

JAPAN | JUNE 20, 2024

aws SUMMIT



# コードから生産性を高める、 AWS 生成 AI サービスの開発者向け活用例

アマゾンウェブサービスジャパン合同会社



# 背景



生成 AI のユースケースが提供する価値の

**75%** は次の4つの分野で生まれる:

- カスタマーオペレーション
- マーケティング & 営業
- **ソフトウェア開発**
- **プロダクト R&D**



# こんな困り事ってありませんか？

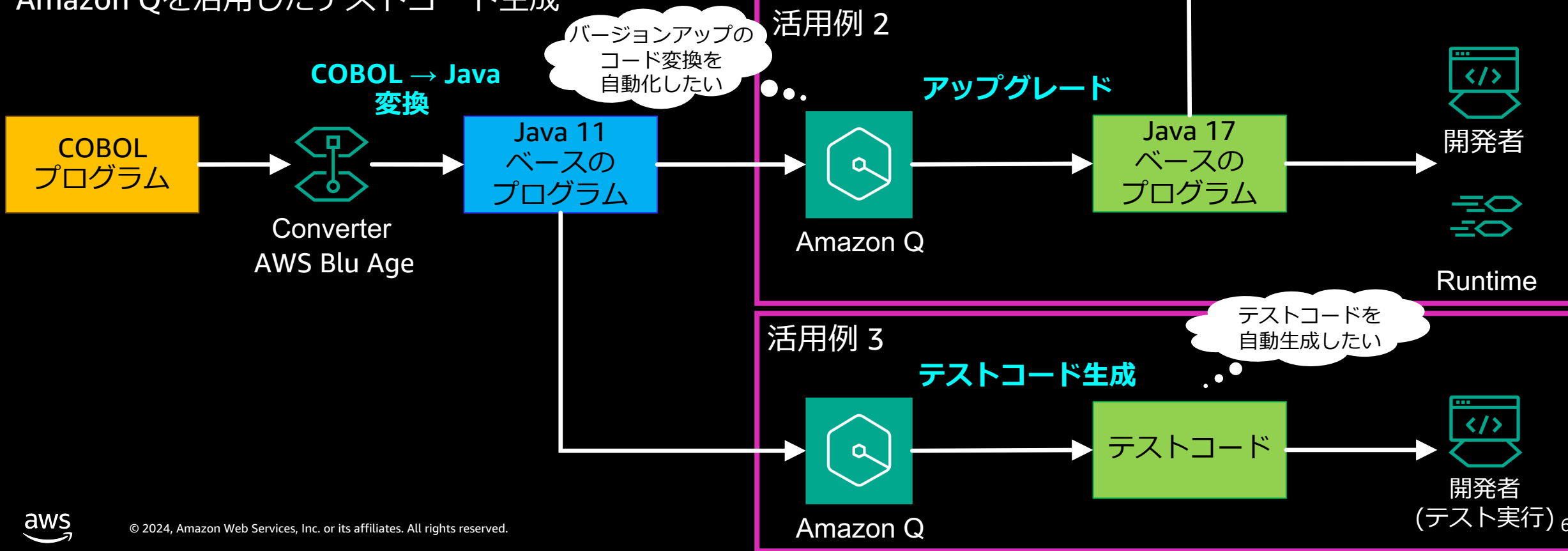
アプリで使用しているプログラミング言語のバージョンのサポート期限が切れてしまったが  
アップグレードに工数を割けない

他人が開発したコードに  
コメントがなく理解が大変

新規開発に忙しくて  
テストコードを書けていない

# こちらをやってみた

- 活用例 1:  
Amazon Bedrockを活用したJava コード内容理解
- 活用例 2:  
Amazon Qを活用したJavaバージョンアップ
- 活用例 3:  
Amazon Qを活用したテストコード生成

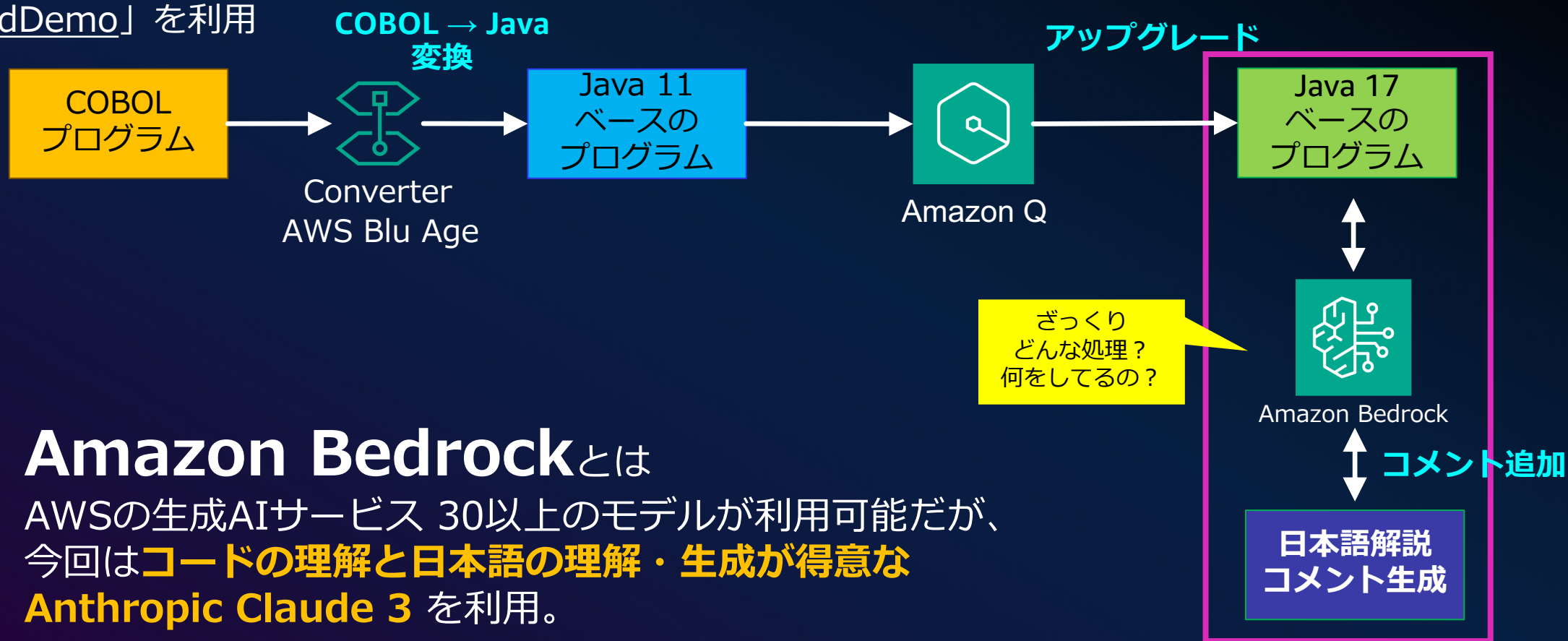




# 活用例1：Amazon Bedrock を活用した Java コード内容理解

# 全体像

COBOL製アプリケーションの「CardDemo」を利用



## Amazon Bedrockとは

AWSの生成AIサービス 30以上のモデルが利用可能だが、今回は**コードの理解と日本語の理解・生成が得意な Anthropic Claude 3** を利用。



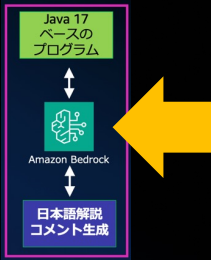
# Blu Ageによって変換されたJava コード(抜粋)



```
public class ConfigurationHelper {  
  
    private static final Logger LOGGER = LoggerFactory.getLogger(ConfigurationHelper.class);  
  
    private static final String PATH_PROPERTY = "configuration.file";  
    private static final String FILES = "files";  
    private static final String PROGRAMS = "programs";  
    private static final String DATASOURCES = "datasources";  
  
    private final String filename;  
    private final Map<String, Object> configuration;  
    private final Map<String, List<DatasourceConfiguration>> datasourceConfigurations = new HashMap<>();  
  
    public ConfigurationHelper() {  
        this("ds-config.yml");  
    }  
}
```

コードにコメントがないと可読性が低下するため、生成AIを活用して可読性を上げる

# コメント生成のためにAmazon Bedrockに与える指示(抜粋)



## 役割を指示

<your role>あなたは優秀なJavaプログラマーで、日本語でわかりやすくプログラムコード解説ができるプロフェッショナルです。</your role>

## 実施内容を指示

<instruction>以下のJavaコードを読み込んで、そのコードの処理内容を理解するため冒頭に解説とパラメータ、戻り値と各コードブロックにおいて、どのような処理を行っているのか、なるべくリッチに各行にコメントを挿入してください。あなたが出力する全ての内容は最初から最後まで、そのままJavaコードとして扱われます。そのため、あなたはコンパイルエラーを避けるため、コメント形式以外の出力は絶対に行わないでください。</instruction>

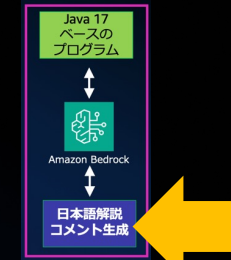
## 制約を指示

<constraint>期待する出力の例を sample に記載するので、この形式を参考にして、可能な限り丁寧に出力してください</constraint>

## 例を指示

```
<sample> /¥*¥* 配列の要素の合計値を計算するメソッド__@param numbers 合計値を求める整数の配列_@return 配列の要素の合計値 ¥*/ public static int calculateSum(int¥[¥] numbers) { int sum = 0; // 合計値を格納する変数を初期化 for (int num : numbers) { // 配列の各要素に対して sum += num; // 要素の値を合計値に加算 } return sum; // 計算された合計値を返す } </sample>
```

# Amazon Bedrockの出力結果(抜粋)



オレンジ色がAmazon Bedrockより生成されたコメント

```
/**
 * 設定ファイルを読み込むためのヘルパークラス
 */
@Component
public class ConfigurationHelper {

    private static final Logger LOGGER = LoggerFactory.getLogger(ConfigurationHelper.class); // このクラスのロガーを初期化

    private static final String PATH_PROPERTY = "configuration.file"; // 設定ファイルのパスを指定するプロパティ名
    private static final String FILES = "files"; // 設定ファイル内の "files" セクションのキー
    private static final String PROGRAMS = "programs"; // 設定ファイル内の "programs" セクションのキー
    private static final String DATASOURCES = "datasources"; // 設定ファイル内の "datasources" セクションのキー

    private final String filename; // 読み込む設定ファイル名
    private final Map<String, Object> configuration; // 読み込んだ設定ファイルの内容を格納するMap
    private final Map<String, List<DatasourceConfiguration>> datasourceConfigurations = new HashMap<>(); // プログラムIDごとの
データソース設定を格納するMap

    /**
     * "ds-config.yml" という名前の設定ファイルを読み込むコンストラクタ
     */
    public ConfigurationHelper() {
        this("ds-config.yml");
    }
}
```

Amazon BedrockのClaude3を活用してコードの可読性があがった

# その他の活用例 (Amazon Bedrock を用いた COBOL プログラム説明の生成)

## CBL

```
COBOLSpecDocGenerator > aws-mainframe-modernization-carddemo-main > app > cbl > CBSTM03A.CBL
257 10 UCB-ADDR PIC X(03).
258    BB NULL-UCB  VALUES LOW-VALUES.
259 05 FILLER    PIC X(04).
260    BB END-OF-TIOT  VALUE LOW-VALUES.
261 *****
262 PROCEDURE DIVISION.
263 *****
264 * Check Unit Control blocks
265 *****
266 SET ADDRESS OF PSA-BLOCK  TO PSAPTR.
267 SET ADDRESS OF TCB-BLOCK  TO TCB-POINT.
268 SET ADDRESS OF TIOT-BLOCK TO TIOT-POINT.
269 SET TIOT-INDEX            TO TIOT-POINT.
270 DISPLAY 'Running JCL : ' TIOTNJOB ' Step ' TIOTJSTP.
271
272 COMPUTE BUMP-TIOT = BUMP-TIOT + LENGTH OF TIOT-BLOCK.
273 SET ADDRESS OF TIOT-ENTRY TO TIOT-INDEX.
274
275 DISPLAY 'DD Names from TIOT: ',
276 PERFORM UNTIL END-OF-TIOT
277     OR TIO-LEN = LOW-VALUES
278     IF NOT NULL-UCB
279         DISPLAY ': ' TIOCDDNM ' -- valid UCB'
280     ELSE
281         DISPLAY ': ' TIOCDDNM ' -- null UCB'
282     END-IF
283 COMPUTE BUMP-TIOT = BUMP-TIOT + LENGTH OF TIOT-
284 SET ADDRESS OF TIOT-ENTRY TO TIOT-INDEX
285 END-PERFORM.
286
287 IF NOT NULL-UCB
288     DISPLAY ': ' TIOCDDNM ' -- valid UCB'
289 ELSE
290     DISPLAY ': ' TIOCDDNM ' -- null UCB'
291 END-IF.
292
293 OPEN OUTPUT STMT-FILE HTML-FILE.
294 INITIALIZE WS-TRNX-TABLE WS-TRN-TBL-CNTR.
295
296 0000-START.
297
298 EVALUATE WS-EN-CD
```

入出力ファイル、  
データ、ロジック  
をドキュメント化

### CBSTM03A.CBL

プログラム名  
CBSTM03A

#### プログラム説明

この COBOL プログラムは、トランザクションデータを含むファイルから、プレーンテキストと HTML の 2 つの形式でアカウントステートメントを印刷するバッチプログラムです。

#### 入力ファイル

No	入力ファイル名	概要
1	TRNXFILE	トランザクションデータを含む入力ファイル

#### 出力ファイル

No	出力ファイル名	概要
1	STMTFILE	プレーンテキスト形式のスタートメント出力ファイル
2	HTMLFILE	HTML 形式のスタートメント出力ファイル

#### データ項目

No	名称	概要
1	WS-TRNX-TABLE	トランザクションデータを保持する 2 次元配列
2	STATEMENT-LINES	プレーンテキスト形式のスタートメントを構成するデータ項目

#### 処理詳細

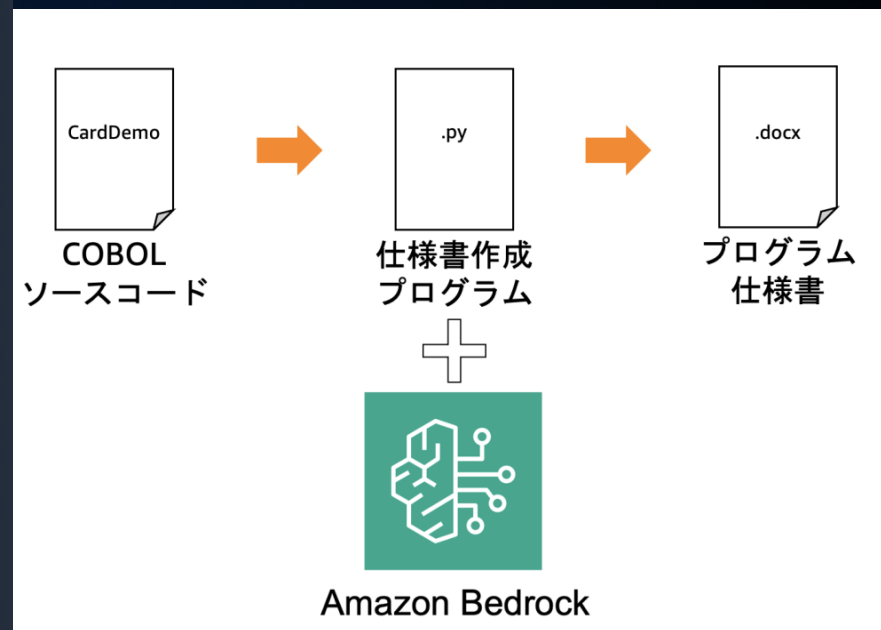
No	概要
1	入力ファイル OPEN
2	トランザクションデータ読み込み
3	顧客データ取得
4	アカウントデータ取得
5	スタートメント作成
6	出力ファイル CLOSE

#### プロシージャ一覧

No	名称	概要
1	1000-MAINLINE	メイン処理

#### COPYBOOK

No	名称	概要
1	COSTM01	トランザクションデータのレイアウト

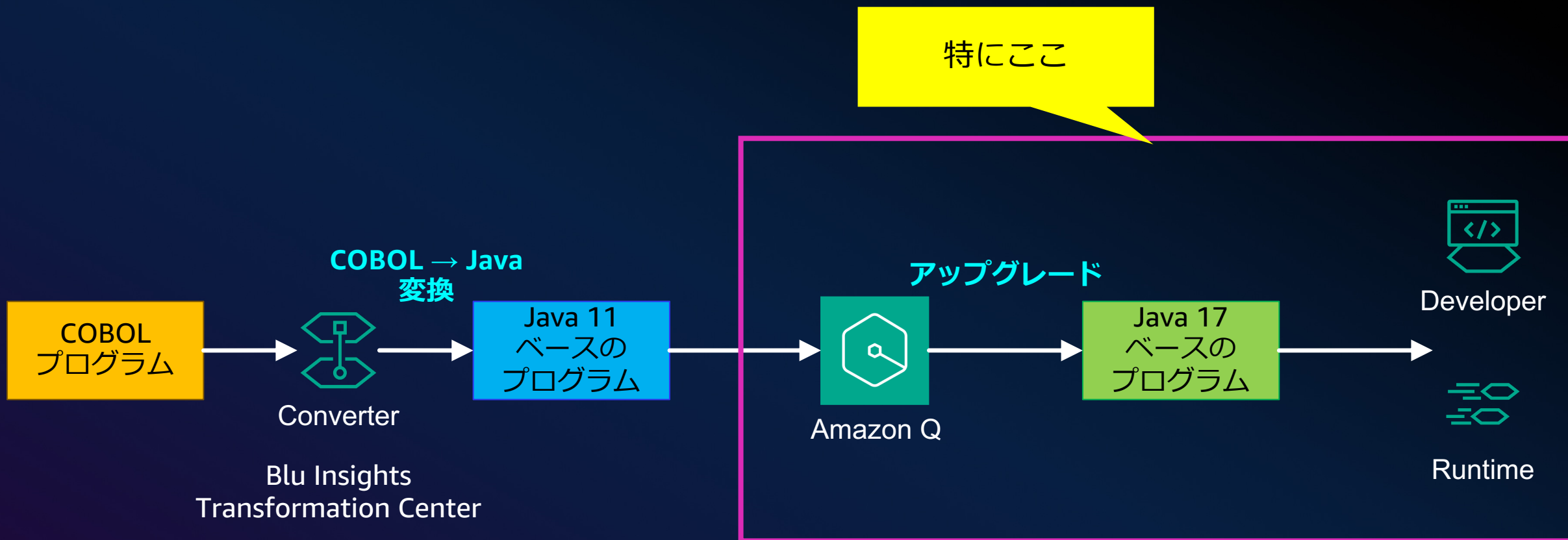


<https://aws.amazon.com/jp/blogs/news/generate-cobol-spec-document-using-amazon-bedrock/>

# 活用例2 : Amazon Q を活用した Java バージョンアップ



# 全体像





# Amazon Q: AWS が開発した生成 AI を搭載したアシスタント

AMAZON Q DEVELOPER



AMAZON Q BUSINESS



コードの作成

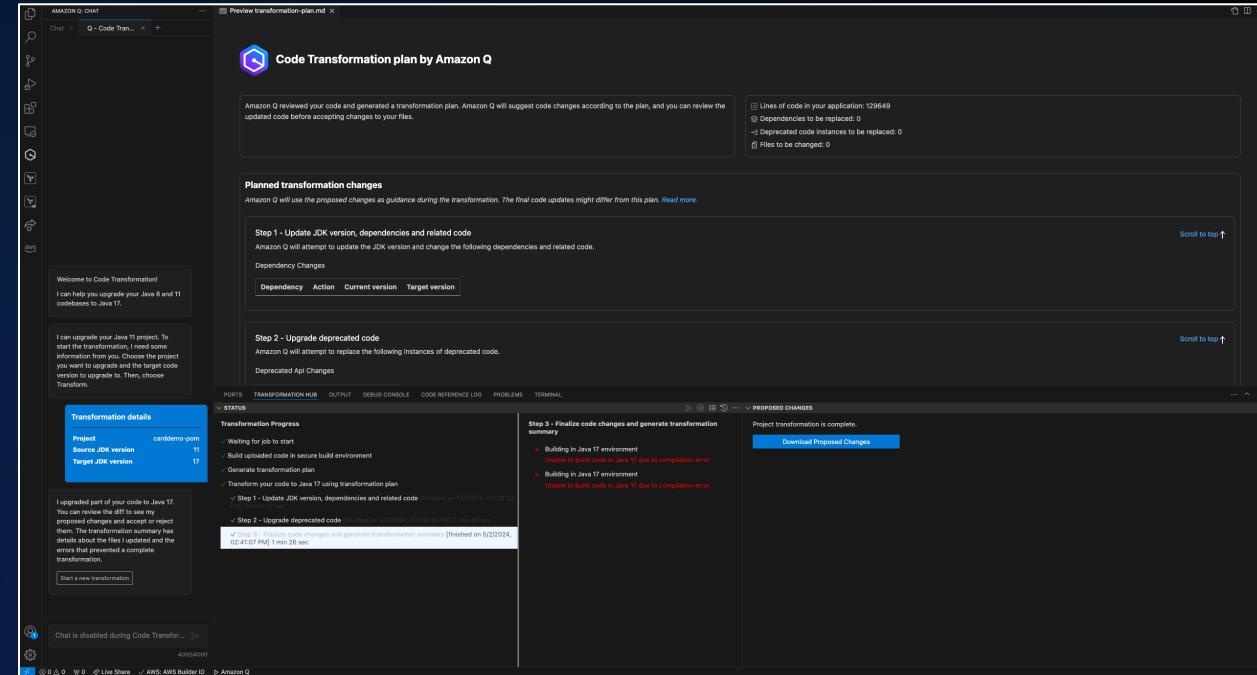
デバッグ

テストコード生成

コードのバージョンアップ

# 1. Java 17 への変換

- [Amazon Q Developer Agent for Code Transformation](#) を使用
- VSCode 上で実施可能
- Maven ベースのアプリが対象



## 2. 設定ファイルの変更

- Servlet API: 6 (Jakarta EE)
- Spring Boot: 3.2
- 他のライブラリを上記に合うよう変更

```
8 <properties>
9   <spring.boot.version>3.2.5</spring.boot.version>
10  <gapwalk.version>4.0.0</gapwalk.version>
11  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
12  <maven.compiler.source>17</maven.compiler.source>
13  <maven.compiler.target>17</maven.compiler.target>
14  <maven.compiler.release>17</maven.compiler.release>
15 </properties>
16 <modules>
17   <module>carddemo-tools</module>
18   <module>carddemo-entities</module>
19   <module>carddemo-service</module>
20   <module>carddemo-web</module>
21 </modules>
```

# 3. Blu Ageでの動作確認

想定通りに動作することを確認

```
carddemo-web Overwrite Mode Setup Theme Help Quit
Tran: CM00          AWS Mainframe Modernization   Date: 05/20/24
Prog: COMEN01C     CardDemo                                           Time: 06:28:49

Main Menu

01. Account View
02. Account Update
03. Credit Card List
04. Credit Card View
05. Credit Card Update
06. Transaction List
07. Transaction View
08. Transaction Add
09. Transaction Reports
10. Bill Payment

Please select an option : 04
```

```
carddemo-web Overwrite Mode Setup Theme Help Quit
Tran: CAVW          AWS Mainframe Modernization   Date: 05/21/24
Prog: COACTVWC     CardDemo                                           Time: 03:59:56

View Account
Account Number : 0000000001 Active Y/N: Y
Opened: 2014-11-20 Credit Limit : + 2,020.00
Expiry: 2025-05-20 Cash credit Limit : + 1,020.00
Reissue: 2025-05-20 Current Balance : + 194.00
Current Cycle Credit: + .00
Account Group: Current Cycle Debit : + .00

Customer Details
Customer id : 00000001 SSN: 020-97-3888
Date of birth: 1961-06-08 FICO Score: 274
First Name Middle Name Last Name :
Immanuel Madeline Kessler
Address: 618 Deshaun Route State NC
Apt. 802 Zip 12546
City Altenwerthshire Country USA
Phone 1: (908)119-8310 Government Issued Id Ref : 000000000049368437
Phone 2: (373)693-8684 EFT Account Id: 0053581756 Primary Card Holder Y/N: Y

Enter or update id of account to display

F3=Exit
```

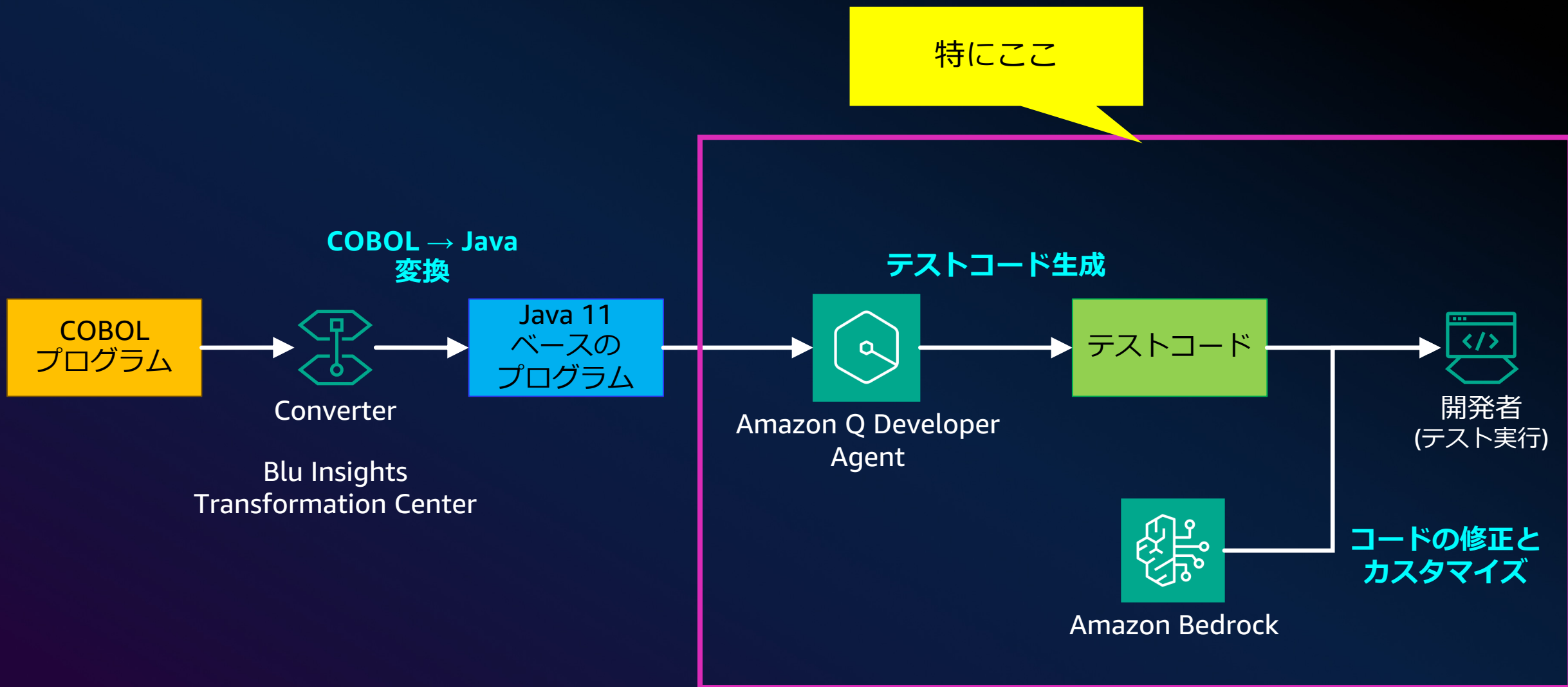
# ポイント

- Java 17 ベースへ変換されたコード以上に、ライブラリの確認が肝
  - Amazon Qは、推奨するライブラリのバージョンは教えてくれるが、ライブラリ自体の変更までは行ってくれない
  - Servlet API 6、Spring Boot 3.2で動作するライブラリを用意できるか？
- 自動テストスクリプトを用意し検証することを推奨
- ほとんどの作業をローカルPCで実施できるため、作業の心理的な敷居を下げることができ、気軽に試すことができる

# 活用例3 : Amazon Q を活用した テストコード生成



# 全体像

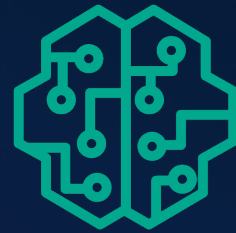


# 生成 AI でテストコードを開発する上での課題

テストコードを生成して



どのクラスのどのメソッドを対象にする？



プロジェクトのディレクトリ構造は？

インポートしている他のクラスの実装は？

生成 AI モデルを扱う際には、いま解きたいタスク (テストコード生成) に必要な情報をプロンプトに入れ込まないといけない

**自律的なソフトウェア開発エージェントの利用が望ましい**

# Amazon Q Developer Agent

Amazon Q Developer Agent はソフトウェア開発ベンチマークの SWE-bench で最高スコアを記録

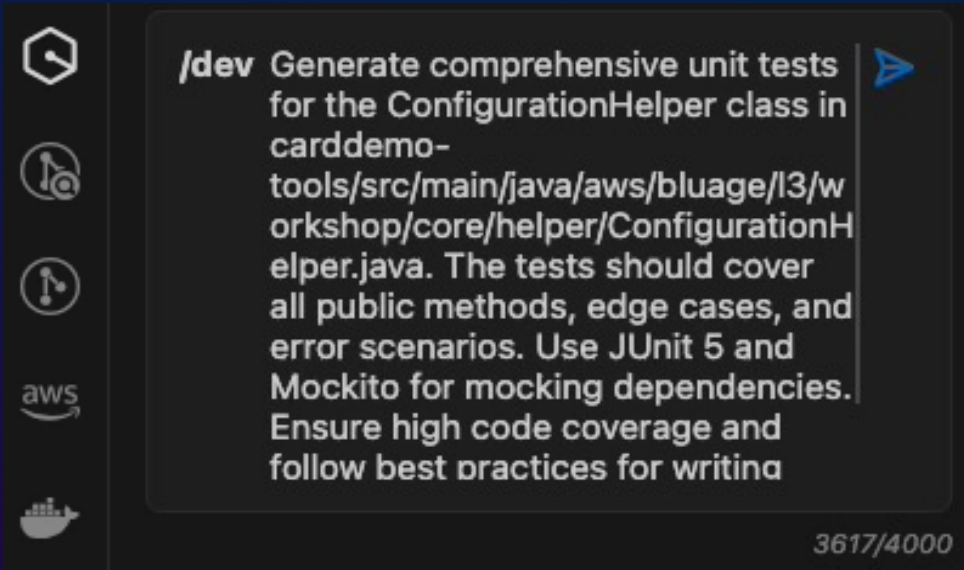
Model	% Resolved	Date	Logs	Trajs	Site	Verified?	Open?
🏆 Amazon Q Developer Agent (v20240430-dev)	13.82	2024-05-09	<a href="#">🔗</a>	-	<a href="#">🔗</a>	✗	✗
🥈 SWE-agent + GPT 4 (1106)	12.47	2024-04-02	<a href="#">🔗</a>	<a href="#">🔗</a>	<a href="#">🔗</a>	✓	✓
🥉 SWE-agent + Claude 3 Opus	10.51	2024-04-02	<a href="#">🔗</a>	<a href="#">🔗</a>	-	✓	✓
RAG + Claude 3 Opus	3.79	2024-04-02	<a href="#">🔗</a>	-	<a href="#">🔗</a>	✓	✓
RAG + Claude 2	1.96	2023-10-10	<a href="#">🔗</a>	-	-	✓	✓
RAG + GPT 4 (1106)	1.31	2024-04-02	<a href="#">🔗</a>	-	-	✓	✓
RAG + SWE-Llama 13B	0.70	2023-10-10	<a href="#">🔗</a>	-	-	✓	✓
RAG + SWE-Llama 7B	0.70	2023-10-10	<a href="#">🔗</a>	-	-	✓	✓
RAG + ChatGPT 3.5	0.17	2023-10-10	<a href="#">🔗</a>	-	-	✓	✓

The **% Resolved** metric refers to the percentage of SWE-bench instances (2294 total) that were resolved by the model.

Amazon Q Developer Agent は、タスク遂行に何が必要か、リポジトリ内のコードを自律的に探索する

# テストコードの生成手順

Amazon Q Developer Agent は VS Code 等の IDE の拡張機能として利用可能



## ① まずはプロンプトに指示を書く

「ConfigurationHelper クラスの包括的なユニットテストを生成してください。」

## ② 実行計画が生成される

「1. pom.xml に JUnit 5 の設定を追加します。  
2. test ディレクトリ以下に ConfigurationHelperTest.java を作成します。  
3. コンストラクタとファイル読み込みのテストコードを生成します。具体的には、(略)」

## ③ 実行計画に基づいてコードを提案

```
class ConfigurationHelperTest {  
  
    private ConfigurationHelper configHelper;  
  
    @TempDir  
    Path tempDir;  
  
    @BeforeEach  
    void setUp() {  
        configHelper = new ConfigurationHelper();  
    }  
  
    @Test  
    void testGetFilesConfiguration() {  
        Map<String, Object> filesConfig = configHelper.getFilesConfiguration();  
        assertNotNull(filesConfig);  
        assertFalse(filesConfig.isEmpty());  
    }  
}
```

## ④ 人手の承認の後、コードを生成

# ポイントと注意事項

- Amazon Q Developer Agent により、リポジトリ全体のコンテキストを把握しながらテストコード生成が可能 (その他機能開発タスクも可能)
- VS Code や JetBrains 系の IDE の拡張機能として利用できる
- Amazon Q Developer Agent では、実行計画を考案した後にコード生成を実行するか、コード生成を行った後にリポジトリにマージするかは人手の承認フローを挟む

## 注意事項

- 生成物は修正が必要なこともある
  - 特にリポジトリ外のコードや、非公開の商用ライブラリが利用されていると難易度が高い
- Amazon Bedrock を利用したアドホックな修正が有用



# まとめ



# ソフトウェア開発ライフサイクルでの生成 AI 活用 ユースケース

## ソフトウェア開発ライフサイクル

要求分析

設計

実装

テスト

デプロイ

保守

- 要件収集や整理
- 既存要件文書の改善
- ユーザーストーリーの生成

- アーキテクチャーや設計パターンの説明
- シーケンス図、フロー図の生成
- データモデル作成
- UX デザイン支援
- 設計抜け漏れチェック

- コード生成
- コード説明
- デバッグ
- コーディングスタイルやベストプラクティス提案
- コードレビューコメント

- テストケース生成
- テスト計画や戦略の作成支援
- テスト結果の解析や報告

- CI/CD コード生成
- IaC コード生成
- 自動化スクリプト作成支援

- リファクタリング
- 問題追跡やバグ報告の整理
- 既存システムのドキュメント改善
- AI アシストのサポート業務
- 開発言語バージョンアップ対応

当資料で紹介した赤枠の活用例以外にも生成AI活用の機会がたくさん考えられる

生成 AI が効率化するユースケース

# Thank you

