
TDD

1장 ~ 3장

23/08/03

들어가는 장

01 권장하는 TDD 훈련 방법

- 간단하고 쉬운 문제들을 TDD로 시도한다. 가능하면 전에 접하고, 프로그래밍해본 문제가 좋다.
- 초록 막대 주기(이전 초록 막대와 다음 초록 막대 사이의 간격)는 가능하면 짧도록 한다.
- 초록 막대 주기의 최대 시간을 정하고, 이를 초과하면 이전 초록 막대 상태로 돌리고 새로 시작한다.
- '진짜로 만들기 전까지만 가짜로 구현하기' 를 적극적으로 사용하려고 노력한다.
- 같은 문제를 여러번풀어본다.
- 초기에는 리팩토링 툴을 사용하지 않는 것이 좋다.

01 권장하는 TDD 훈련 방법

개발을 할 때 하항상 다음 두가지 법칙을 따른다.

- 어떤 코드건 작성하기 전에 실패하는자동화된 테스트를 작성할 것
- 중복을 제거할 것

뱅크 샐러드 개발자님과과의 만남

최근 실제로 사내 대부분의 IOS 사업부에서 사용할 스위프트 자동화 테스트 툴을 만들어 배포하고, TDD를 적극 선호하는 개발자님과 만나 사건을 들을 기회가 있었다. 그 분이 강조하신 사항은 다음과 같다.

01 뱅크 샐러드 개발자님과과의 만남

TDD로 개발했을 때 좋은 점은?

버그를 잡기 위해서 X

문서 작성에 매우 도움이 된다 ○

TDD로 개발을 할 때 버그를 잡고자 하는 기대는 하지 말자. 시간에 비해 효과 X
TDD는 나의 코드의 엄밀성을 증명하는 문서를 작성할 때 큰 도움이 된다.

01 뱅크 샐러드 개발자님과과의 만남

EXTREME PROGRAMMING

결국은 TDD도 CI/CD 등과 같은 더 잦은 양질의 피드백을 사람이 아닌 머신으로부터 얻기 위한 익스트림 프로그래밍의 한 방법론이다. TDD에 너무 목매지 말 것.

애자일에서의 단위 테스트는 유저의 스토리를 기반으로 해야 한다.
예를 들어 버튼을 눌렀을 때 작동해야 하는 일이라던가 , 스와이프를 했을 때의 반응 등등.

이는 관점에 따라 (변수 , 클래스 혹은 프로그램 까지) 통합 테스트로 보일 수도, 단위 테스트로 보일 수도 있다.

01 뱅크 샐러드 개발자님과과의 만남

TDD를 실제로 경험해보는 법

9월에 인프런에 강좌를 낼 예정이다. (홍보인가?)

신입 개발자를 뽑을 때

- 내 코드의 무결성을 테스트 코드로 증명할 수 있다면 BEST이다.
- QA 자격증인 ISTQB 또한 큰 도움이 된다.

1부 화폐 예제

01장 다중 통화를 지원하는 Money 객체

TDD의 리듬

- 재빨리 테스트를 하나 추가 (가짜로 구현)
- 모든 테스트를 실행하고, 새로 추가한 것의 실패 여부 확인
- 코드 수정
- 모든 테스트를 실행하고, 전부 성공하는지 확인
- 리팩토링을 통한 중복 제거

01장 다중 통화를 지원하는 Money 객체

다음과 같은 MONEY 객체가 있다고 가정해보자.

종목	주	가격	합계
IBM	1000	25	25000
GE	400	100	40000
		합계	65000

01장 다중 통화를 지원하는 Money 객체

다음과 같은 MONEY 객체가 있다고 가정해보자.

종목	주	가격	합계
IBM	1000	25	25000
GE	400	100	40000
		합계	65000

01장 다중 통화를 지원하는 Money 객체

이를 다중 통화를 지원하는 보고서를 만들기 위해선 통화 단위를 추가하는 등 수정이 불가피하다.

종목	주	가격	합계
IBM	1000	25USD	25000USD
GE	400	100CHF	40000CHF
		합계	65000

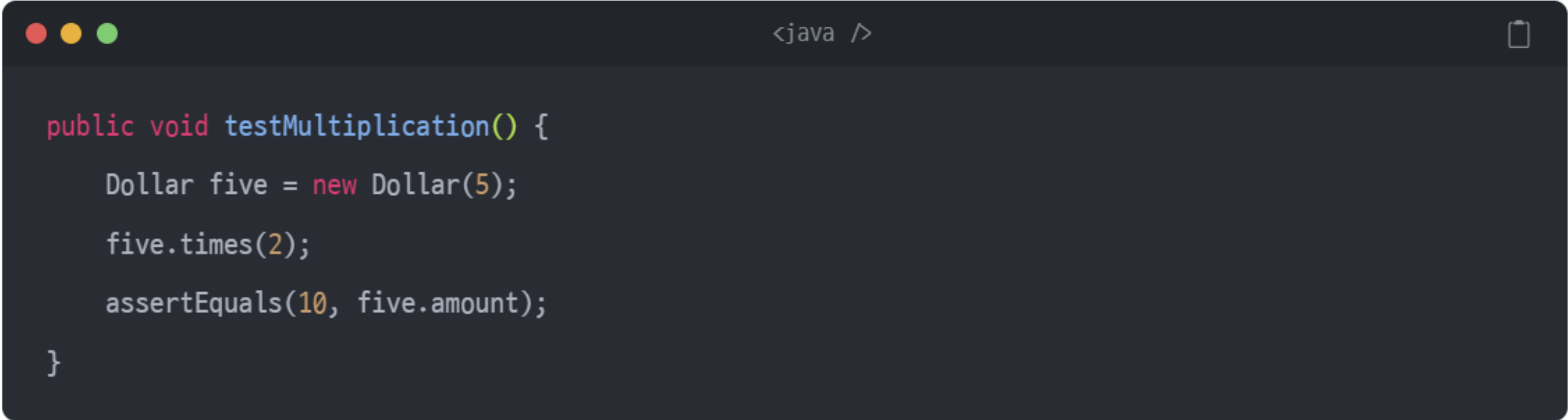
기준	변환	환율
CHF	USD	1.5

종목	주	가격	합계
IBM	1000	25USD	25000USD
GE	400	100CHF	40000CHF
		합계	65000

기준	변환	환율
CHF	USD	1.5

01장 다중 통화를 지원하는 Money 객체

객체를 만들면서 시작하는 것이 아닌 테스트를 먼저 만들어야 한다.
가장 빨리 간단한 테스트를 위해, 주식 개수만큼 값을 곱해 금액을 구하는
함수를 만들어보자 .

A code editor window with a dark background and light-colored text. The window has a title bar with three colored circles (red, yellow, green) on the left and a tab labeled "<java />" on the right. The code is a Java test method named testMultiplication. It creates a Dollar object named five with a value of 5, calls the times method with 2, and then asserts that the amount is 10.

```
<java />

public void testMultiplication() {
    Dollar five = new Dollar(5);
    five.times(2);
    assertEquals(10, five.amount);
}
```

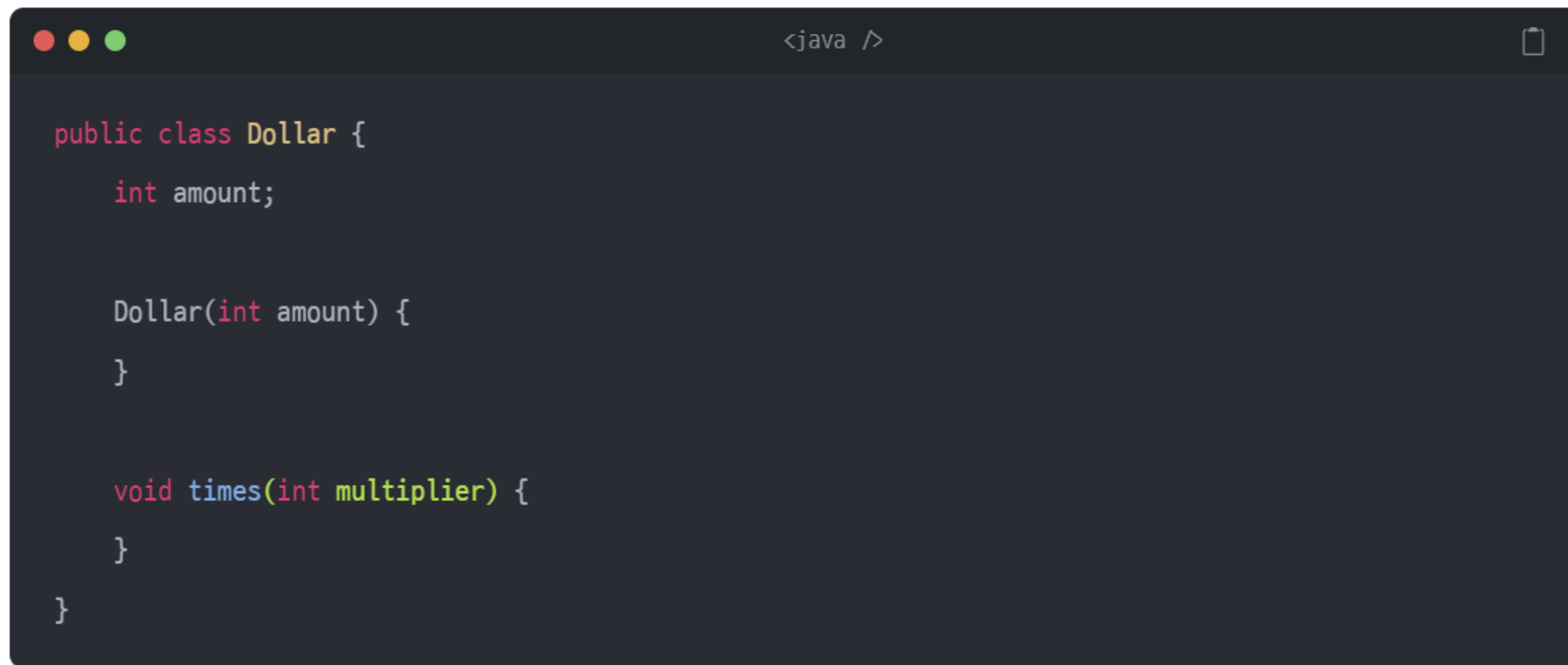
01장 다중 통화를 지원하는 Money 객체

할 일 목록

- $\$5 + 10\text{CHF} = \10 (환율이 2:1인 경우)
- $5 \times 2 = 10$ (가지고 있는 주식 수 만큼 가격 계산)
- amount를 private으로 만들기
- Dollar 부작용?
- money 반올림?

01장 다중 통화를 지원하는 Money 객체

위 코드를 실행하면 다음 컴파일 에러들이 발생한다.
빨간 막대를 보기 위해 컴파일 에러부터 해결한다.



```
<java />

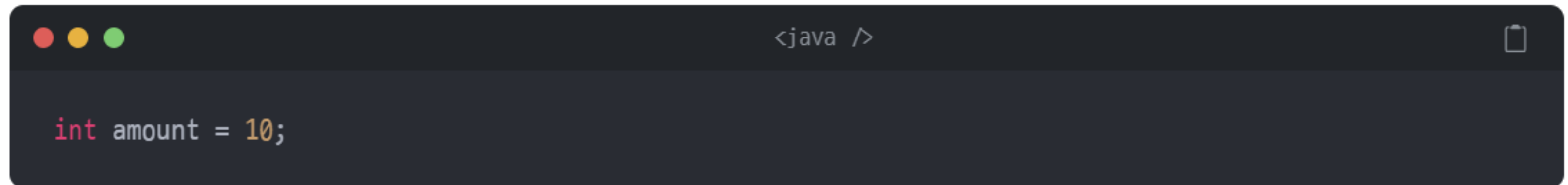
public class Dollar {
    int amount;

    Dollar(int amount) {
    }

    void times(int multiplier) {
    }
}
```


01장 다중 통화를 지원하는 Money 객체

테스트를 실행하면, 결과로 10이 나와야 하지만 0이 나오기 때문에, 빨간 막대를 보게 된다.
최소한의 작업으로 초록 막대를 보려면 다음과 같이 수정한다.

A code editor window with a dark background. The title bar shows three colored circles (red, yellow, green) on the left, the text "<java />" in the center, and a clipboard icon on the right. The code area contains the text "int amount = 10;" where "int" is red, "amount" is white, "=" is white, "10" is orange, and ";" is white.

```
<java />  
  
int amount = 10;
```

01장 다중 통화를 지원하는 Money 객체

마지막으로 중복 제거를 실행한다.

A code editor window with a dark background and light-colored text. The window has a title bar with three colored circles (red, yellow, green) on the left and a close button on the right. The code is written in Java and defines a public class named Dollar. It has a private integer attribute named amount. The constructor Dollar(int amount) initializes the amount attribute. There is also a method named times(int multiplier) that multiplies the amount by the multiplier. The code is as follows:

```
<java />

public class Dollar {
    int amount;

    Dollar(int amount) {
        this.amount = amount;
    }

    void times(int multiplier) {
        amount *= multiplier;
    }
}
```

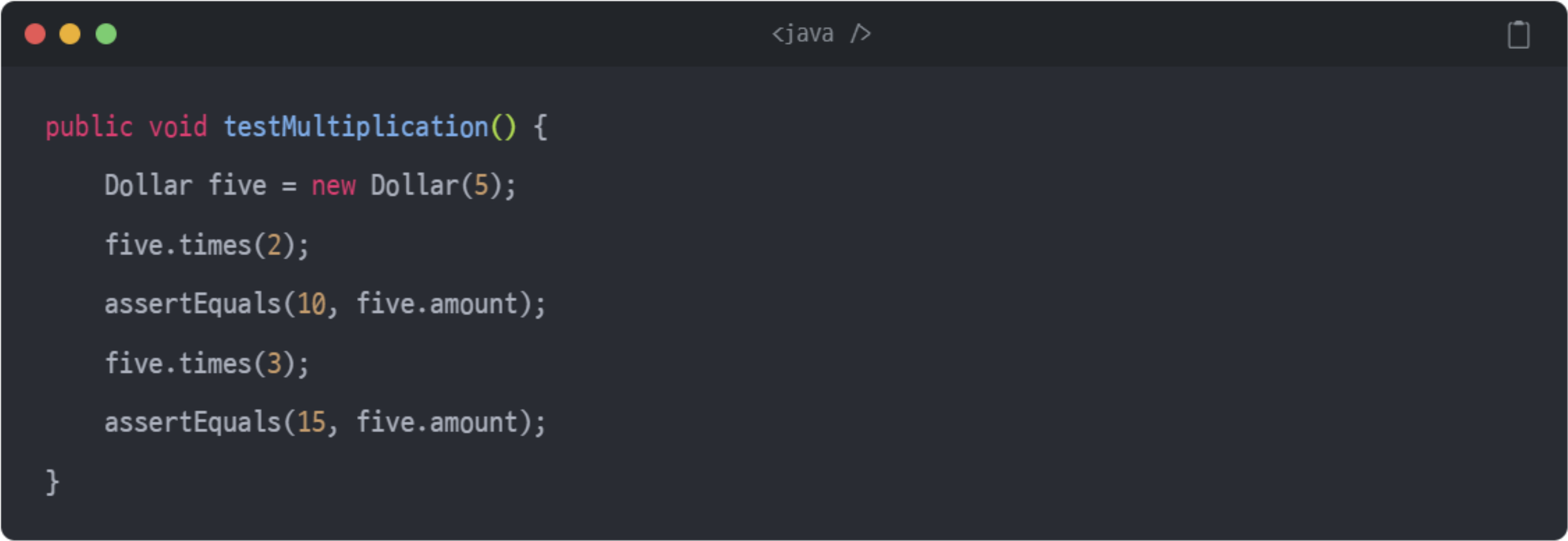
01장 다중 통화를 지원하는 Money 객체

1장에서 한 작업들

- 알고 있는, 작업해야 할 테스트 목록 만들기
- 오퍼레이션이 외부에서 어떻게 보이길 원하는지 코드로표현
- junit 상세 사항 무시
- 스텝 구현(구현되지 않은 함수 임의 생성)을 통해 테스트 컴파일
- 꼼직한 구현(하드 코딩) 을 통해 테스트 통과
- 상수를 변수로 변경하며 점진적 일반화
- 새로운 할일들을 처리하는 대신 할 일 목록에 추가하고 넘어가기

02장 타락한 객체

현재까지 짠 코드에서는 Dollar에 대한 연산을 수행하고 나면 Dollar의 값이 바뀌고 만다.

A code editor window with a dark background and light-colored text. The window has a title bar with three colored circles (red, yellow, green) on the left and a tab labeled "<java />" on the right. The code is as follows:

```
public void testMultiplication() {  
    Dollar five = new Dollar(5);  
    five.times(2);  
    assertEquals(10, five.amount);  
    five.times(3);  
    assertEquals(15, five.amount);  
}
```

02장 타락한 객체

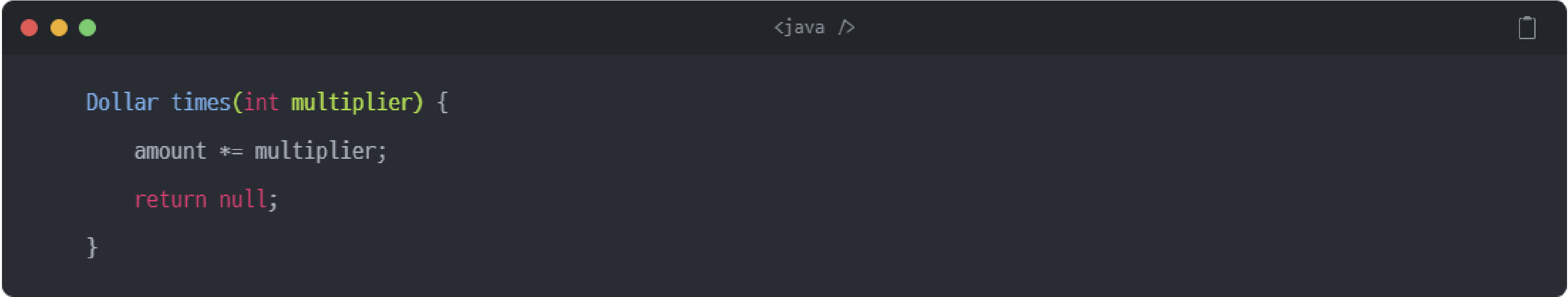
한번 times를 실행하고 나면, 객체는 값이 바뀌고 오염되므로, 새로운 객체를 반환하기를 원한다.

```
<java />

public void testMultiplication() {
    Dollar five = new Dollar(5);
    Dollar product = five.times(2);
    assertEquals(10, product.amount);
    product = five.times(3);
    assertEquals(15, product.amount);
}
```

02장 타락한 객체

우선 컴파일이 되도록 한다.

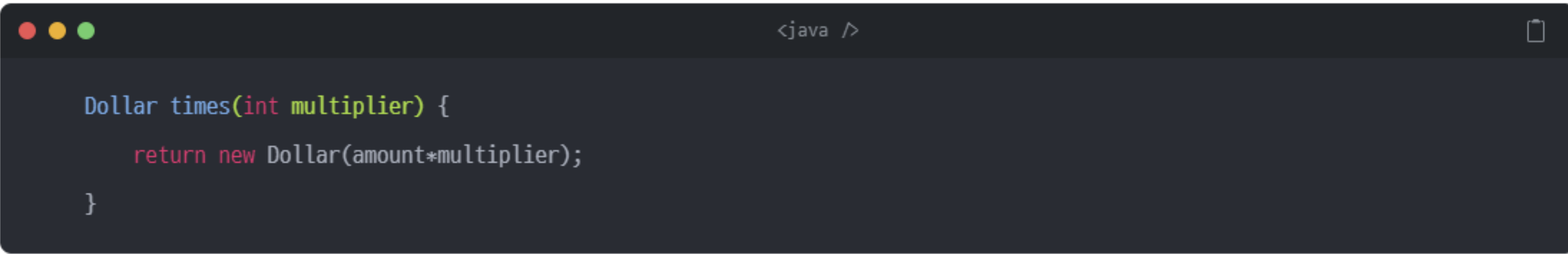


```
<java />

Dollar times(int multiplier) {
    amount *= multiplier;
    return null;
}
```

02장 타락한 객체

올바른 값을 반환해 테스트를 통과하게 해주자.

A code editor window with a dark background. The title bar shows a red, yellow, and green window control icon on the left, the text "<java />" in the center, and a clipboard icon on the right. The code inside is a Java method named "times" in the "Dollar" class, which takes an "int multiplier" parameter and returns a new "Dollar" object with the value "amount*multiplier".

```
Dollar times(int multiplier) {  
    return new Dollar(amount*multiplier);  
}
```

02장 타락한 객체

지금까지 해온 TDD의 흐름은 다음과 같다.

- 설계상의 결함을 그 결함으로 인해 실패하는 테스트로 변환했다.
- 스텝 구현으로 빠르게 컴파일을 통과하도록 했다.
- 올바르다고 생각하는 코드를 입력해 테스트를 통과한다.


03장 모두를 위한 평등

객체의 값(정확히는 amount) 를 비교해야 하기 때문에 equals 구현
모든 인스턴스 변수들이 그렇듯이 amount를 private로 만들기 위해 필요.

Dollar를 해시 테이블의 키로 쓸 경우엔 , hashCode()도 구현 필요.
(다만, 바로 작업하지 않고 할일 목록에 적어둔다.)

03장 모듈을 위한 평등

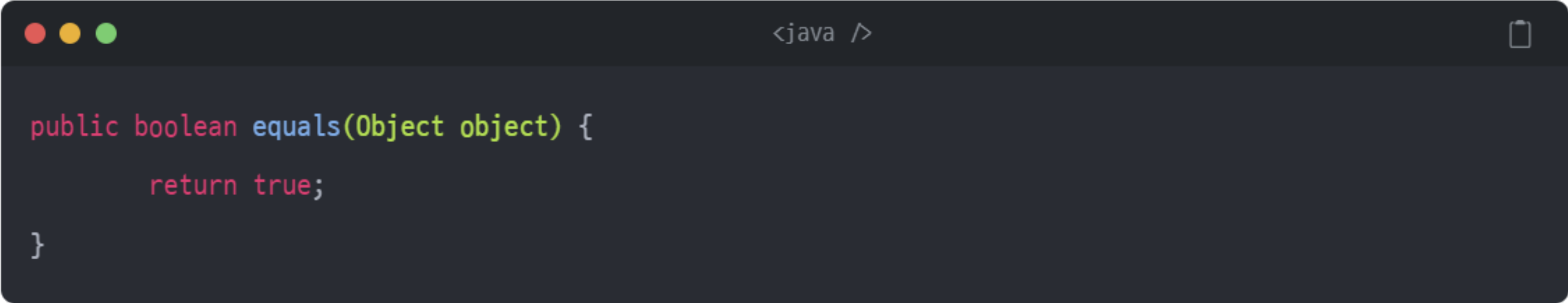
테스트 코드 작성

A code editor window with a dark background. The title bar shows three colored circles (red, yellow, green) on the left, the text "<java />" in the center, and a clipboard icon on the right. The code is written in a light color with syntax highlighting: "public void testEquality()" in red, "assertEquals" in blue, "new Dollar(5)" in red, and "5" in yellow. The code is enclosed in curly braces.

```
<java />  
  
public void testEquality() {  
    assertEquals(new Dollar(5), new Dollar(5));  
}
```

03장 모듈 위한 평등

빨간 막대 제거를 위한 가짜 구현



```
<java />

public boolean equals(Object object) {
    return true;
}
```

03장 모듈 위한 평등

삼각 측량이란 ?

쉽게 말해 예제가 두개 이상 있어야지만 코드를 일반화 할 수 있다는 것.

\$5 == \$5 로 true 만 검사하는 것이 아니라

\$5 != \$6와 같이 false 도 검사할 필요가 있다.

```
<java />

public void testEquality() {
    assertTrue(new Dollar(5).equals(new Dollar(5)));
    assertFalse(new Dollar(5).equals(new Dollar(6)));
}
```

03장 모듈을 위한 평등

equality 일반화

```
<java />  
  
public boolean equals(Object object) {  
    Dollar dollar = (Dollar) object;  
    return amount == dollar.amount;  
}
```

03장 모듈을 위한 평등

동일성 문제는 일시적으로 해결했으나,
널 값이나 다른 객체와 비교할 때는 해결 x
따라서 할일 목록에 다음 추가

- Equal null
- Equal object

03장 모듈을 위한 평등

이 장에서 한 작업들 검토

- 오퍼레이션을 테스트로 작성했다.
- 오퍼레이션을 간단히 구현했다. (항상 true를 반환하는 스텝)
- 공장 리팩토링 하는 대신 테스트를 좀 더 보완했다.
- 두 경우를 모두 수용할 수 있도록 리팩토링했다.