             Answer with a JSON document containing:
            - the 'evaluation' key set to 'POSITIVE' if the review is positive, 'NEGATIVE' otherw
            - the 'message' key set to a message thanking the customer in the case of a positive

            ---
            {review}
            ---
            """)
    TriagedReview triage(String review);
```

local / offline LLM / AI

- Docker e.g. Ollama
- ONNX / GGUF direct integration
- JNI / Vector API / Panama
- HTTP API to localhost
- memory and CPU and GPU hungry
- behaves like another legacy microservice (e.g. PL1 / COBOL integration)
- unstable
- high latency

Java LLM Edition (21)😀

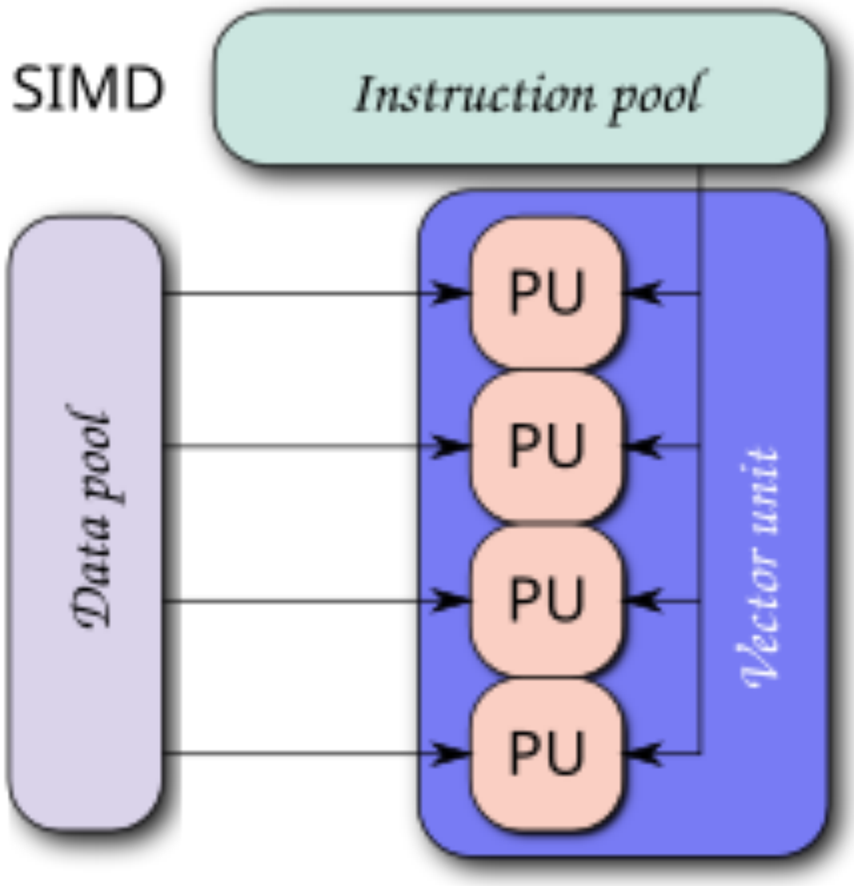
Half Floats

```
static void main(String... args) {
  short halved = Float.floatToFloat16(1f);
  System.out.println(halved);
  var full = Float.float16ToFloat(halved);
  System.out.println(full);
}
```

Regular Multiplication

```
static void scalarComputation(float[] a, float[] b, float[] c) {
  for (int i = 0; i < a.length; i++) {
    c[i] = (a[i] * a[i] + b[i] * b[i]) * -1.0f;
  }
}
```

SIMD



https://en.wikipedia.org/wiki/Single_instruction,_multiple_data

Java Vector API

- (Clear and concise API)
- Platform agnostic
- Reliable runtime compilation and performance on x64 and AArch64:
  - Streaming SIMD Extensions
  - Advanced Vector Extensions (AVX)
  - ARM Aarch64: NEON and SVE

Java Vector API

- Graceful degradation
- Alignment with Project Valhalla

Java Vector API

- “efficient vector computations in Java”
- vector operations optimally compiled to vector instructions on CPU architectures
- in Java 23 is the eight preview. Vector API is available in Java 21 as preview
- Vector<E>, VectorSpecies<E>, VectorMask<E>, VectorShuffle<E>
- matrix vector multiplication is crucial in a neural networks

Vector API / SIMD

```
VectorSpecies<Float> SPECIES = FloatVector.SPECIES_PREFERRED;

static void vectorComputation(float[] a, float[] b, float[] c) {
  int upperBound = SPECIES.loopBound(a.length);
  for (int i = 0; i < upperBound; i += SPECIES.length()) {
    var va = FloatVector.fromArray(SPECIES, a, i);
    var vb = FloatVector.fromArray(SPECIES, b, i);
    var vc = va.mul(va)
        .add(vb.mul(vb))
        .neg();
    vc.intoArray(c, i);
  }
}
```

DeepNetts

- full stack Java
- netbeans based IDE
- build ML models with Java
- creation and distribution of neural network models
- reference implementation for JSR-381 (Visual Recognition (VisRec) Specification): https://github.com/JavaVisRec/visrec-api/wiki/Getting-Started-Guide

https://www.deepnetts.com/

Deep Java Library (DJL.ai)

- Engine-agnostic design allows code to run on multiple deep learning frameworks like MXNet, PyTorch, and TensorFlow without code changes
- Feels like a regular Java library with native development experience
- Supports multithreading and works naturally with Java
- automatic CPU/GPU detections and configuration
- special memory collector (NDManager) prevents memory leaks
- efficient handling of multidimensional arrays through NDArray class
- supports model training and inference

LLAMA

LLAMA — the new standard

- open-source model with licensing restrictions
- developed by Meta (Facebook)
- NLP tasks
- base for many fine-tuned models

https://llama.meta.com/

GGUF (GPT-Generated Unified Format)

- a file format for AI models
- designed for efficient inference
- successor to GGML format
- introduced by llama.cpp project
- quickly adopted by open-source AI projects
- used by Mistral
- A self-contained parser for Java is available: Llama3.java

GGUF (GPT-Generated Unified Format)

- a file format for AI models
- designed for efficient inference
- successor to GGML format
- introduced by llama.cpp project
- quickly adopted by open-source AI projects
- used by Mistral
- A self-contained parser for Java is available: Llama3.java

Java LLM Edition (21++)🚀

Java 21++

- Code Reflection
- Babylon
- Project Valhalla
- TornadoVM

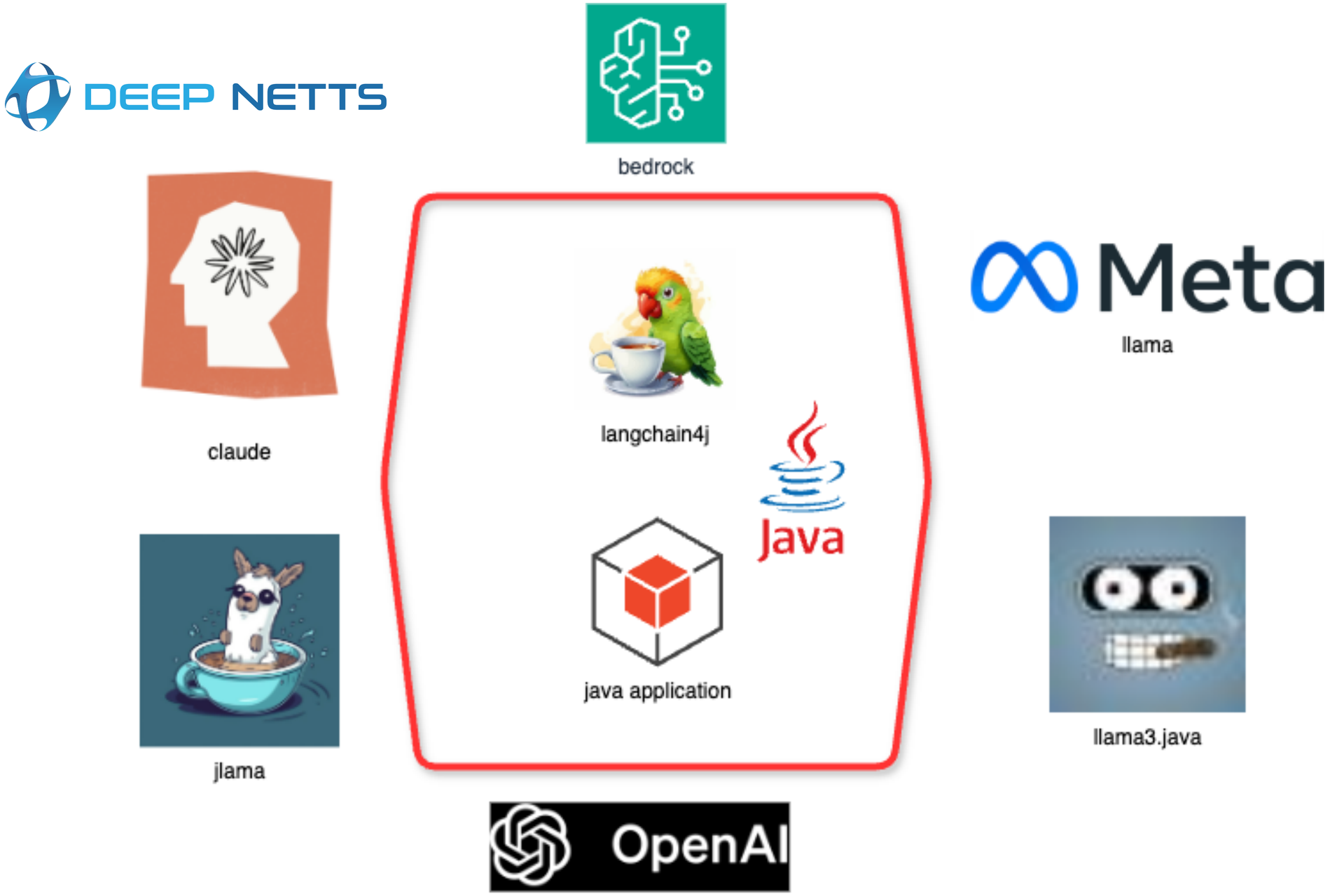
EAI (Enterprise Application Integration 😀) 2.0?

offline challenges

- native access
- CI/CD, IaC, provisioning
- concurrency
- garbage collection, memory management
- provisioning, Infrastructure as Code
- hardware costs

What about langchain4j?

Hexagonal Architectures


DEEP NETTS
bedrock
claude
langchain4j
Meta
llama
Java
java application
jlama
llama3.java
OpenAI
ChatGPT

The Fat Enterprise “AI” JAR
👉FAIAR🚀

LLM👉just another unstable service

```
using claude model: claude-3-5-sonnet-20240620
claude is overloaded, please try again later {"type":"error","error":{"type":"overloaded_error","message":"Overloaded"}}
content key not found in: {"type":"error","error":{"type":"overloaded_error","message":"Overloaded"}}
```

Fault Tolerance

The Remedy

@Timeout

@Retry

@Fallback

@CircuitBreaker

@Bulkhead

@Asynchronous

…and interceptors

The Future is Now

AI Architectures

AI Architectures

Tools, Functions, RAG

Code Interpreter

MCP

Agents

Agent Swarms

Agentic Workflows

Autonomous Agents

The Future

airhacks.live

NEW online, live virtual workshops

Continuous coding, explaining, interacting and sharing with Adam Bien

Live, Virtual Online Workshops, Summer 2025:

- LLM / GenAI Patterns, Architectures and Use Cases with Java, 10 July 2025
- Hardcore Serverless Java on AWS, 17 July 2025

Tickets are also available from: airhacks.eventbrite.com and meetup.com/airhacks

by Adam Bien

You don't like live, interactive virtual workshops? Checkout video courses: airhacks.io

Thank YOU!

airhacks.industries