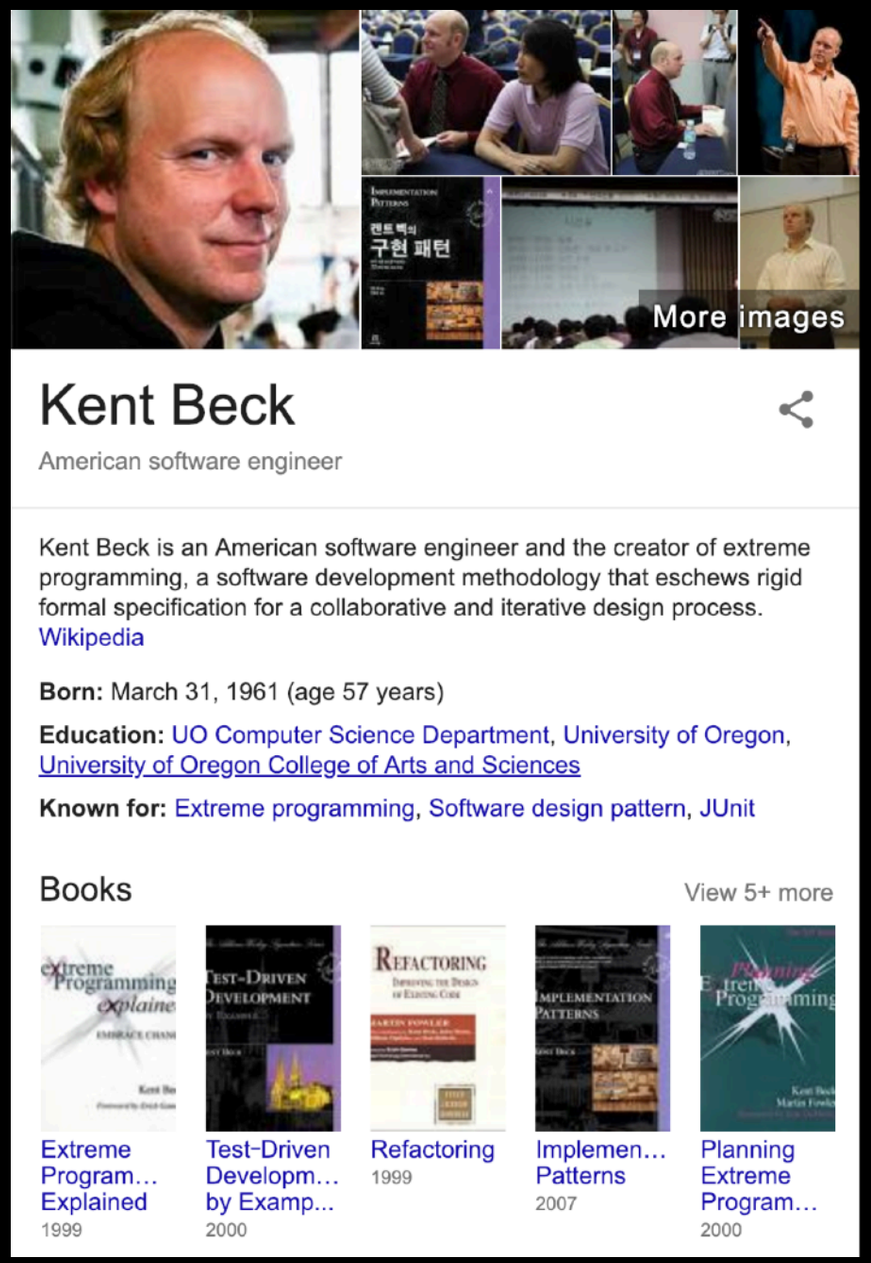
Test-Driven Development

조상연

TDD



켄트 벡

미국의 소프트웨어 엔지니어

익스트림 프로그래밍 창시자

익스트림 프로그래밍

요구사항이 급변하는 분야에 적합한 개발 방법론

구체적인 실천 방법 제안

Whole Team
Planning Game
Customer Tests
Small Releases
Simple Design
Test-Driven Development
Pair Programming

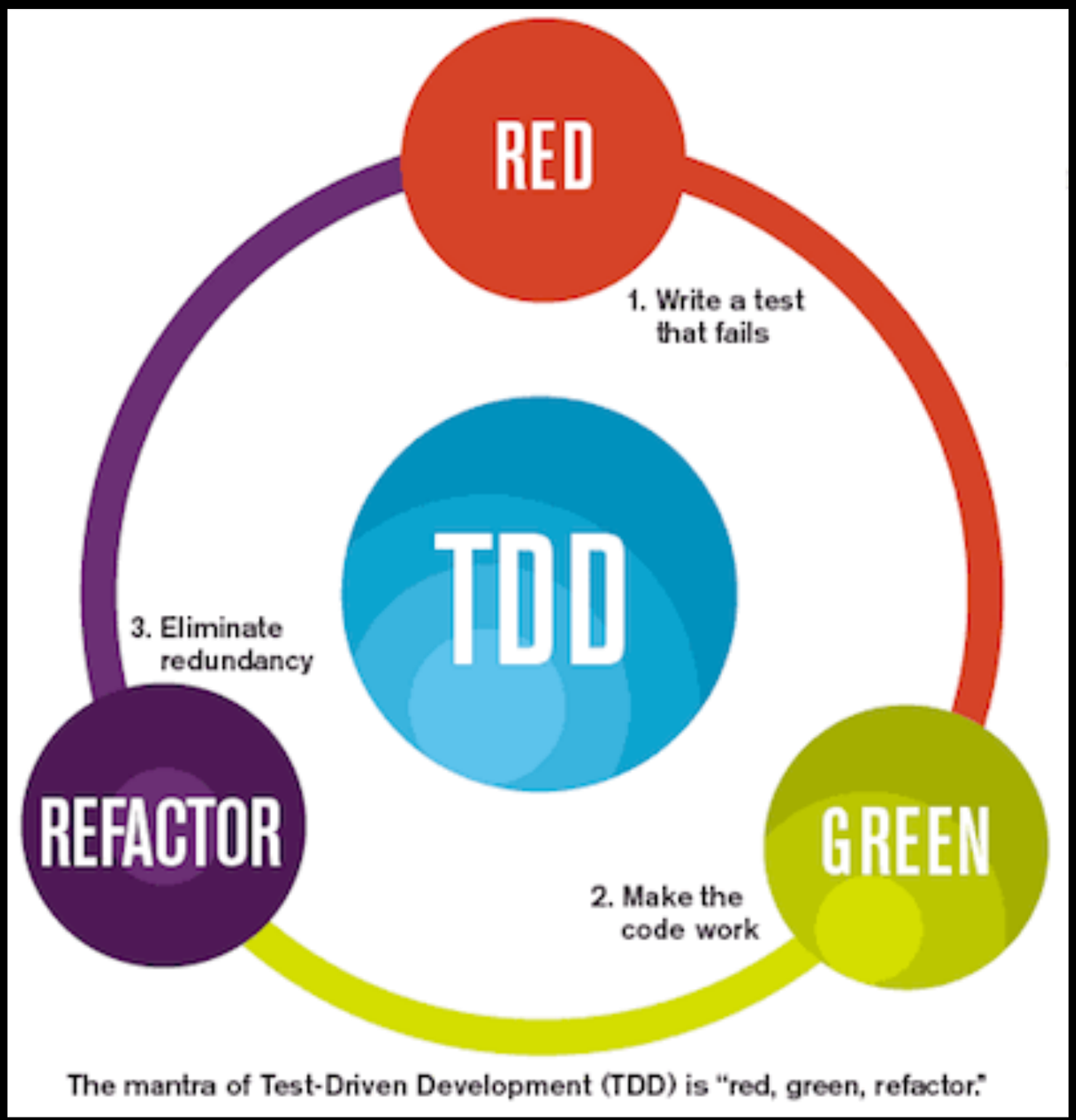
TDD

- What?
- Why?
- How?

What

- 테스트 코드를 먼저 짜고 그 이후에 코드를 짜는 방법
- 1. 요구사항 파악
- 2. 테스트 코드 작성
- 3. 테스트 코드가 돌아가도록 코드 작성
- 4. 리팩토링
- 5. 다시 1번부터

Cycle


RED
1. Write a test that fails
TDD
3. Eliminate redundancy
REFACTOR
2. Make the code work
GREEN
The mantra of Test-Driven Development (TDD) is "red, green, refactor."

Why

- 예전에는…., 해외 취업, 컨트리뷰트
- 장점
  - 1. 동작하는 코드에 대한 자신감 (Clean Code that works)
  - 2. 과도한 설계를 피하고, 간결성 증대
  - 3. 실행 가능한 문서를 가짐
  - 4. 디자인적 유연함, 의존성 관리 편함

How

- Django를 통한 실습

Refactoring

- 코드를 다듬는 과정
- 예시)
  - 쉼표 or 콜론을 구분자로 가지는 문자열을 입력받는 경우
  - 구분자를 기준으로 분리한 각 숫자의 합을 반환

테스트 코드 작성

```java
public class StringCalculatorTest {
    @Test
    public void null_또는_빈값() {
        assertThat(StringCalculator.splitAndSum(null)).isEqualTo(0);
        assertThat(StringCalculator.splitAndSum("")).isEqualTo(0);
    }

    @Test
    public void 값_하나() {
        assertThat(StringCalculator.splitAndSum("1")).isEqualTo(1);
    }

    @Test
    public void 쉼표_구분자() {
        assertThat(StringCalculator.splitAndSum("1,2")).isEqualTo(3);
    }

    @Test
    public void 쉼표_콜론_구분자() {
        assertThat(StringCalculator.splitAndSum("1,2:3")).isEqualTo(6);
    }

}
```

코드 작성

```
public class StringCalculator {
    public static int splitAndSum(String text) {
        int result = 0;
        if (text=null || text.isEmpty()) {
            result = 0;
        } else {
            String[] values = text.split(",|:");
            for (String value : values) {
                result += Integer.parseInt(value);
            }
        }
        return result
    }
}
```

리팩토링 연습

오직 한 단계의 들여쓰기 (indent)를 한다

```
public class StringCalculator {
    public static int splitAndSum(String text) {
        int result = 0;
        if (text=null || text.isEmpty()) {
            result = 0;
        } else {
            String[] values = text.split(",|:");
            for (String value : values) {
                result += Integer.parseInt(value);
            }
        }
        return result
    }
}
```

리팩토링 연습

오직 한 단계의 들여쓰기 (indent)를 한다

```
public class StringCalculator {
    public static int splitAndSum(String text) {
        int result = 0;
        if (text=null || text.isEmpty()) {
            result = 0;
        } else {
            String[] values = text.split(",|:");
            result = sum(values);
        }
        return result
    }
}

private static int sum(String[] values) {
    int result = 0;
    for (String value : values) {
        result += Integer.parseInt(value);
    }
    return result
}
```

리팩토링 연습

else를 하지 않는다

```
public class StringCalculator {
    public static int splitAndSum(String text) {
        int result = 0;
        if (text=null || text.isEmpty()) {
            result = 0;
        } else {
            String[] values = text.split(",|:");
            result = sum(values);
        }
        return result
    }
}

private static int sum(String[] values) {
    int result = 0;
    for (String value : values) {
        result += Integer.parseInt(value);
    }
    return result
}
```

리팩토링 연습

else를 하지 않는다

```
public class StringCalculator {
    public static int splitAndSum(String text) {
        if (text=null || text.isEmpty()) {
            return 0;
        }
        String[] values = text.split(",|:");
        return sum(values);
    }
}

private static int sum(String[] values) {
    int result = 0;
    for (String value : values) {
        result += Integer.parseInt(value);
    }
    return result
}
```

리팩토링 연습

메소드가 한가지 일만 하게 하기

```
public class StringCalculator {
    public static int splitAndSum(String text) {
        if (text=null || text.isEmpty()) {
            return 0;
        }
        String[] values = text.split(",|:");
        return sum(values);
    }
}

private static int sum(String[] values) {
    int result = 0;
    for (String value : values) {
        result += Integer.parseInt(value);
    }
    return result
}
```

리팩토링 연습

메소드가 한가지 일만 하게 하기

```
public class StringCalculator {
    public static int splitAndSum(String text) {
        if (text=null || text.isEmpty()) {
            return 0;
        }
        String[] values = text.split(",|:");
        int[] numbers = toInts(values);
        return sum(values);
    }
}

private static int[] toInts(String[] values) {…
}

private static int sum(int[] values) {…
}
```

리팩토링 연습

로컬 변수가 꼭 필요한가?

```
public class StringCalculator {
    public static int splitAndSum(String text) {
        if (text=null || text.isEmpty()) {
            return 0;
        }
        String[] values = text.split(",|:");
        int[] numbers = toInts(values);
        return sum(values);
    }
}

private static int[] toInts(String[] values) {…
}

private static int sum(int[] values) {…
}
```

리팩토링 연습

로컬 변수가 꼭 필요한가?

```
public class StringCalculator {
    public static int splitAndSum(String text) {
        if (text=null || text.isEmpty()) {
            return 0;
        }
        return sum(toInts(text.split(",|:")));
    }
}

private static int[] toInts(String[] values) {…
}

private static int sum(int[] values) {…
}
```

리팩토링 연습

Compose Method 패턴 적용

추상화 정도가 같아야 한다

```
public class StringCalculator {
    public static int splitAndSum(String text) {
        if (text=null || text.isEmpty()) {
            return 0;
        }
        return sum(toInts(text.split(",|:")));
    }
}

private static int[] toInts(String[] values) {…
}

private static int sum(int[] values) {…
}
```

리팩토링 연습

Compose Method 패턴 적용

추상화 정도가 같아야 한다

```
public class StringCalculator {
    public static int splitAndSum(String text) {
        if (isBlank(text)) {
            return 0;
        }
        return sum(toInts(split(text)));
    }
}
private static boolean isBlank(String text) {…
}
private static String[] split(String text) {…
}
private static int[] toInts(String[] values) {…
}
private static int sum(int[] values) {…
}
```

최종

```java
public class StringCalculator {
    public static int splitAndSum(String text) {
        int result = 0;
        if (text=null || text.isEmpty()) {
            result = 0;
        } else {
            String[] values = text.split(",|:");
            for (String value : values) {
                result += Integer.parseInt(value);
            }
        }
        return result
    }
}
```

```java
public class StringCalculator {
    public static int splitAndSum(String text) {
        if (isBlank(text)) {
            return 0;
        }
        return sum(toInts(split(text)));
    }
}
private static boolean isBlank(String text) {…
}
private static String[] split(String text) {…
}
private static int[] toInts(String[] values) {…
}
private static int sum(int[] values) {…
}
```