System design, software integration and testing
Design documents

2018.01.30

Contents

1 Document format

1.1 System design description

We need to design our system, from the current functional block, to the class of the code level, the relationship and the description of the use case. Therefore, we can refer to the UML draw class diagram and use case diagram, everyone in the program before, to think about this program I really need what classes, each class will have what method, using UML to draw the design, and UML can be derived Java code generation. The following diagram is a few of the most basic steps that the system design description needs to be considered, and can be seen to learn.

Here is **five steps** of **system design description**.

1 system introduction

2 system module decomposition: according to the system design, the system module decomposition diagram is given to show the responsibilities of each module and the cooperation between modules. (for individual Java software development, you can simply understand the module as a package, and of course, the software modules and the packages in the Java syntax are not completely corresponding.)

3, the description of important classes: the required classes based on CRC card or other design methods, describing the responsibilities and helpers of each class, making the content of CRC card or using the strict class diagram in UML.

3.1 types of 1: responsibilities, collaborators, class diagrams (including the relationships with other classes)

3.2 types of 2: responsibilities, collaborators, class diagrams (including the relationships with other classes)

3.3...

4 description of important collaboration relations: refer to system function description in section second of this chapter, select important use cases (or part use cases), and use sequence diagram to describe the way of function completion.

4.1 use case 1: sequence diagram

4.2 use case 2: sequence diagram

4.3...

5 other notices: other important design decisions that need to be mentioned.

1.2 Software integration and construction

Software integration, incorporating a single software component into a whole software development activity. The Java community uses Ant, Maven (which we can do after the development). Apache Ant is an automated software building tool for

Apache Software Foundation. It is a Java class library, and also a command-line tool. It can perform some specified tasks according to a specified XML file written by a user. Ant becomes a build file through a configuration script.
Ant:

The Ant script contains the project (Projects), the target (Targets), and the task (Tasks).

Each construction file contains one project element; one project element contains multiple target elements; each target element is made up of a set of task elements.

A task completes a function, such as copying a file, compiling a project, or building a JAR file.

A goal is a set of tasks and attributes. One goal can depend on other goals.

A successful construction consists of the following three parts:

1. all source code is compiled from scratch
2. links and deployments
3. start automated test set

1.3 Test classification

Traditionally, software development can be divided into white box testing (White Box Testing, or structural testing Structural Testing) and black box testing (Black Box Testing, or function test Functional Testing).

White box testing refers to testing when the tester knows the internal data structure and algorithms of the program and can get its specific source code. The white box test checks the logic of the program, determines the test case, and covers as many code and logical combinations as possible.

Black box tests think that software loves you as a "black box," and testers do not know their internal implementation. The test engineer determines the test case according to the requirements specification, tests the function of the software, and does not need to understand the internal structure of the program.

According to the different objects implemented during the software development process, they can be divided into

1. unit test (Unit Testing)

Unit testing is usually performed by a programmer in writing code, used to test the function of a section of code, usually at the level of a function or class.

2. integration test (Integration Testing)

Integration tests are used to test the interface and the correctness of the interface and interaction between the interactive program modules during the program integration. The integration of software modules can be done in an iterative way or completed by all modules, which is generally considered to be more reasonable.

3. system test (System Testing)

System testing is concerned with the whole system's behavior and is used to test whether the integrated system is in line with its requirements and specifications. Others think that the system test is suitable for evaluating the non functional requirements of the system, such as safety, speed and reliability.

1.3.1 Unit testing

Unit testing is a test carried out by a programmer during the development process, which mainly tests the correctness of the program module.

Agile software development practice - Test driven development (TDD).

Automated unit testing.

It is important to note that test cases do not allow other test codes to be called, otherwise the absurdity of writing "tests" for the test code is created. In addition, unit testing can be done in the process of development, because we use the way of Agile development and need iterating.

1.3.2 Unit test coverage

The simplest white box test is Statement Coverage, which is to design a series of test cases, so that all the statements in the program can be executed back. But such a test does not guarantee that all the branches are tested.

The improved approach is the Branch Coverage, which designs a series of test cases to ensure that all the branches are tested.

The most complex sentence is Path Coverage, which is to design a series of test cases to ensure statements or all possible combinations in the covering program.

1.3.3 Automated unit test tool (JUnit)

JUnit (http://www.junit.org/) is an open source automation unit testing framework. Unit testing is used to verify that code behavior is an effective means to meet the expected requirements, and the JUnit framework can assist programmers in unit testing.

Without a test framework, a lot of work will be done for the writing and execution of a large number of unit tests.

2 Program format

2.1 Software integration and construction of specific cases

```
public class HelloWorld{
  public static void main(String []args){
   System.out.println("Hello World!");
  }
}
```

```
<project name="hello" default="compile">

<target name="prepare">
<mkdirdir="/tmp/classes"/>
</target>
  <target name="compile" depends="prepare">
<javacsrcdir="./src" destdir="/tmp/classes"/>
</target>

</project>
```

The above ant file can be compiled into the "/temp/classes" directory under the Java source in the current directory "SRC". The following code is the output.

```
$ ant
Buildfile: build.xml
prepare:
[mkdir] Created dir: /tmp/classes
compile:
[javac] Compiling 1 source file to /tmp/classes
BUILD SUCCESSFUL
```

2.2 Automated unit test tool (JUnit)

This example relates to the book information to add, modify and delete book information, book information of three test methods, the use of automatic testing method is the most commonly used and most convenient JUnit, this test method can

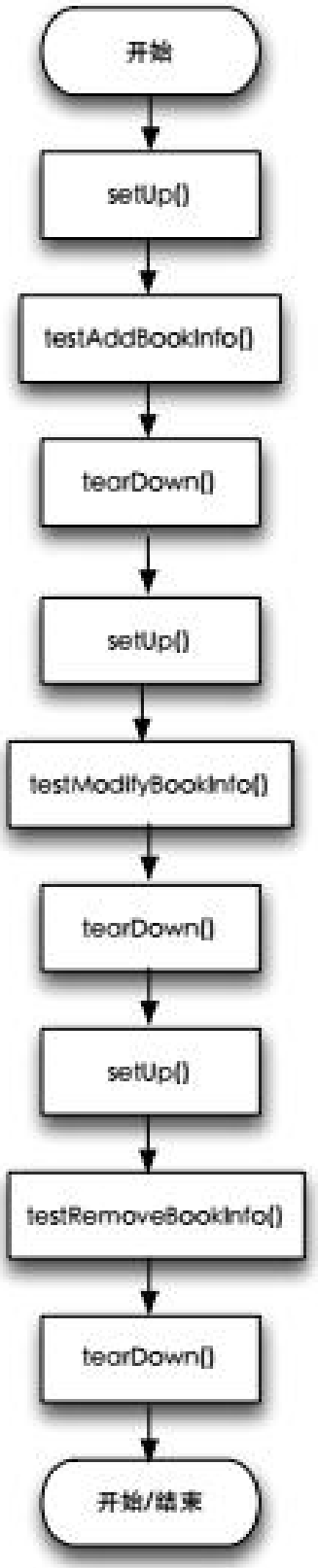
be used for reference before we PDC when using the test example. The core is to write a test method for each method. It compares the actual output value and the expected output value by assertion by giving an input value and the expected output value.

```
public class CatalogTestextends TestCase{
  privateBookInfobooka;
  privateBookInfobookb;
  privateBookInfobookc;
  private static Catalog testCatalog = new Catalog();

  publicCatalogTest(String method) {
   super(method);
}

protected void setUp(){
    //初始化
     booka = new BookInfo("123456", "Head First Java", "小明", "清华出版社", 2010, true);
     bookb = new BookInfo("234567", "Head First C++", "小力", "清华出版社", 2010, true);
     bookc = new BookInfo("345678", "Head First C#", "小白", "清华出版社", 2010, true);
}
protected void teardown(){
  //回收资源
}
//测试addBookInfo()方法
public void testAddBookInfo ()
{
   ...
}
//测试modifyBookInfo()方法
publicvoidtestModifyBookInfo(){
     ...
}
//测试removeBookInfo()方法
publicvoidtestRemoveBookInfo(){
    ...
    }
```

Specific code style



The running life cycle of JUnit