<NextDate>

测试计划

版本 <1.0>

修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| <11日/5月/2019年> | <1.1> | <NextDate程序的白盒测试计划编写> | <原帅> |
| <12日/5月/2019年> | <1.2> | <添加Date类测试用例> | <茅悦田> |
|  |  |  |  |
|  |  |  |  |

目录

1. 简介 4

1.1 目的 4

1.2 背景 4

1.3 范围 4

2. 测试需求 6

3. 测试策略 7

3.1 测试类型 7

3.1.1 单元测试 7

4. 可交付工件 11

4.1 测试模型 11

4.2 程序源代码 11

4.3 程序测试代码 11

1. 附录 12

测试计划

# 简介

## 目的

<NextDate> 的这一“测试计划”文档有助于实现以下目标：

• 确定现有项目的信息和应测试的软件构件。

• 列出推荐的测试需求（高层次）。

• 推荐可采用的测试策略，并对这些策略加以说明。

• 确定所需的资源，并对测试的工作量进行估计。

• 列出测试项目的可交付元素

## 背景

NextDate是java命令行程序，主要功能是计算给定日期的下一天的日期，由3个模块组成。

第一个模块由四个类组成，Day,Month,Year三个类继承自抽象类CalendarUnit，各自实现setCurrentPos，increment，isValid方法，能够设置自己的值，判断是否有效，和增加一个单位的值。

第二个模块由类Date组成，将Day,Month,Year三个类组合，并抽象为日期类，实现了设置日期和计算下一天日期的主要功能。

第三个模块是Main函数，实现了主要功能的调用。给定一个输入日期，通过调用Date函数的set和increment方法，得到输出结果。

## 范围

本次测试中，仅涉及到单元测试阶段，针对的是功能测试。

测试对象涉及到NextDate程序中的所有6个类的所有非抽象方法，包括：  
 **CalendarUnit**: setCurrentPos , getCurrentPos,

**Day**: Day, increment, setDay, getDay, isValid, equals

**Month**: Month, increment, getMonthSize, setMonth, getMonth, isValid, equals

**Year**: Year, setYear, getYear, increment, isLeap, isValid, equals

**Date**: Date, increment, printDate, getDay, getMonth, getYear, equals, toString

**Main**: NextDate

## 

下表列出了制定*测试计划*所用的文档，并标明了文档的可用性：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **文档 （版本/日期）** | **已创建或可用** | **已被接受或已经过复审** | **作者或来源** | **备注** |
| 需求规约 | 🞏 是 🗹 否 | 🞏 是 🞏 否 |  |  |
| 功能性规约 | 🞏 是 🗹 否 | 🞏 是 🞏 否 |  |  |
| 用例报告 | 🞏 是 🗹 否 | 🞏 是 🞏 否 |  |  |
| 原型 | 🗹 是 🞏 否 | 🞏 是 🞏 否 |  |  |

# 测试需求

下面列出了那些已被确定为测试对象的项目（用例、功能性需求和非功能性需求）。此列表说明了测试的对象。

1. 输入一个日期（年月日类型为int），输出这个日期下一天的日期，格式为month/day/year。
2. 以类的方法为粒度的单元测试。

# 测试策略

使用junit工具，在intelij idea环境下进行测试。桩的开发使用Mockito工具，利用mock函数开发外部调用的桩，利用spy函数开发类内部方法的桩。

本次测试中使用白盒测试的路径测试方法，主要考虑Miller的覆盖度量来设计测试用例。

由于NextDate中并不涉及到循环体，这里采用C1p 判断覆盖指标。

测试以类方法为粒度，主要进行路径测试。

需要注意的是，NextDate使用的是面向对象的编程方法，在Day，Month，Date类中涉及到对其他类的调用，需要开发被调用类的桩来进行测试。实际上对于类的内部方法间的调用，严格来说也需要打桩来进行测试，但是对于类的构造函数， getter， setter 方法来说，由于这三个方法互相耦合，并且常常用来作为单元测试正确性的验证方法，因此不对这三个函数打桩。

由于利用mokito工具模拟的桩函数只能模拟返回值，并不能进行实际上的操作，因此某些情况下，并不能对方法的实际功能进行测试验证，比如Date的increment方法中，调用了day，month，year的increment方法。若使用这三个类的桩来测试，并不能通过调用increment方法来实际改变day，month，year的值。因此这种情况下，测试用例只能验证函数调用路径的正确性，保证路径覆盖度。

对于main函数中的NextDate方法，是测试对象的主要功能。如果从单元测试的角度看，main函数只有对Date类set和increment方法的线性调用，不需要进行复杂的测试，对于主要功能的测试应该放在集成测试中。但这里我们认为还是应该从实际功能出发，做一个对计算nextdate功能的验证测试。方法是将main中的函数调用改为inline的方式，根据实际的调用逻辑画出路径图（见附录图一），然后设计测试用例实现条件覆盖。

## 测试类型

### 单元测试

|  |  |
| --- | --- |
| 测试目标： | 以条件覆盖的准则，测试每个类的每个函数 |
| 用例： | 1. Year类的测试  1.1 setYear  输入2000，能正常执行，通过get方法得到2000  输入0，抛出异常信息”Not a valid year”  1.2 getYear  创建year为2019的类，调用getYear得到结果2019  1.3 increment  year为2019，调用后，查看year 为2020  Year为-1，调用后查看year为1  1.4 isLeap  Year为2019，返回false  Year为2016，返回true  Yaer为1900，返回false  Year为2000，返回true  Year为-5，返回true  Year为-101，返回false  Year为-401，返回true  Year为-2，返回false  1.5 isValid  Year为2019，返回true  Year为0，返回false  1.6 equals  Year(2019)与Year(2019)比较，返回true  Year(2019)与Year(2020)比较，返回false  Year(2019)与Object类比较，返回false  2.Day类的测试  2.1 increment  对month打桩，设定  when(month.getMonthSize()).thenReturn(30).thenReturn(29);  when(month.isValid()).thenReturn(true); |
|  | 用例：  Day(1, month)，调用返回true，并且此时day为2  Day(29, month)，调用返回false  2.2 setDay  对month打桩，设定  when(month.getMonthSize()).thenReturn(30);  when(month.isValid()).thenReturn(true);  对Day的内部方法isValid打桩，设定  when(testday.isValid()).thenReturn(true).thenReturn(false)  用例：  testday.setDay(5, month)，调用getday返回5  testday.setDay(32, month)，抛出异常信息”Not a valid month”  2.3 getDay  new Day(1, month),调用后返回1  2.4 isValid  对month打桩，设定  isValid返回值依次为ture,true,true,false,true  when(month.getMonthSize()).thenReturn(30)  用例：  Day(1, month)，返回true  day.setCurrentPos(-1)后，调用返回false  day.setCurrentPos(32)后，调用返回false  month.isValid为false时，调用返回false  month为null时，调用返回false  2.5 equals  Day(1,Month(1,Year(1)))与Day(1,Month(1,Year(1)))，返回true  Day(1,Month(1,Year(1)))与Day(2,Month(1,Year(1)))，返回false  Day(1,Month(1,Year(1)))与Day(1,Month(2,Year(1)))，返回false  Day(1,Month(1,Year(1)))与Object类，返回false |
|  | 3.Date类的测试  3.1 increment  对day，month，year打桩，设定数据为2010/2/28  验证调用了day.setDay  对day，month，year打桩，设定数据为2010/12/31  验证调用了day.setDay和month.setMonth  对day，month，year打桩，设定数据为2010/12/31  验证没有调用了month.increment  3.2 printDate、getDay、getMonth、getYear  预设的日期为 2010/2/28  printDate函数返回值与字符串”2010/2/28”比较  getDay函数返回值与整数28比较  getMonth函数返回值与整数2比较  getYear函数返回值与整数2010比较  3.3 equals  创建Year、Month、Day对象  以上面三个对象为参数创建Date对象进行测试  比较2010/3/21与2010/3/21，返回true  比较2010/3/21与2010/3/22，返回false  比较2010/3/21与2010/4/21，返回false  比较2010/3/21与2011/3/21，返回false  3.4 toString  输入参数为Date对象，2010/2/28  比较返回值是否等于字符串”2010/2/28” |
|  | 4. Month类的测试  1.1 setMonth  输入5，能正常执行，通过get方法得到5  输入13，抛出异常信息”Not a valid month”  1.2 getMonth  创建month为5的类，调用getMonth得到结果5  1.3 increment  month为1，调用后，查看month 为2  month为12，调用后返回false  1.4 getMonthSize  用桩替换year  Year.leap()为false时，month=2返回28  Year.leap()为true时，month=2返回29  1.5 isValid  month为5，返回true  month为5，Year为null，返回false  month为0，返回false  month为13，返回false  month为5，year.isValid()为false时，返回false  1.6 equals  Year(5，year)与Year(5，year)比较，返回true  Year(5，year)与Year(7，year)比较，返回false  Year(5，year)与Year(5，Year(2020))比较，返回false  Year(5，year)与Object比较，返回false |
|  | 5.Main类的测试  Main类中的调用的实际路径图见附图1，测试用例满足该图的条件覆盖。  5.1 nextDate  输入1,1,0, 抛出异常信息”Not a valid year”  输入 0,1,1,. 抛出异常信息”Not a valid month”  输入 13,1,1,. 抛出异常信息”Not a valid month”  输入 1,0,1,. 抛出异常信息”Not a valid day”  输入 1,32,1,. 抛出异常信息”Not a valid day”  输入 2,28,1600,. 返回2/29/1600  输入 2,28,2100,. 返回3/1/2100  输入 2,14,114514,. 返回2/15/114514  输入 12,31,2004,. 返回1/1/2005  输入 12,31,-1,. 返回1/1/1  输入 3,31,-16401,. 返回4/1/-16401  输入 2,14,-301,. 返回2/15/-301  输入 2,28,-5,. 返回2/29/-5 |
| 完成标准： | • 所计划的测试已全部执行  • 所发现的缺陷已全部解决 |
| 需考虑的特殊事项： | 无 |

# 可交付工件

## 测试评估报告

见测试评估报告文档

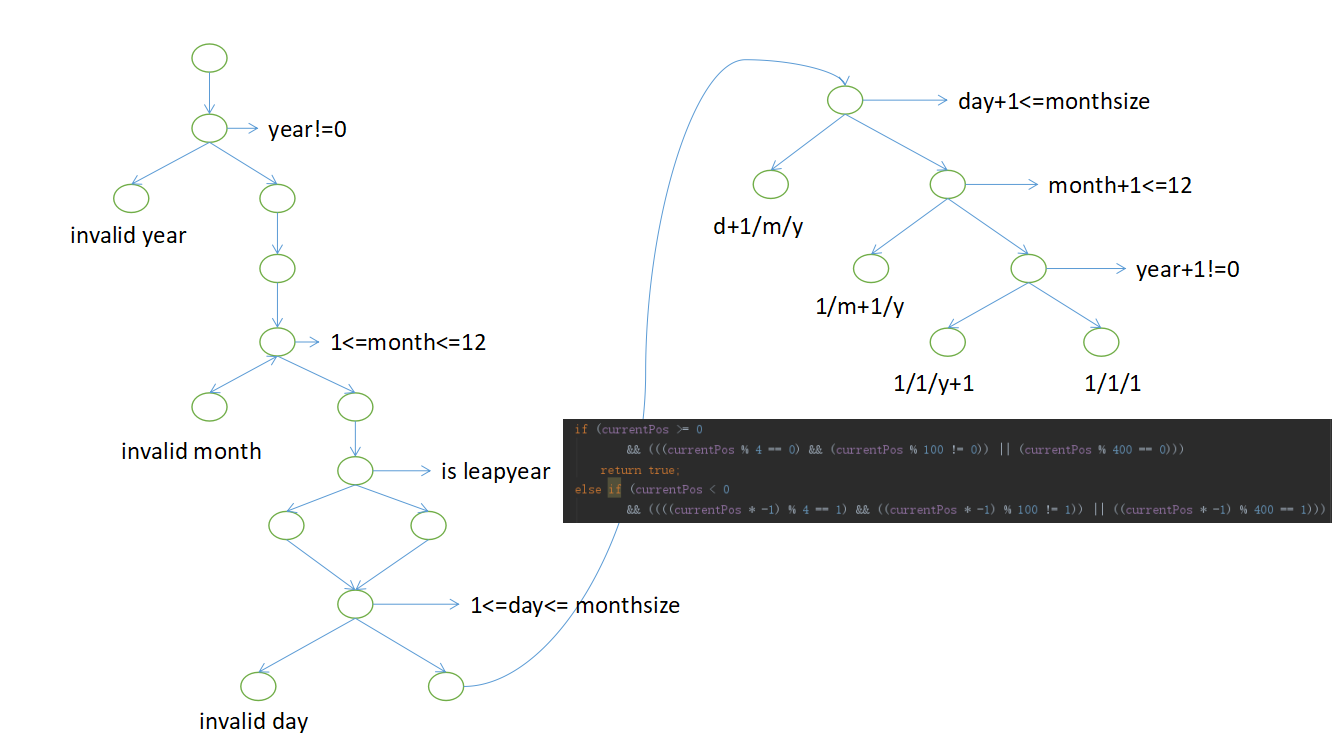
## 程序源代码

见NextDate/src/main目录

## 程序测试代码

见NextDate/src/test目录

# 附录



附图1