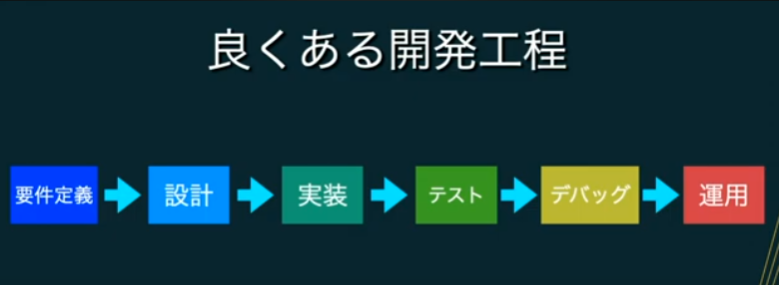
Junit5とは

まずは、よくある開発工程の流れを図にする



実際はテスト工程とデバッグを行き来したりする

デバッグしてテスト工程に後戻りして、繰り返しテストしたらバグが見つかっての繰り返し

最もメジャーなテストフレームワーク

Junit5は３つのサブプロジェクトに含まれる複数のモジュールで構成される

JUnit 5 = JUnit Platform + JUnit Jupiter + JUnit Vintage

**Junit Platform**

・JVM上でテストフレームワークを起動するための基盤となる

・上記のプラットフォーム上で動作するテストフレームワークを開発するためのTestEngine APIを定義している

・コマンドラインからプラットフォームを起動するためのConsole Launcher や、GradleやMaven用のビルドプラグイン、Junit4ベースのテストランナーなど提供し、あらゆるTestEngineを実行できるようになってる

**JUnit Jupiter**

・JUnit5でテストを書くために最低限必要な依存関係をまとめたもの

**JUnit Vintage**

JUnit3またはJUnit4ベースのテストを実行するためのTestEngineを提供する

とりあえずJUnit5でテストを書き始めたいなら、JUnit Jupiterを依存関係に追加しとけばＯＫ

アノテーション

＠BeforeAll 付与されたメソッドは、一番最初に一度だけ実行される

メソッドはstaticである必要がある

「テスト工程」の問題

・人間が行うテストの信頼性

・コスト、時間の制限から回帰テストが難しい

・デバッグ完了を開発者が確認しない

・バグを指摘される心理的ダメージ

・テスト工程要因のためにバグを残す？

テスト工程の為にわざとバグを残したりするらしい

回帰テスト

・一度クリアしているテストを（デバッグを終えてコードが修正された状態）全てもう一回、既に正しいとされている部分もテストし直す事、後戻りが生じていないか、新しいバグが生まれてないか、既に動いていた部分が壊れていないかなどを検証するテスト

テストを自動化するメリット

・人間よりも高い信頼性

・回帰テストを現実的なコストで実現可能

・デバッグ完了を実装時に確認

・デバッグが楽しくなる

Junitとは

・Javaのテストコードを簡単に実装できるフレームワーク

他の言語にも色々なフレームワークがある

Java　JUnit、TestNG

Python　unittest

PHP PHPUnit

Ruby test/unit RSpec

C# NUnit MSTest

今回のテスト対象:FizzBuzz

・0～99までの数字について

・３の倍数の場合　”Fizz”をプリント

・５の倍数 ”Buzz”をプリント

・どちらでもない場合　数字をプリント

FizzBuzzとはコーニングの面接でやったりするやつ

プログラミングの基礎的な問題や課題の一つ、主にプログラミングの初学者や面接などで使用される

FizzBuzzの問題は、１から順番に数を数えながら以下のルールに従って値を出力すること

・３で割り切れる場合は「Fizz」と出力する

・５で割り切れる場合は「Buzz」と出力する

・３と５で割り切れる場合は「FizzBuzz」と出力する

・上記のいずれでもない場合は、その数値を出力する

例えば１から１５までの数字をFizzBuzzのルールに従って出力すると以下のようになる

1,2,Fizz 4,Buzz, 7,8,Fizz,Buzz, 11,Fizz 13,14,FIzzBuzz

この問題はプログラミングの制御構造（条件分岐やループ）を理解し正しく実装できるかを確認するための簡単な問題

FizzBuzzを解くことでプログラミングの基本的なロジックやアルゴリズムの考え方を養う事ができる

言語によっては、FizzBuzzの解法は短くシンプルに書くことができる

初心者から上級者まで様々なアプローチや解法が存在する

普通のFizzBuzzコード

public class FizzBuzz {

public static void main(String[] args) {

for(int i=0; i<100; i++) {

System.out.println(fizzBuzz(i));

}

}

public static String fizzBuzz(int number) {

// ここを実装

}

}

最小限のJUnitテスト

import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.\*;

class FizzBuzzTest {

@ Test　//テストを行うメソッドだと宣言

void fizzBuzz {

assertEquals(“FizzBuzz”,FizzBuzz.fizzBuzz(15));

}

}

@ Test

・アノテーション

Junitのフレームワークが@ Test というアノテーションを見つけたらテストを実行してくれる

assertEquals

・アサーションメソッド