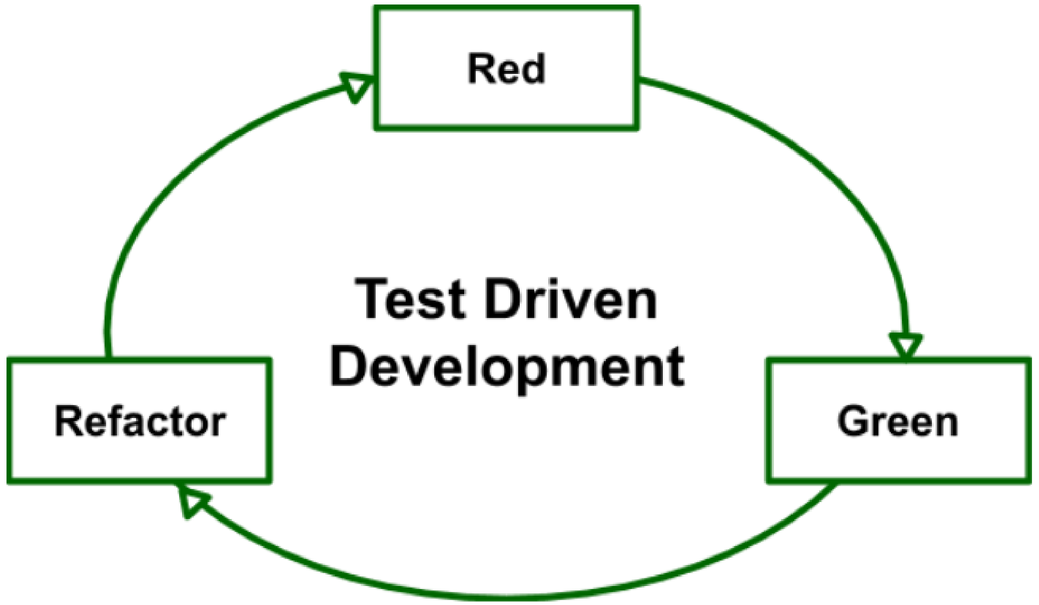
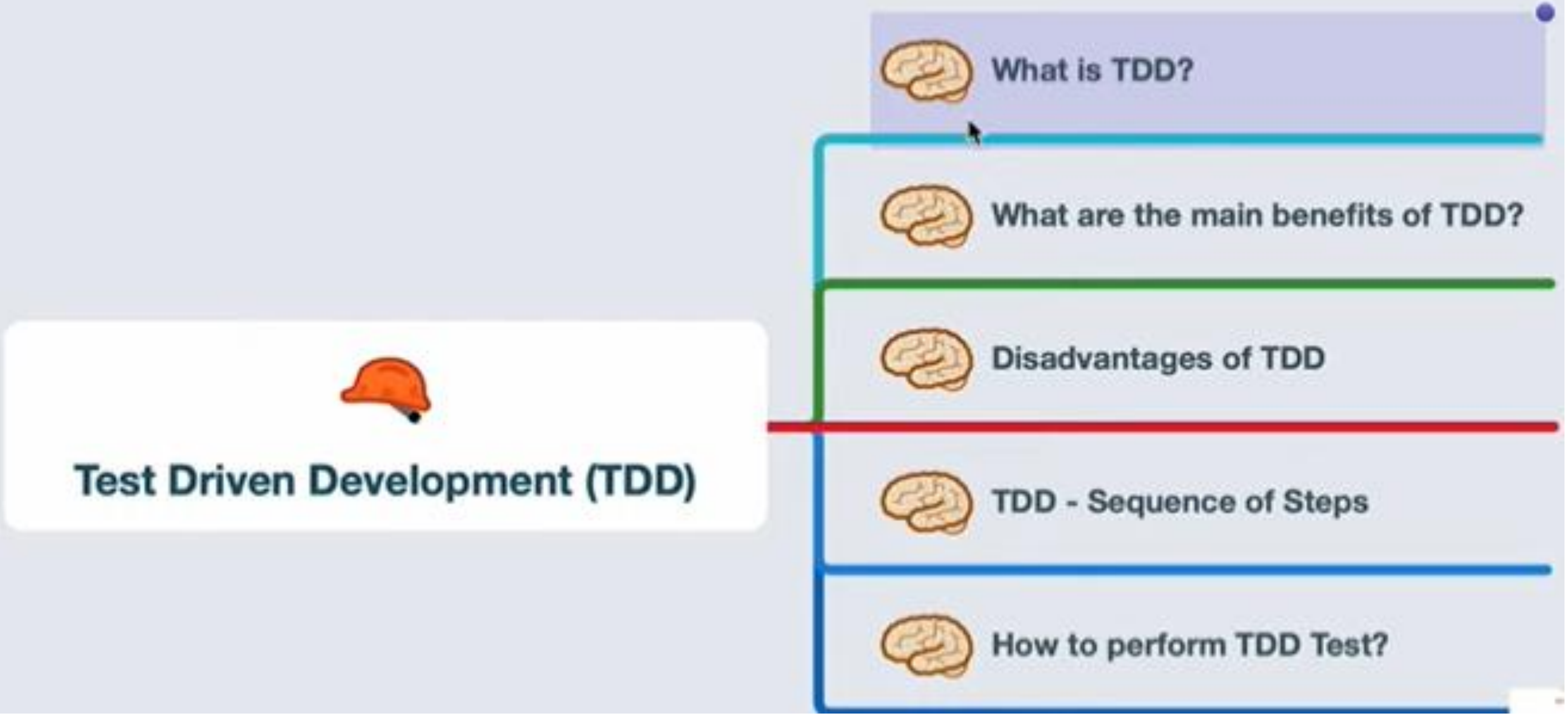


TDD

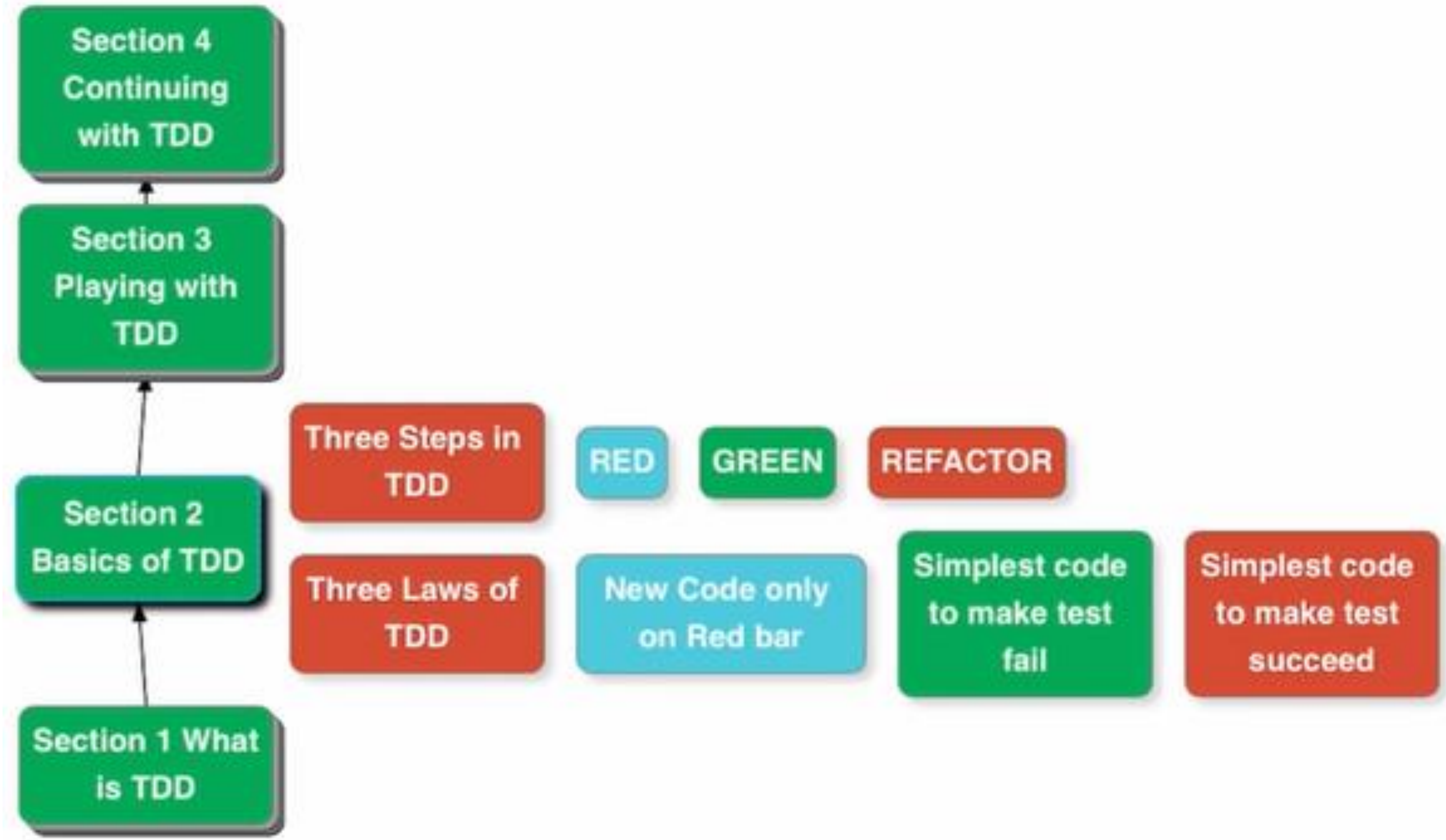
Test Driven Development

What is Test Driven Development (TDD)? with Example



Test Driven Development (TDD)

- What is TDD?
- What are the main benefits of TDD?
- Disadvantages of TDD
- TDD - Sequence of Steps
- How to perform TDD Test?


Section 4 Continuing with TDD
Section 3 Playing with TDD
Three Steps in TDD
RED
GREEN
REFACTOR
Section 2 Basics of TDD
Three Laws of TDD
New Code only on Red bar
Simplest code to make test fail
Simplest code to make test succeed
Section 1 What is TDD

Test First Development (TFD)

TDD Rhythm – Test, Code, Refractor

Steps for Test First Development:

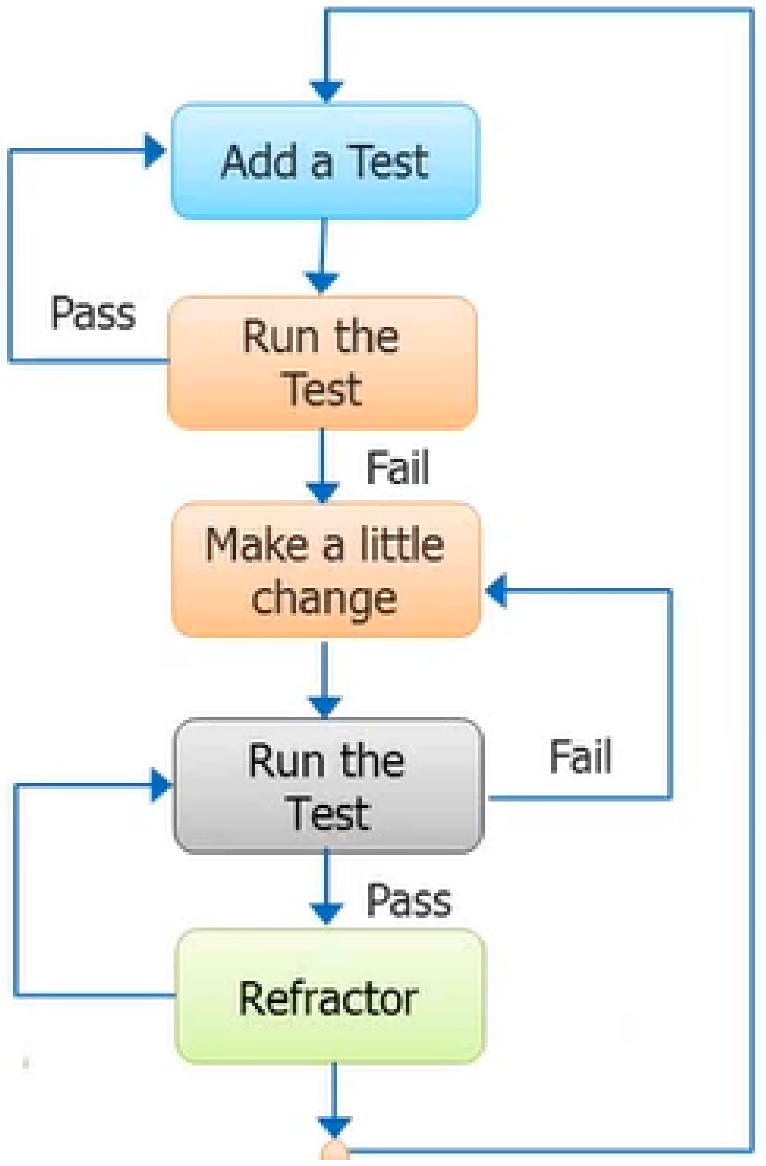
Step 1: Add a test - Basically enough code to fail

Step 2: Run your tests – Take a subset of the code and run it to check if the code is correct or if the code fails

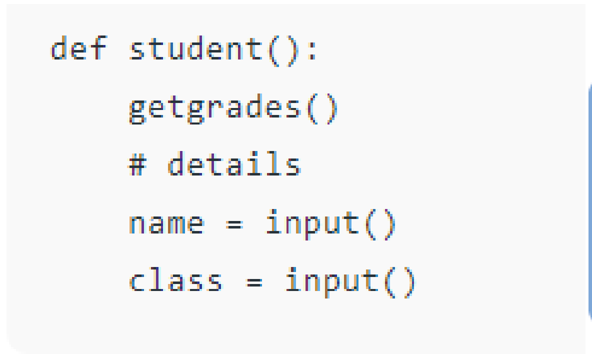
Step 3: Update the functional code – To make the new test pass

Step 4: Run your tests again – If they fail, update the functional code and retest

Step 5: Refactor – Once the test is passed, the next step is to start over

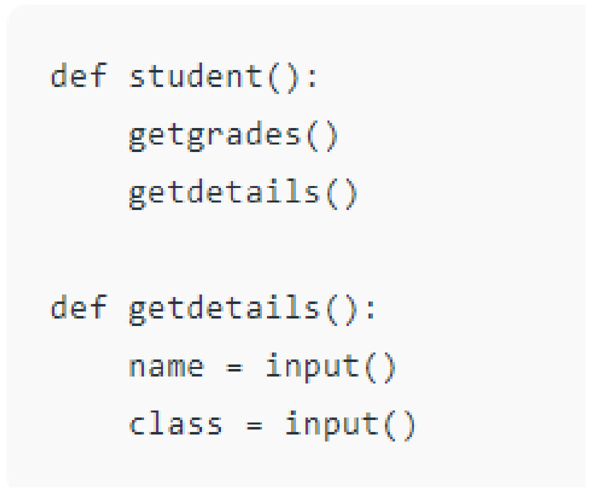


The code refactoring process

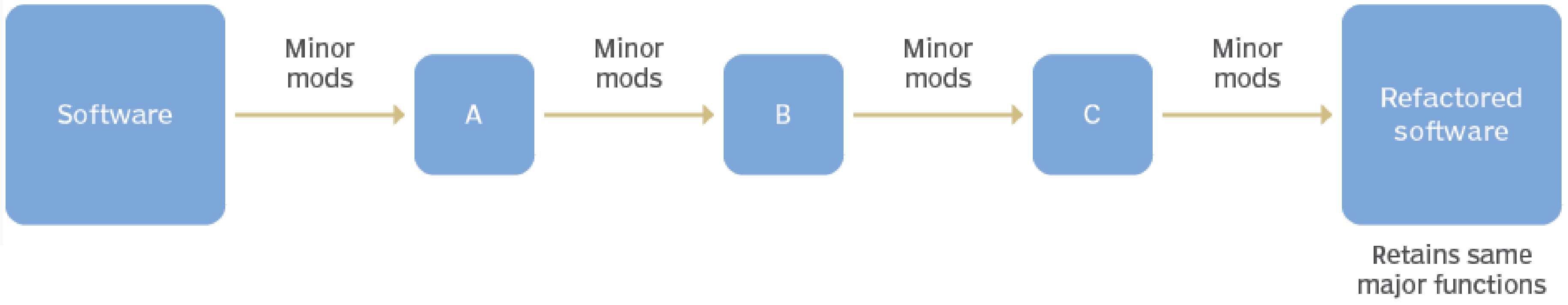


```
def student():
    getgrades()
    # details
    name = input()
    class = input()
```

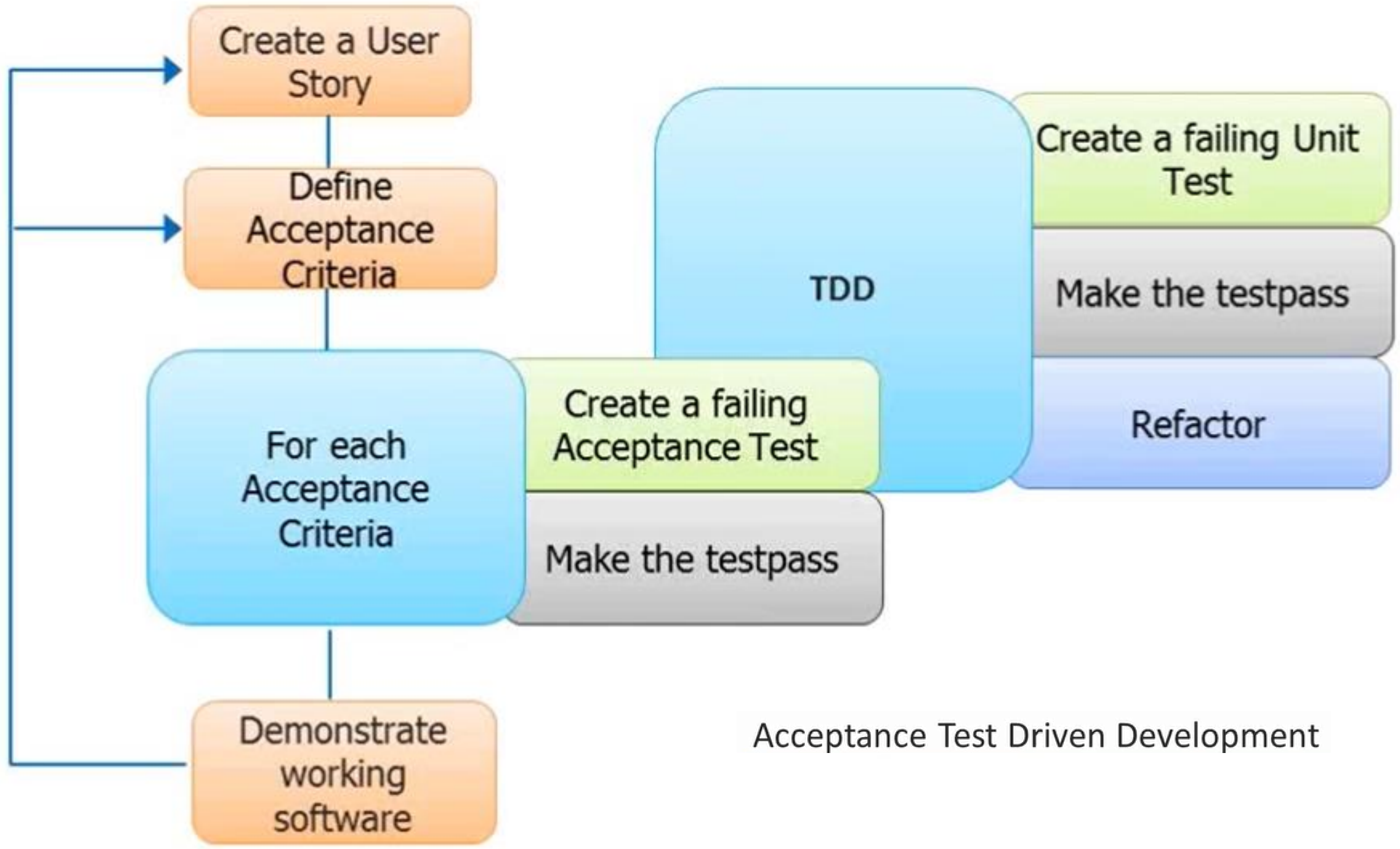
1. This could be refactored as:



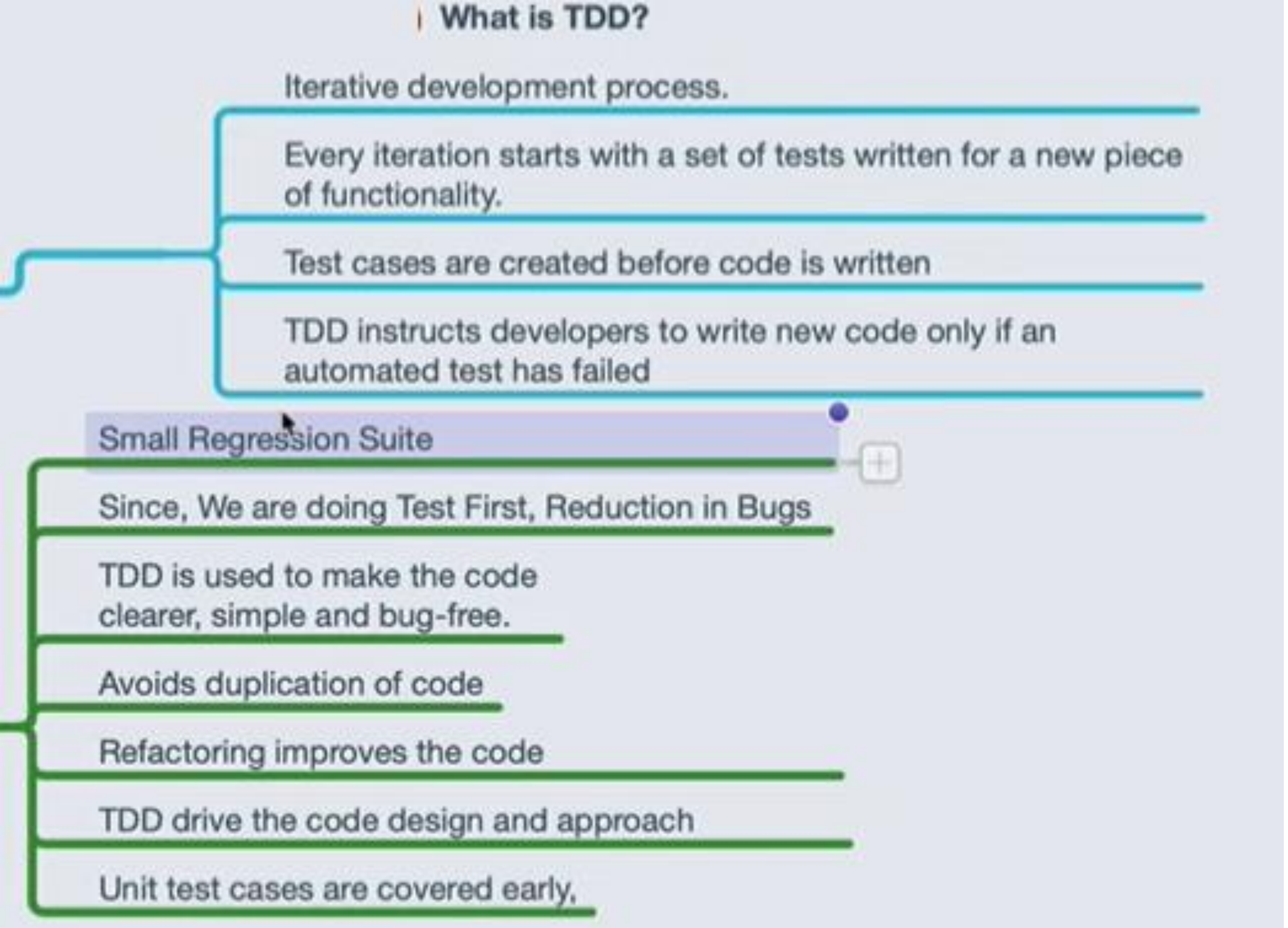
```
def student():
    getgrades()
    getdetails()

def getdetails():
    name = input()
    class = input()
```



Refactoring or **Code Refactoring** is defined as systematic process of improving existing computer code, without adding new functionality or changing external behaviour of the code


Create a User Story
Define Acceptance Criteria
For each Acceptance Criteria
Create a failing Acceptance Test
Make the testpass
TDD
Create a failing Unit Test
Make the testpass
Refactor
Demonstrate working software

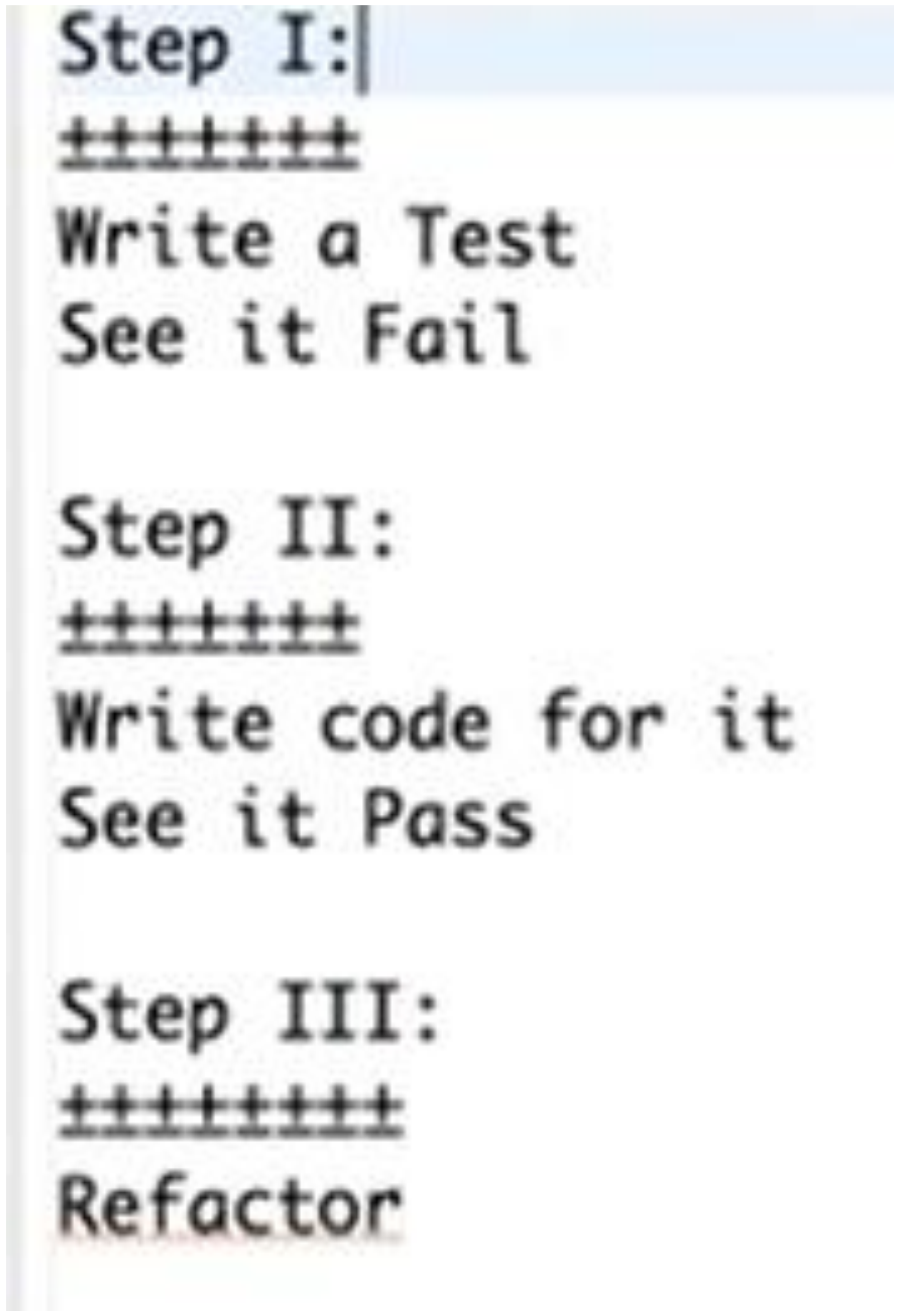
Acceptance Test Driven Development



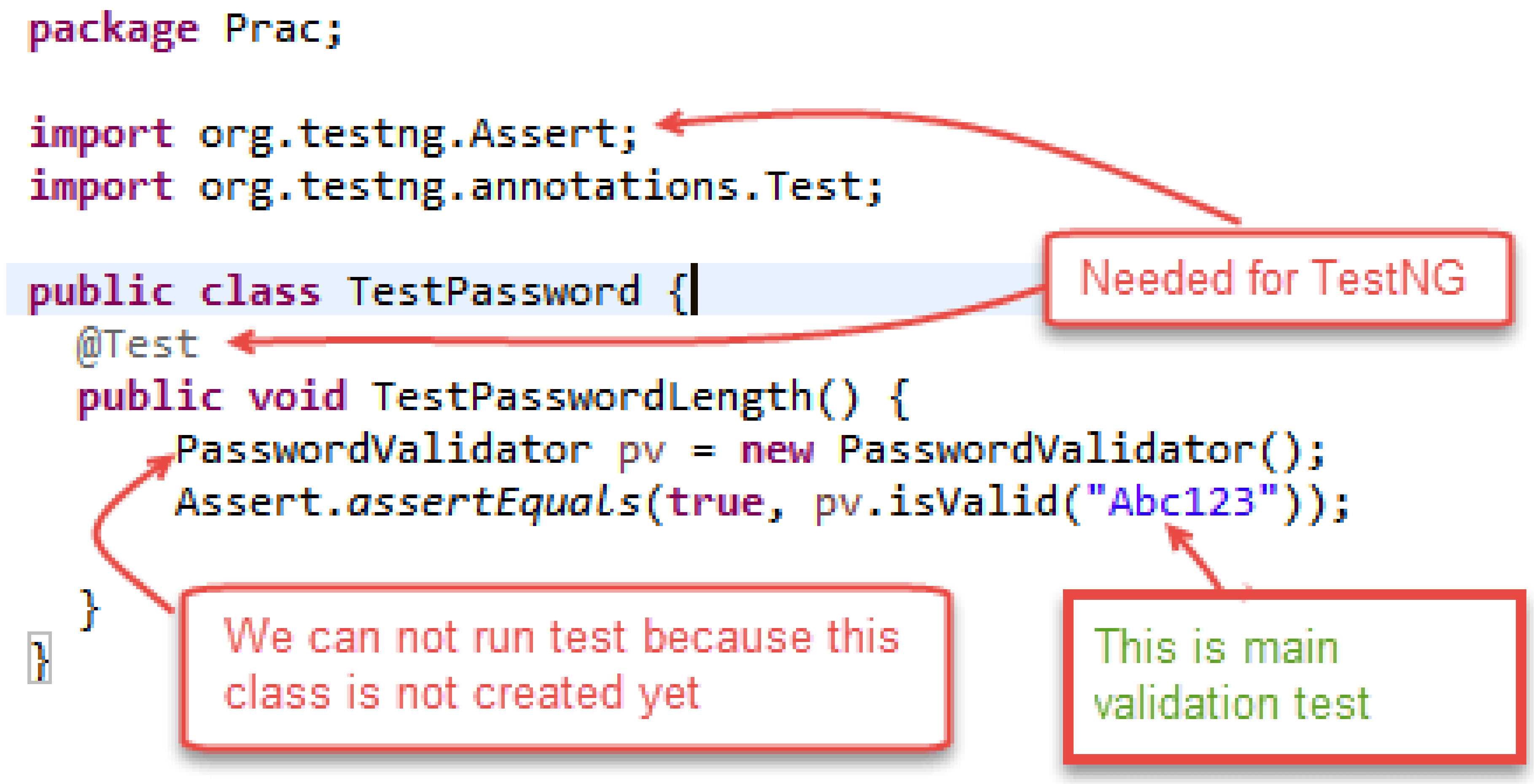
What is TDD?

- Iterative development process.
- Every iteration starts with a set of tests written for a new piece of functionality.
- Test cases are created before code is written
- TDD instructs developers to write new code only if an automated test has failed

- Small Regression Suite
- Since, We are doing Test First, Reduction in Bugs
- TDD is used to make the code clearer, simple and bug-free.
- Avoids duplication of code
- Refactoring improves the code
- TDD drive the code design and approach
- Unit test cases are covered early,



```
Step I:
+++++++
Write a Test
See it Fail

Step II:
+++++++
Write code for it
See it Pass

Step III:
++++++++
Refactor
```

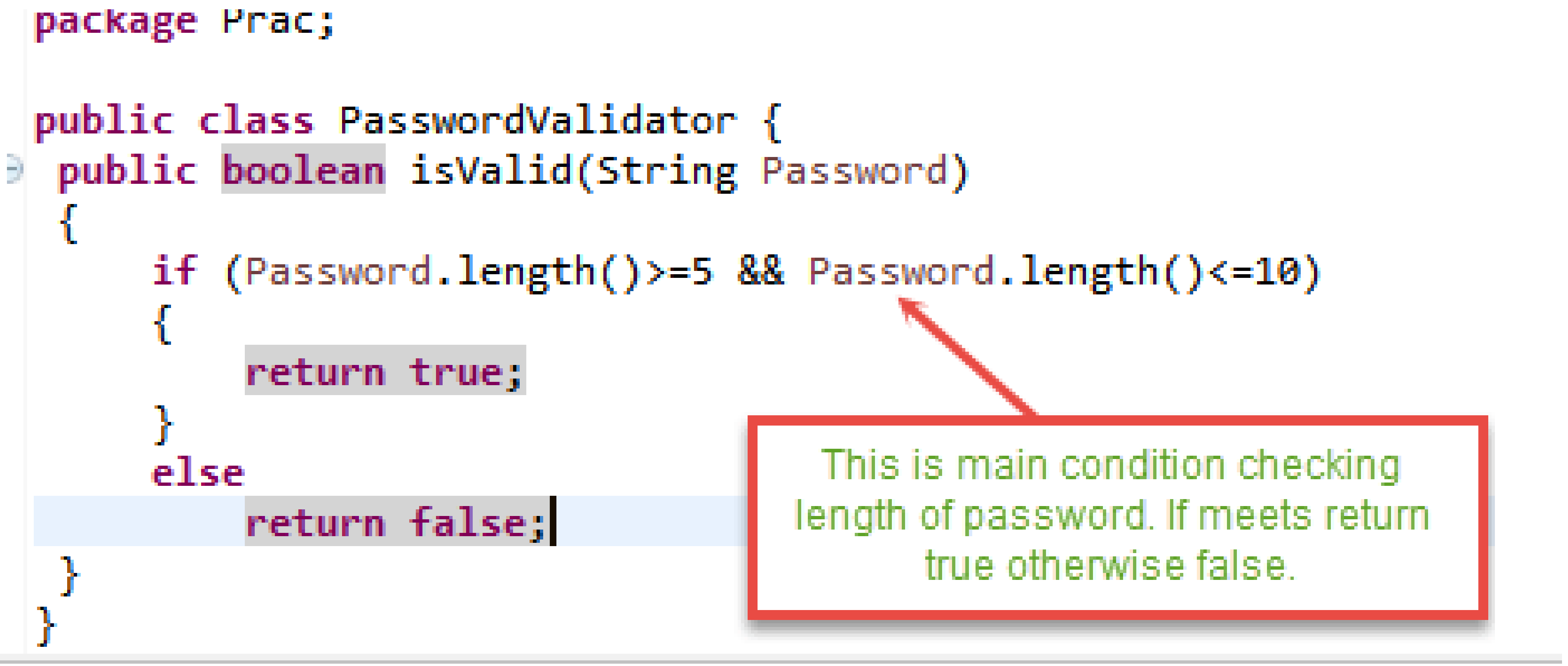


```
package Prac;

import org.testng.Assert;
import org.testng.annotations.Test;

public class TestPassword {
  @Test
  public void TestPasswordLength() {
      PasswordValidator pv = new PasswordValidator();
      Assert.assertEquals(true, pv.isValid("Abc123"));

  }
}
```

Needed for TestNG

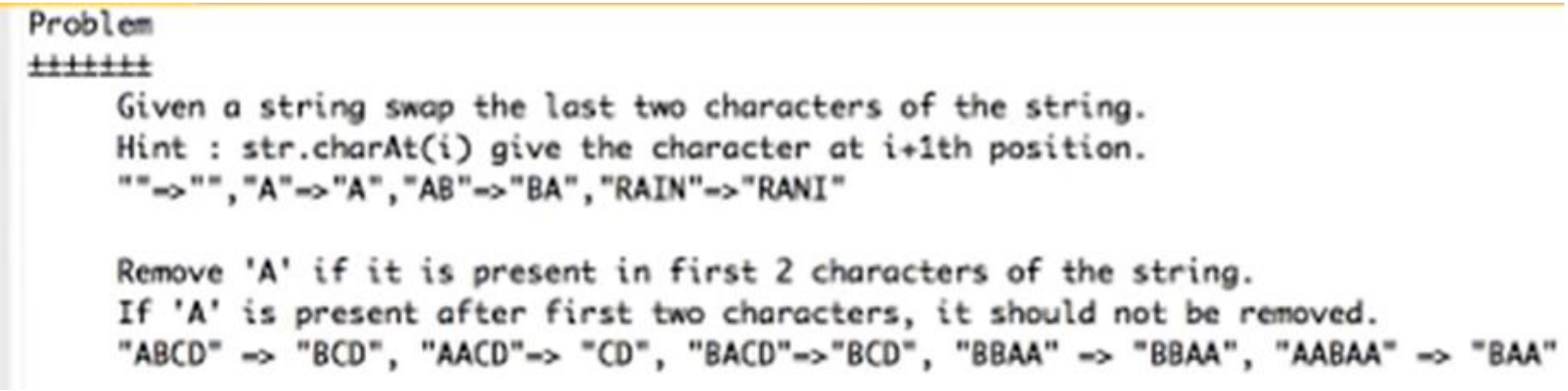
We can not run test because this class is not created yet

This is main validation test

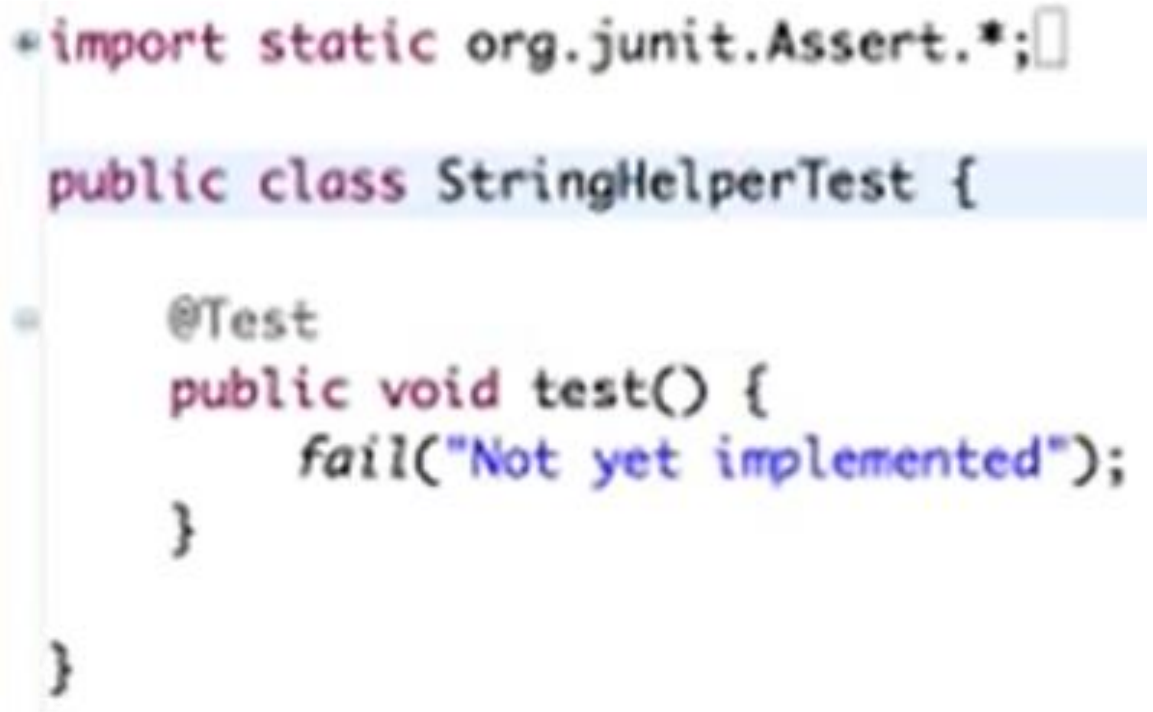


```
package Prac;

public class PasswordValidator {
 public boolean isValid(String Password)
 {
     if (Password.length()>=5 && Password.length()<=10)
     {
         return true;
     }
     else
         return false;
 }
}
```

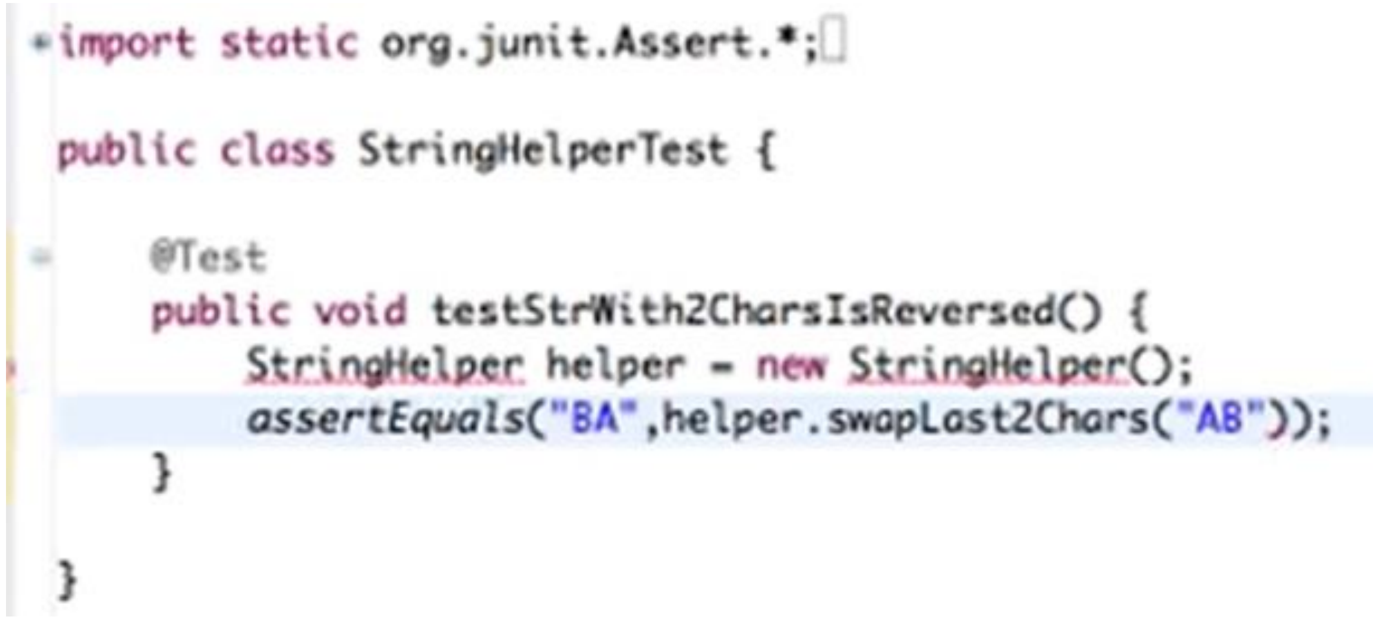
This is main condition checking length of password. If meets return true otherwise false.



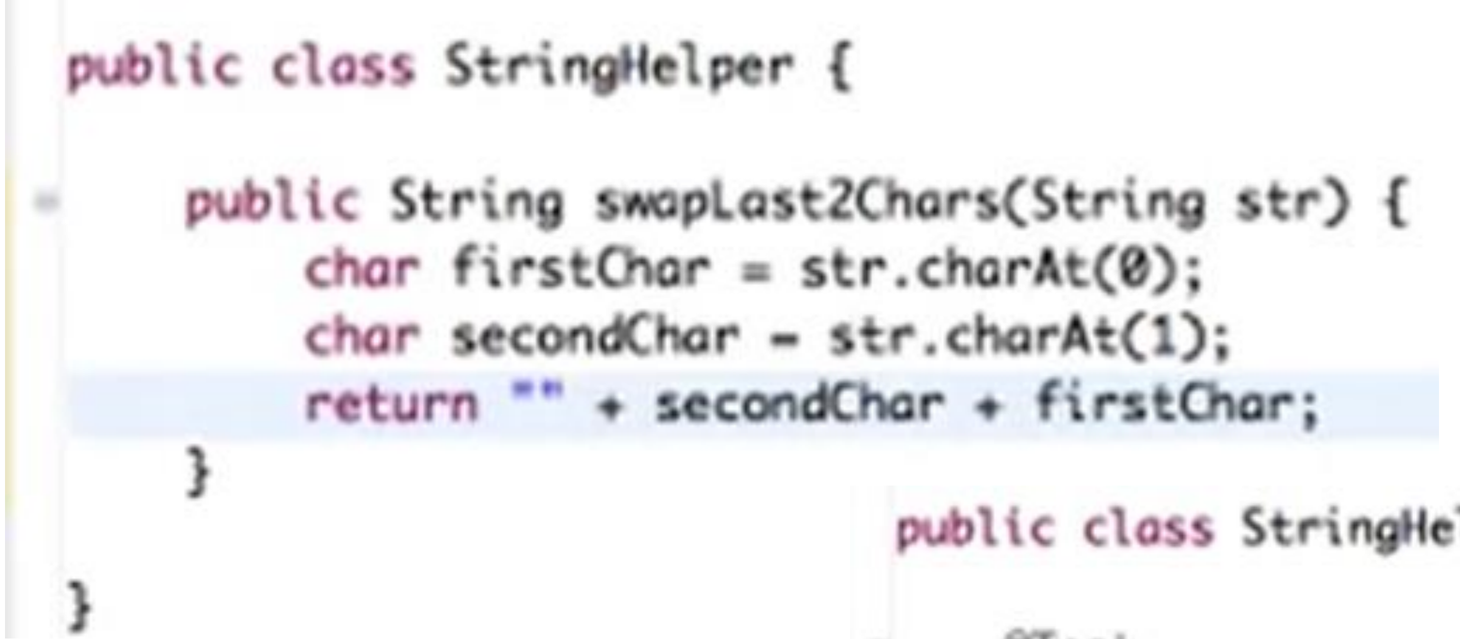
```
Problem
+++++++
        Given a string swap the last two characters of the string.
        Hint : str.charAt(i) give the character at i+1th position.
        ""->"","A"->"A","AB"->"BA","RAIN"->"RANI"

        Remove 'A' if it is present in first 2 characters of the string.
        If 'A' is present after first two characters, it should not be removed.
        "ABCD" -> "BCD", "AACD"-> "CD", "BACD"->"BCD", "BBAA" -> "BBAA", "AABAA" -> "BAA"
```



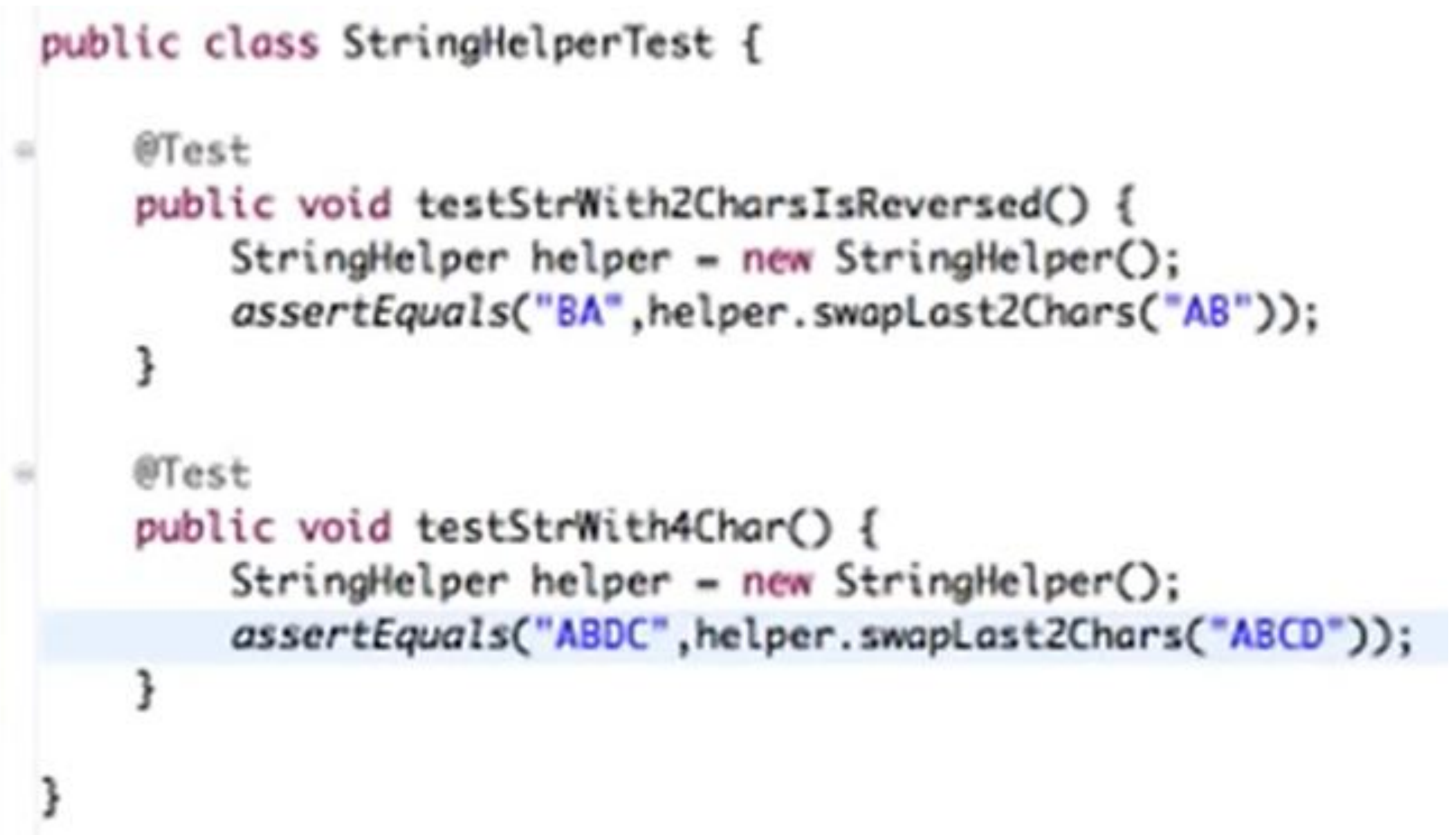
```java
import static org.junit.Assert.*;

public class StringHelperTest {

    @Test
    public void test() {
        fail("Not yet implemented");
    }

}
```



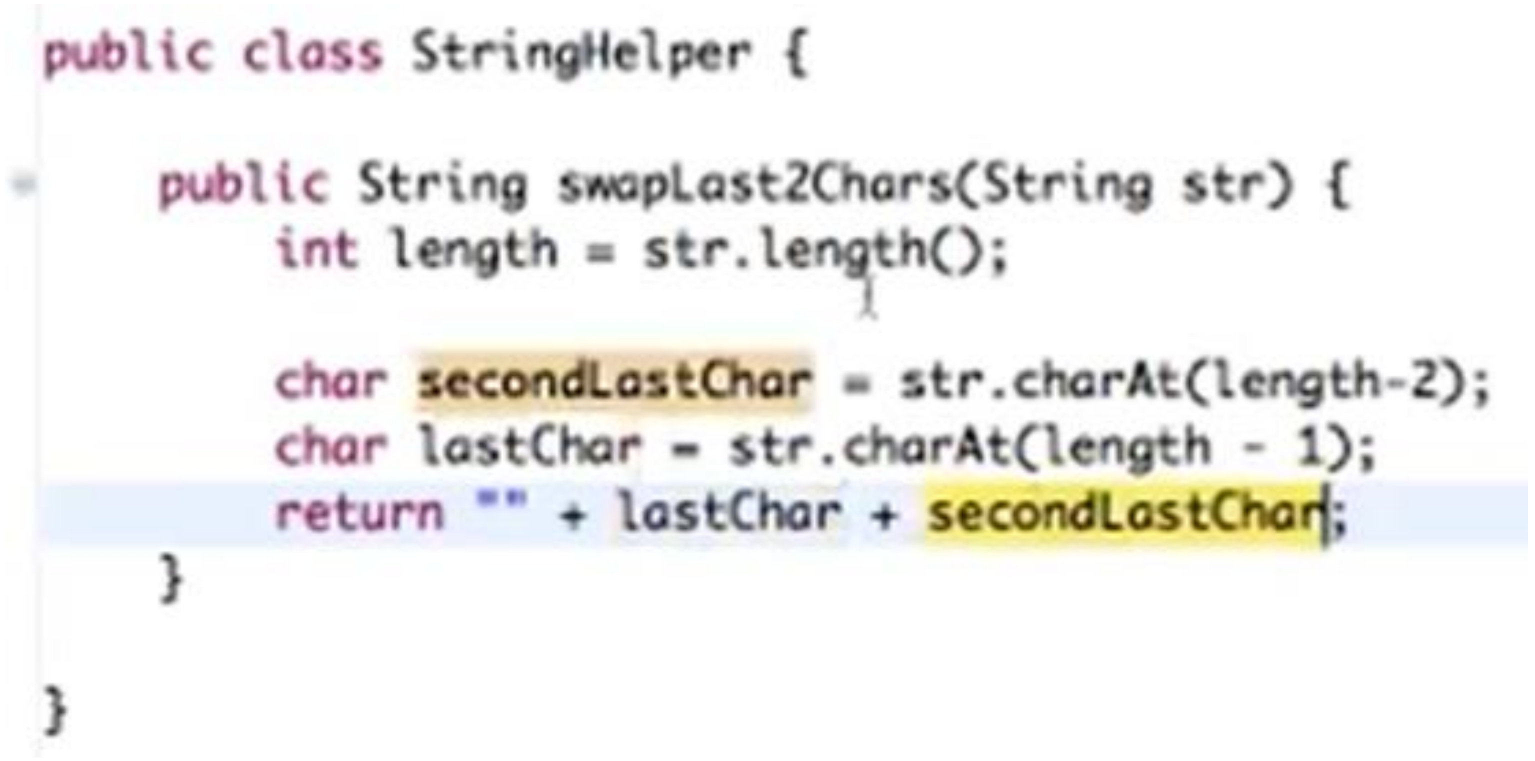
```java
import static org.junit.Assert.*;

public class StringHelperTest {

    @Test
    public void testStrWith2CharsIsReversed() {
        StringHelper helper = new StringHelper();
        assertEquals("BA",helper.swapLast2Chars("AB"));
    }

}
```



```java
public class StringHelper {

    public String swapLast2Chars(String str) {
        char firstChar = str.charAt(0);
        char secondChar = str.charAt(1);
        return "" + secondChar + firstChar;
    }

}
```



```java
public class StringHelperTest {

    @Test
    public void testStrWith2CharsIsReversed() {
        StringHelper helper = new StringHelper();
        assertEquals("BA",helper.swapLast2Chars("AB"));
    }

    @Test
    public void testStrWith4Char() {
        StringHelper helper = new StringHelper();
        assertEquals("ABDC",helper.swapLast2Chars("ABCD"));
    }

}
```



```java
public class StringHelper {

    public String swapLast2Chars(String str) {
        int length = str.length();

        char secondLastChar = str.charAt(length-2);
        char lastChar = str.charAt(length - 1);
        return "" + lastChar + secondLastChar;
    }

}
```