Kiểm thử đơn vị - Unit Testing

BM. Công nghệ phần mềm
Khoa. Công nghệ thông tin

KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

Nội dung

1. **Mở đầu**
2. Giới thiệu
3. Tại sao cần kiểm thử đơn vị
4. Unit testing framework
5. Các bước thực hiện
6. Một số thuật ngữ
7. Mock Object
8. Demo

1. Mở đầu

Hàm tính khoảng cách

Hàm khởi tạo

???

Lớp **Diem**

Hàm tìm kiếm

Hàm tính toán

???

Hàm khởi tạo

Lớp …

Hàm tính chu vi

Lớp **TamGiac**

Hàm tính diện tích

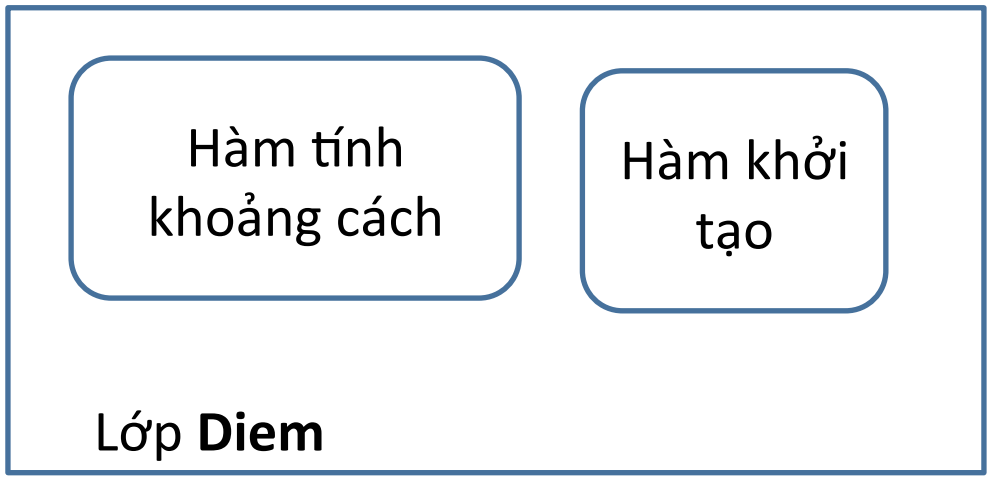
???

Hàm khởi tạo

Hàm đúng hay sai???

1. Mở đầu

- Áp dụng các kỹ thuật => các **test case**
- Thực thi các test case



Hàm tính khoảng cách
Thực thi test case 1
Thực thi test case 2
…
Hàm khởi tạo
Thực thi test case 1
Thực thi test case 2
…

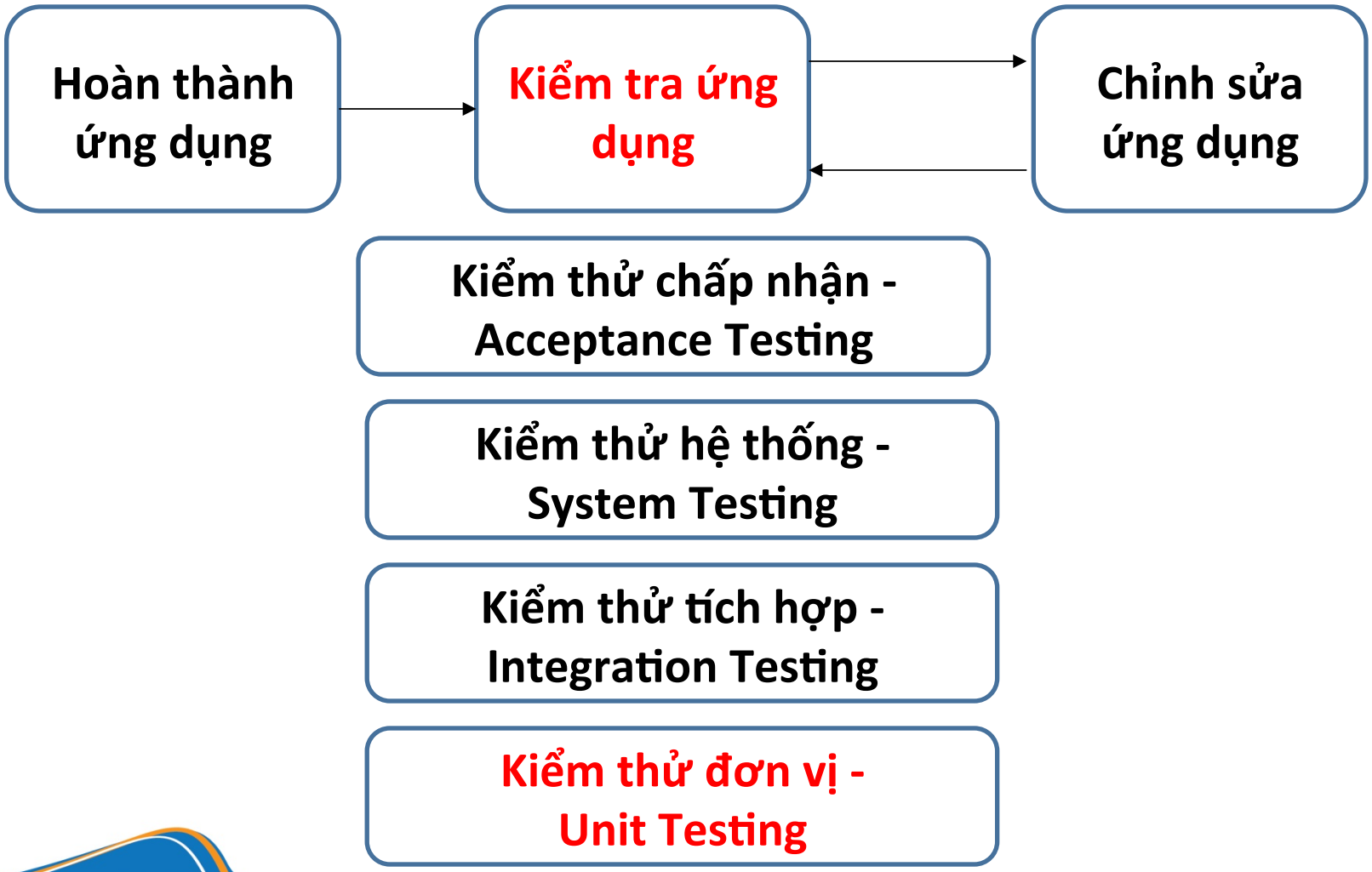
Tương tự cho lớp **TamGiac**, …

Chương trình có **nhiều lớp**, mỗi lớp **nhiều hàm** => thực thi **bao nhiêu lần**???

1. Mở đầu

- Viết code ???
- Lớp **Điểm**, hàm **tính khoảng cách**
  - Hàm thực thi test case 1:
    - Khởi tạo điểm 1 => (0, 0)
    - Khởi tạo điểm 2 => (0, 2)
    - **Gọi hàm tính khoảng cách** giữa điểm 1 và điểm 2 => **kết quả thực tế**
    - **Kết quả mong đợi** = 2
    - So sánh **kết quả mong đợi** và **kết quả thực tế**
    - Tùy vào kết quả so sánh, xuất ra **pass**, **fail**, **error**
  - Tương tự cho các **test case khác** …
- Tương tự cho **lớp** và **hàm khác** …

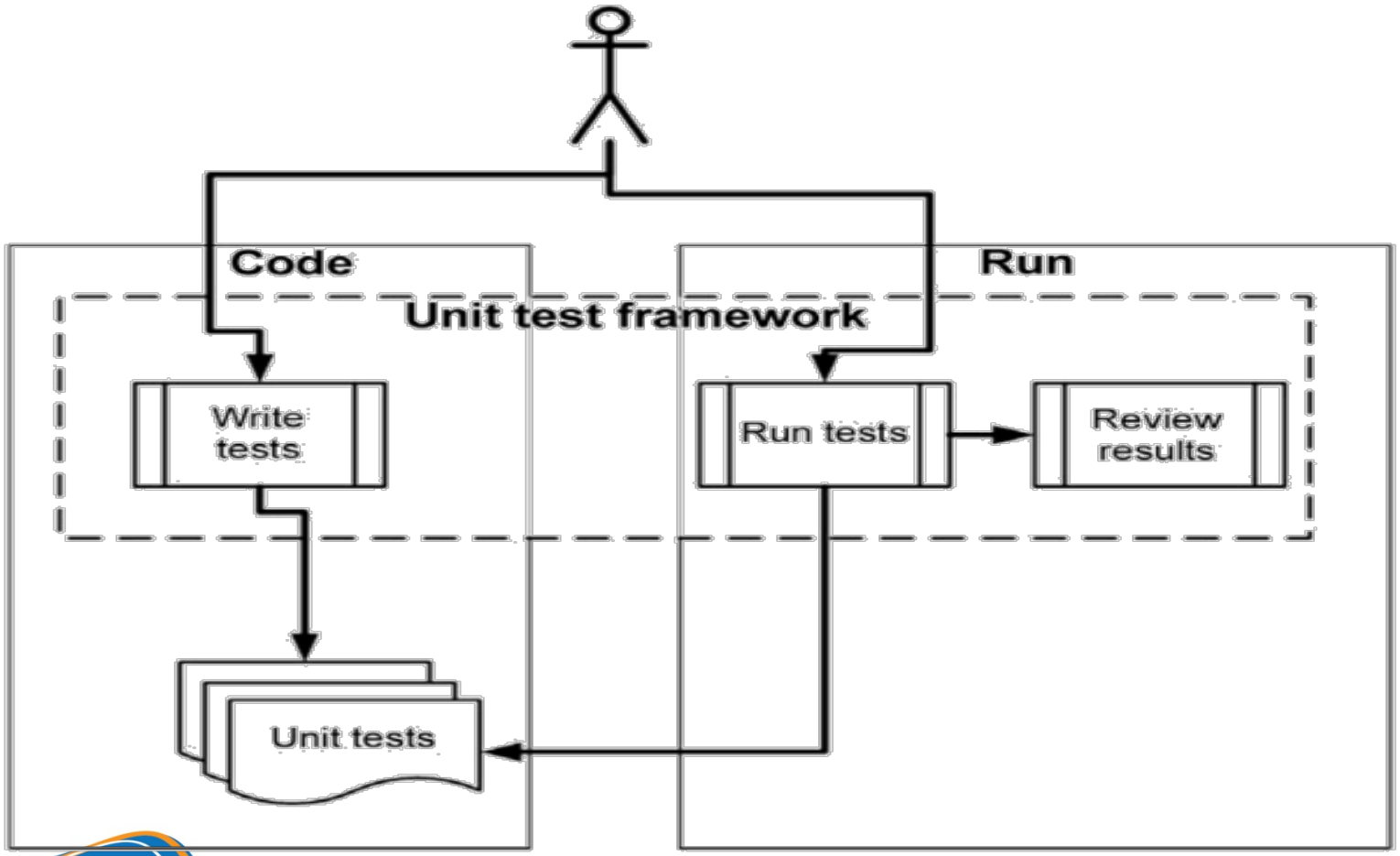
2. Giới thiệu


Hoàn thành ứng dụng
Kiểm tra ứng dụng
Chỉnh sửa ứng dụng
Kiểm thử chấp nhận - Acceptance Testing
Kiểm thử hệ thống - System Testing
Kiểm thử tích hợp - Integration Testing
Kiểm thử đơn vị - Unit Testing

3. Tại sao cần kiểm thử đơn vị

- Đảm bảo chất lượng từng đơn vị mã nguồn trong phần mềm
- Phát hiện lỗi sớm và chỉnh sửa kịp thời
- …

4. Unit testing framework



4. Unit testing framework

- Thư viện hỗ trợ - Script
- Dữ liệu - Data driven
- Thực thi - Run
- Thống kê - Report
- Các framework hỗ trợ cho từng ngôn ngữ lập trình
  - JUnit - java
  - NUnit - .NET
  - CPPUnit - C++
  - PyUnit - Python
  - …
  - **xUnit**

5. Các bước thực hiện

Hàm tính khoảng cách

???

Hàm Kiểm tra …

Hàm tính diện tích

???

Thực thi hàm kiểm tra

Hàm tính chu vi

???

Báo cáo

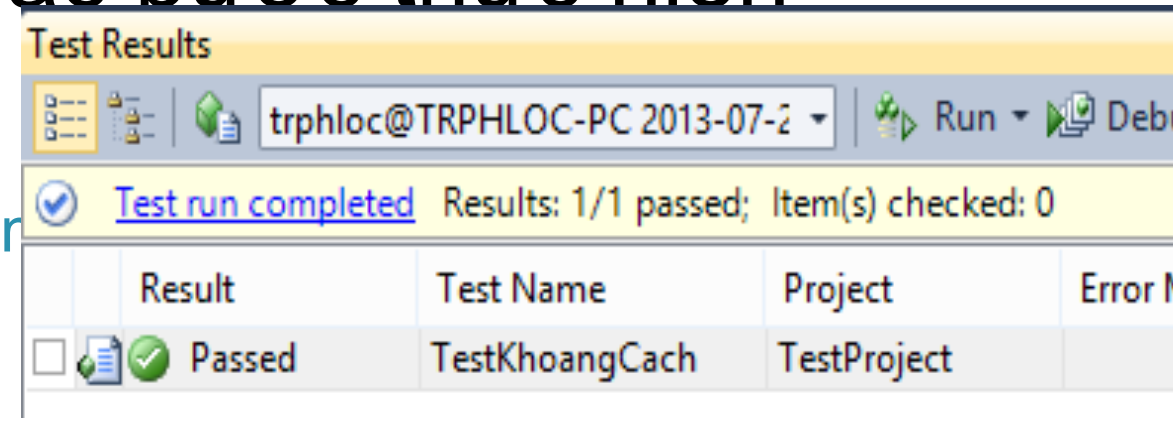
Hàm tìm kiếm

???

5. Các bước thực hiện

- Hàm thực thi test case 1 của hàm tính khoảng cách:
  - Khởi tạo điểm 1 => (0, 0)
  - Khởi tạo điểm 2 => (0, 2)
  - **Gọi hàm tính khoảng cách** giữa điểm 1 và điểm 2 => **kết quả thực tế**
  - **Kết quả mong đợi** = 2
  - So sánh **kết quả mong đợi** và **kết quả thực tế**
  - Tùy vào kết quả so sánh, xuất ra **pass**, **fail**, **error**

5. Các bước thực hiện




```
[TestClass]
    public class TestDier
    {
        [TestMethod]
        public void TestKhoangCach()
        {
            Diem d1 = new Diem(0, 0);
            Diem d2 = new Diem(0, 2);
            double KQThucTe = d1.KhoangCach(d2);
            double KQMongDoi = 2;
            Assert.AreEqual(KQMongDoi, KQThucTe);
        }
    }
```

6. Một số thuật ngữ

- Kết quả của hàm kiểm thử
- So sánh: **Assert**
- Hàm khởi tạo và hủy
- Dữ liệu – data driven

6.1. Kết quả của hàm kiểm thử

- Pass
- Fail
- Error

6.2. So sánh: assert

- Điều kiện đúng/sai (boolean)

  isTrue(dieukien)

  isFalse(dieukien)

- Đối tượng tồn tại (NULL)

  isNull(doituong)

  isNotNull(doituong)

6.2. So sánh: assert

- Đối tượng giống nhau (same)

  AreSame(MongDoi, ThucTe)

  AreNotSame(MongDoi, ThucTe)

- Đối tượng bằng (equal)

  AreEqual(MongDoi, ThucTe)

  AreNoteEqual(MongDoi, ThucTe)

Hocsinh 1 & hoc sinh 2
Nhanvien 1 & nhan vien 2

- Số thực (sai số)

  AreEqual(MongDoi, ThucTe, saiso)

6.2. So sánh: assert

- Mảng
  - Số phần tử bằng nhau
  - Mỗi phần tử bằng nhau

- Kiểm thử ngoại lệ

```
try
{
    exceptionCausingMethod();
    // If this point is reached, the expected
    // exception was not thrown.
    fail("Exception should have occurred");
}
catch ( ExceptedTypeOfException exc )
{
    String expected = "A suitable error message";
    String actual = exc.getMessage();
    Assert.assertEquals( expected, actual );
}
```

6.3. Hàm khởi tạo và hủy

- Chạy mỗi lần bắt đầu và kết thúc **hàm**
- Chạy mỗi lần bắt đầu và kết thúc **quá trình kiểm thử**
- Thứ tự gọi hàm
  - Hàm **khởi tạo lớp**
  - Hàm **khởi tạo**
  - Hàm kiểm tra của hàm tính chu vi
  - Hàm **hủy**
  - Hàm **khởi tạo**
  - Hàm kiểm tra của hàm tính diện tích
  - Hàm **hủy**
  - Hàm **hủy lớp**

6.4. Data driven

- Đọc dữ liệu trong quá trình kiểm thử.
- Thực thi nhiều test case giống nhau.

6.4. Data driven

```
public class TestDiem
{
    [TestMethod]
    [DeploymentItem("FPNWIND.MDB")]
    [DataSource("System.Data.OleDb", "Provider=Microsoft.Jet.OLEDB.4.0;Data Source
    public void TestMethod()
    {
        Console.WriteLine("EmployeeID: {0}, LastName: {1}",
            TestContext.DataRow["EmployeeID"], TestContext.DataRow["LastName"]);
    }
```

7. Mock object

- Hàm thực thi test case 1 của hàm tính khoảng cách:
  - **Khởi tạo điểm 1 => (0, 0)**
  - **Khởi tạo điểm 2 => (0, 2)**
  - Gọi hàm tính khoảng cách giữa điểm 1 và điểm 2 => kết quả thực tế
  - Kết quả mong đợi = 2
  - So sánh kết quả mong đợi và kết quả thực tế
  - Tùy vào kết quả so sánh, xuất ra pass, fail, error

=> Khó khăn trong quá trình khởi tạo đối tượng

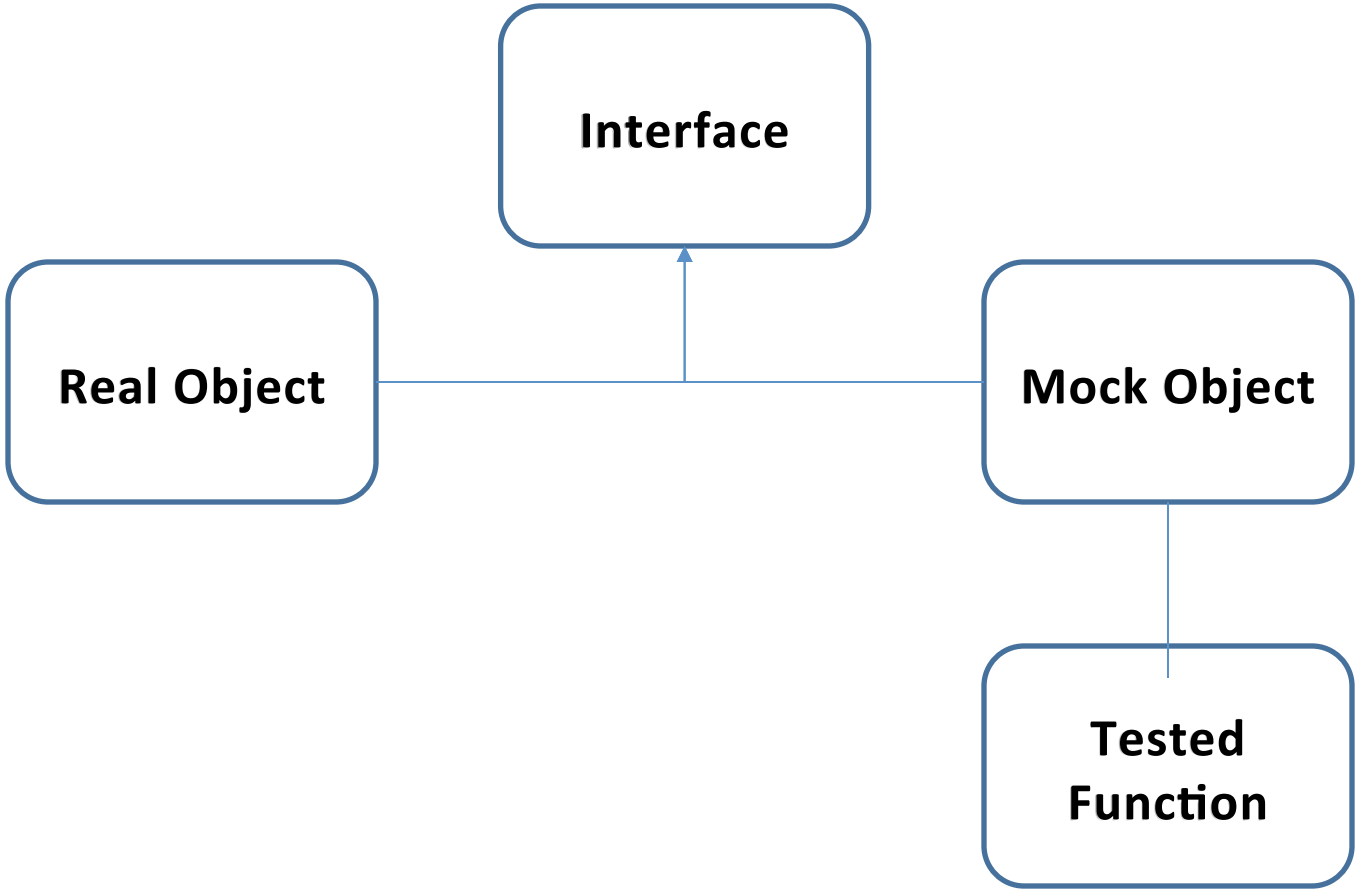
7. Mock object

- Mock object (MO): là đối tượng ảo
  - Mô phỏng các tính chất và hành vi giống hết như đối tượng thực.
- Truyền vào lệnh đang vận hành nhằm kiểm tra tính đúng đắn của các hoạt động bên trong.

7.1. Thiết kế Mock Object

1. Đưa ra interface để mô tả đối tượng.
2. Viết nội dung cho đối tượng thực dựa trên interface.
3. Trích interface từ đối tượng thật và triển khai Mock Object (MO) dựa trên interface đó

7.1. Thiết kế Mock Object



7.2 Cài đặt và sử dụng

- Thêm tham chiếu tới thư viện chứa lớp Mock Object
- Ví dụ trong Nunit là: nuint.mocks

Demo

- NUnit
- JUnit

Thảo luận