華中科技大學

课 程 实 验 报 告

课程名称： 软件工程理论课实验

专业班级： 软件 2202 班

学　　号： U202217216

姓　　名： 郑德凯

报告日期： 2024/4/8

软件学院

目录

一、实验目的、内容和要求

1.1 实验名称

- TDD 测试驱动开发实验

1.2 实验目的

- 理解 TDD 测试驱动开发的思想
- 掌握 TDD 测试驱动开发的基本过程。

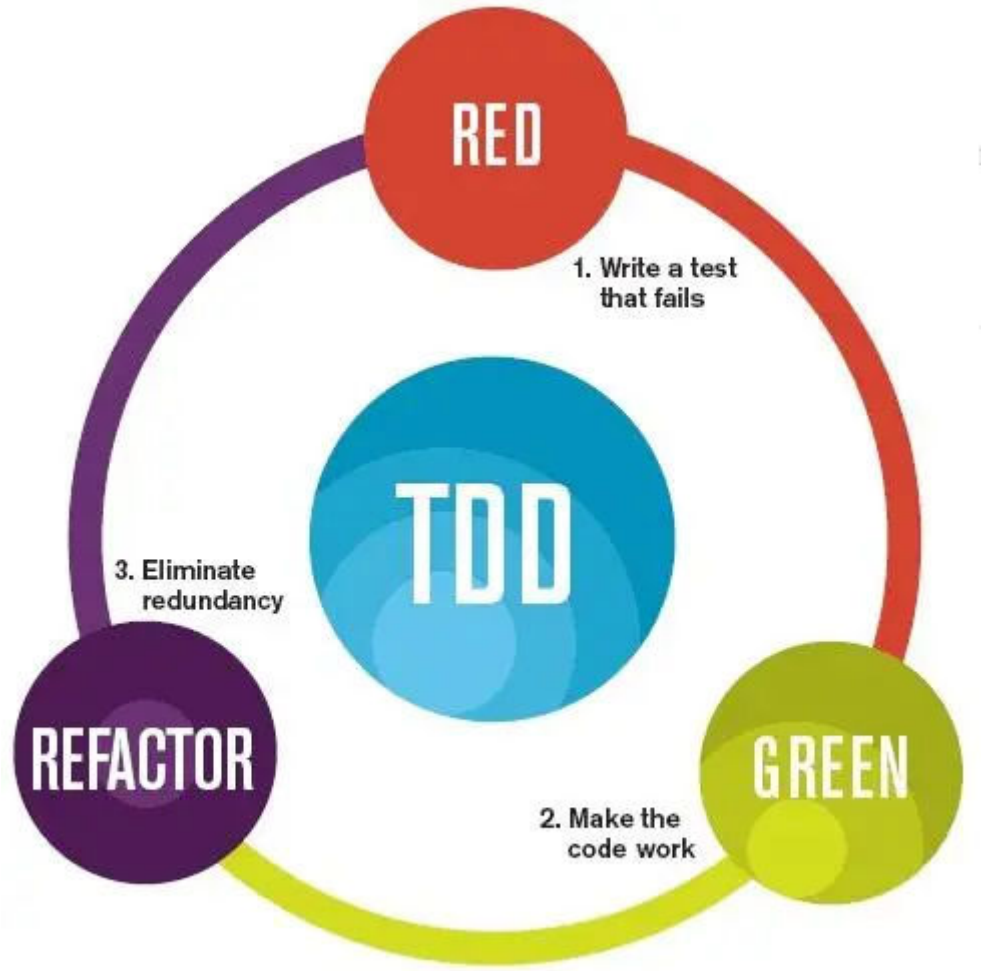
1.3 实验内容和要求

- 阅读：Kent Beck 《测试驱动开发》

- 基于 TDD 开发一个判断字符串是 IP4 地址的功能
- 使用任何 OO 语言
- 使用 xUnit 测试框架

二、实验背景知识

2.1 TDD 测试驱动开发



The mantra of Test-Driven Development (TDD) is "red, green, refactor."

TDD 的基本流程是：红，绿，重构。

更详细的流程是：

- 写一个测试用例
- 运行测试
- 写刚好能让测试通过的实现
- 运行测试

- 识别坏味道，用手法修改代码
- 运行测试

2.2ipv4 地址有效性

IPv4 地址有效性的判断主要包括以下几个方面：

- 格式正确性：IPv4 地址由四个十进制数字组成，每个数字的范围在 0 到 255 之间，用点号分隔。例如，192.168.1.1 就是一个有效的 IPv4 地址，而 192.168.1.256 则是无效的。
- 数字范围：IPv4 地址中每个数字的范围应该在 0 到 255 之间，且不能以 0 开头，除非该数字是 0 本身。例如，192.168.1.1 是有效的，而 192.168.01.1 是无效的。
- 四个数字：IPv4 地址应该由四个数字组成，用点号分隔。例如，192.168.1.1 是有效的，而 192.168.1 是无效的。
- 不允许其他字符：IPv4 地址中只允许数字和点号，不允许其他字符出现。例如，192.168.1.1 是有效的，而 192.168.1.a 是无效的。

2.3xUnit 测试框架

xUnit 是一种通用的测试框架家族，用于各种编程语言的单元测试。它是从 Smalltalk 的 SUnit 框架演变而来，其中的“x”代表了各种不同的编程语言（例如 JUnit 用于 Java、PHPUnit 用于 PHP、PyTest 用于 Python 等）。xUnit 框架遵循一组基本原则和设计模式，使得编写和运行单元测试变得更加简单和一致。

- xUnit 测试框架通常提供以下基本功能：
- 测试用例：编写单元测试的基本单位。每个测试用例描述了对代码的一个特定方面进行测试的情况。测试用例通常是一个类或者一个函数，其中包含了测试代码和断言。
- 测试套件：用于组织和运行多个测试用例的集合。测试套件可以包含多个测试用例，并提供统一的接口来运行这些测试用例。
- 断言：用于验证代码行为的预期结果。断言通常是在测试用例中使用的一种方法，用于比较实际结果和期望结果是否一致。
- 测试运行器：用于执行测试套件并生成测试结果的组件。测试运行器负责加载测试用例、运行测试用例、收集测试结果，并生成测试报告。

在本次实验中，我选用的是基于 JUnit 测试框架的 unittest 测试框架

三、TDD 测试驱动开发

3.1 编写测试用例

首先，我们编写测试用例：

```
# test.py

import unittest

from my_fun  import is_ipv4_address


class TestIsIPv4Address(unittest.TestCase):


    def test_valid_ipv4_addresses(self):

        self.assertTrue(is_ipv4_address("192.168.1.1"))

        self.assertTrue(is_ipv4_address("0.0.0.0"))

        self.assertTrue(is_ipv4_address("255.255.255.255"))


    def test_invalid_ipv4_addresses(self):

        self.assertFalse(is_ipv4_address("256.256.256.256"))

        self.assertFalse(is_ipv4_address("192.168.1.256"))

        self.assertFalse(is_ipv4_address("192.168.1"))

        self.assertFalse(is_ipv4_address("192.168.1.1.1"))

        self.assertFalse(is_ipv4_address("192.168..1"))

        self.assertFalse(is_ipv4_address("192.168.1.a"))


# my_fun.py

def is_ipv4_address(address):

    pass
```

3.2 运行测试

由于功能代码 funtion.py 尚未编写，故运行失败：

```python
# test.py

import unittest
from my_fun import is_ipv4_address

class TestIsIPv4Address(unittest.TestCase):

    def test_valid_ipv4_addresses(self):
        self.assertTrue(is_ipv4_address("192.168.1.1"))
        self.assertTrue(is_ipv4_address("0.0.0.0"))
        self.assertTrue(is_ipv4_address("255.255.255.255"))

    def test_invalid_ipv4_addresses(self):
        self.assertFalse(is_ipv4_address("256.256.256.256"))
        self.assertFalse(is_ipv4_address("192.168.1.256"))
        self.assertFalse(is_ipv4_address("192.168.1"))
        self.assertFalse(is_ipv4_address("192.168.1.1.1"))
        self.assertFalse(is_ipv4_address("192.168..1"))
        self.assertFalse(is_ipv4_address("192.168.1.a"))

if __name__ == "__main__":
    unittest.main()
```

问题 输出 调试控制台 终端 端口

```
    self.assertTrue(is_ipv4_address("192.168.1.1"))
AssertionError: None is not true

----------------------------------------------------------------------
Ran 2 tests in 0.000s

FAILED (failures=1)
```

3.3 编写功能代码

```
def is_ipv4_address(address):
    parts = address.split('.')
    if len(parts) != 4:
        return False
    for part in parts:
        if not part.isdigit() or not 0 <= int(part) <= 255:
            return False
    return True
```

3.4 运行测试

目前完整代码如下：

```
# test.py
```

```
import unittest
from my_fun import is_ipv4_address

class TestIsIPv4Address(unittest.TestCase):

    def test_valid_ipv4_addresses(self):
        self.assertTrue(is_ipv4_address("10.16.204.66"))
        self.assertTrue(is_ipv4_address("0.0.0.0"))
        self.assertTrue(is_ipv4_address("255.255.255.255"))

    def test_invalid_ipv4_addresses(self):
        self.assertFalse(is_ipv4_address("256.256.256.256"))
        self.assertFalse(is_ipv4_address("192.168.1.256"))
        self.assertFalse(is_ipv4_address("192.168.01.255"))
        self.assertFalse(is_ipv4_address("192.168.1.1.1"))
        self.assertFalse(is_ipv4_address("192.168..1"))
        self.assertFalse(is_ipv4_address("192.168.1.a"))

if __name__ == "__main__":
    unittest.main()

#my_fun.py

def is_ipv4_address(address):
    parts = address.split('.')
    if len(parts) != 4:
        return False
    for part in parts:
        if not part.isdigit() or not 0 <= int(part) <= 255:
```

return False

return True

运行结果如下：

```
test.py

import unittest
from my_fun import is_ipv4_address

class TestIsIPv4Address(unittest.TestCase):

    def test_valid_ipv4_addresses(self):
        self.assertTrue(is_ipv4_address("192.168.1.1"))
        self.assertTrue(is_ipv4_address("0.0.0.0"))
        self.assertTrue(is_ipv4_address("255.255.255.255"))

    def test_invalid_ipv4_addresses(self):
        self.assertFalse(is_ipv4_address("256.256.256.256"))
        self.assertFalse(is_ipv4_address("192.168.1.256"))
        self.assertFalse(is_ipv4_address("192.168.1"))
        self.assertFalse(is_ipv4_address("192.168.1.1.1"))
        self.assertFalse(is_ipv4_address("192.168..1"))
        self.assertFalse(is_ipv4_address("192.168.1.a"))

if __name__ == "__main__":
    unittest.main()
```

问题　输出　调试控制台　终端　端口

```
FAILED (failures=1)
PS D:\test1> & C:/Users/郑德凯/AppData/Local/Programs/Python/Python311/python.exe d:/test1/1.py
..
----------------------------------------------------------------------
Ran 2 tests in 0.000s

OK
```

四、实验总结

通过这次实验，我深刻体会到了测试驱动开发（TDD）的方法。TDD 不仅可以帮助我们编写更加可靠的代码，还可以提高代码的质量和可维护性。在实践中，我学会了如何编写简洁清晰的测试用例，并根据测试用例逐步编写功能代码。这种迭代式的开发过程使得我们可以更快地发现和修复问题，提高了开发效率。我相信这种经验将对我未来的学习和工作有很大的帮助。