

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**Український державний університет науки і технологій**

Кафедра «Комп’ютерні інформаційні технології»

**Лабораторна робота №2  
з дисципліни «Якість програмного забезпечення та тестування»**

**на тему:**

«Тестування методами чорної скриньки.»

Виконав:

студент гр. ПЗ2011

Кулик С.В.

Прийняла:

Куроп’ятник О.С.

Дніпро, 2024

**Лабораторна робота №2**

**Тема:** Тестування методами чорної скриньки.

**Мета:** Отримати практичні навички тестування методами чорної скриньки.

**Завдання**

Підготувати дві чи більше функцій для тестування. Функції повинні містити не менше шести керуючих конструкцій (розгалужень і циклів). Зазначені структури мають бути вкладеними. Виконати тестування функцій методами чорної скриньки.

**Специфікація функції**

ФункціяcalculateSalary обчислює загальну заробітну плату протягом року. Якщо працівник відпрацював 10 і менше 20 днів, обчислюється зарплата за формулою zp=zpd\*day+300

Функція countWorkDays підраховує кількість робочих днів протягом місяця. Перебирає кожен день місяця і перевіряє, чи присутній на роботі ( користувач вводить: 1 – якщо був на роботі, 0 – якщо не був ).

**Класи еквівалентності**

|  |  |  |
| --- | --- | --- |
| Вхідні умови | Правильні класи еквівалентності | Неправильні класи еквівалентності |
| Масив 1 | Цілі числа(1) | Дробові числа |
| Числа більше 0 (2) |
| Масив 0 | Вхідні дані є 0 (4) | Не цілі числа |

**Тести за методом еквівалентних розбиттів**

|  |  |  |  |
| --- | --- | --- | --- |
| ***Класи що покриваються*** | ***Тест*** | ***Вхід*** | ***Вихід*** |
| 3, 6, 7 | 1 | daysWorked : 15,18,16,18,17,19,14,16,18,18,15 ,19 dailyRate: 450 | calculateSalary: 94 950 |
| 3,6,8 | 2 | visitWork : 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 | 15 |
| 1, 5, 9 | 3 | daysWorked : 22,23,15,21,7,17,14,19,13,9,10,19 dailyRate: 500 | calculateSalary: 96 600 |

**Тести за методом припущення про помилку**Виконаємо тестування методом припущення про помилку. Висунемо ряд припущень:

1. Якщо для двох параметрів ввести всюди 0, чи відбудеться помилка?
2. Якщо в нас один з параметрів є 0, а другий заповнений, яке буде виведення першого параметру?
3. Якщо ввести для масиву який зберігає данні про дні значення більше 1, чи відбудеться зарахування на виході?
4. Якщо ввести текст замість значень, чи приймиться як за число?

|  |  |  |  |
| --- | --- | --- | --- |
| ***Припущення*** | ***Тест*** | ***Вхід*** | ***Вихід*** |
| 1 | 1 | visitWork : 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 daysWorked : 0,0,0,0,0,0,0,0,0,0,0 ,0 | 0 |
| 2 | 2 | visitWork : 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 |
| 3 | 3 | visitWork : 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 | 30 |
| 4 | 4 | visitWork : 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 гав | 30 |

**Тести за методом причин та наслідків**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Причини та наслідки | Умовне позначення | Значення | Помітки про присутність | | |
| 1 | 2 | 3 |
| Причини | П1 | Відпрацьовано 10 і менше 20 днів | 1 |  |  |
| П2 | Відпрацьовано менше 10 днів |  | 1 |  |
| П3 | 0 днів |  |  | 1 |
| Наслідки | Н1 | Обчислюється зарплата за формулою zp=zpd\*day+300 | 1 |  |  |
| Н2 | Обчислюється зарплата за формулою zp=zpd\*day |  | 1 |  |
| Н3 | Зарплата 0 |  |  | 1 |

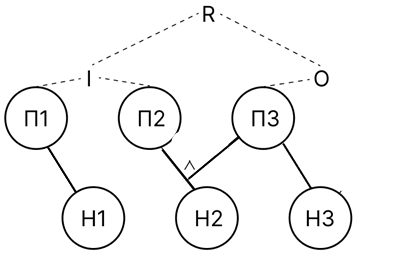


рис. 1, функціональна діаграма

**Тестування методом граничних умов**

|  |  |  |  |
| --- | --- | --- | --- |
| ***Помітка*** | ***Тест*** | ***Вхід*** | ***Вихід*** |
| 1 | 1 | daysWorked : 15,18,16,18,17,19,14,16,18,18,15 ,19 dailyRate: 450 | calculateSalary: 94 950 |
| 2 | 2 | daysWorked : 5,8,6,8,7,9,4,6,8,8,5,9 dailyRate: 550 | calculateSalary: 423550 |

**Специфікація функції №2**

ФункціяcalculateSalary обчислює загальну заробітну плату протягом року. Якщо працівник відпрацював 10 і менше 20 днів, обчислюється зарплата за формулою zp=zpd\*day+300

Функція countWorkDays підраховує кількість робочих днів протягом місяця. Перебирає кожен день місяця і перевіряє, чи присутній на роботі ( користувач вводить: 1 – якщо був на роботі, 0 – якщо не був ).

**Класи еквівалентності**

|  |  |  |
| --- | --- | --- |
| Вхідні умови | Правильні класи еквівалентності | Неправильні класи еквівалентності |
| Масив 1 | Цілі числа(1) | Дробові числа |
| Числа більше 0 (2) |
| Масив 0 | Вхідні дані є 0 (3) | Не цілі числа |

**Тести за методом еквівалентних розбиттів**

|  |  |  |  |
| --- | --- | --- | --- |
| ***Класи що покриваються*** | ***Тест*** | ***Вхід*** | ***Вихід*** |
| 3, 6, 7 | 1 | daysWorked : 15,18,16,18,17,19,14,16,18,18,15 ,19 dailyRate: 450 | calculateSalary: 94 950 |
| 3,6,8 | 2 | visitWork : 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 | 15 |
| 1, 5, 9 | 3 | daysWorked : 22,23,15,21,7,17,14,19,13,9,10,19 dailyRate: 500 | calculateSalary: 96 600 |

**Тести за методом припущення про помилку** Виконаємо тестування методом припущення про помилку. Висунемо ряд припущень:

1. Якщо для двох параметрів ввести всюди 0, чи відбудеться помилка?
2. Якщо в нас один з параметрів є 0, а другий заповнений, яке буде виведення першого параметру?
3. Якщо ввести для масиву який зберігає данні про дні значення більше 1, чи відбудеться зарахування на виході?
4. Якщо ввести від’ємне значення, як воно зарахується??

|  |  |  |  |
| --- | --- | --- | --- |
| ***Припущення*** | ***Тест*** | ***Вхід*** | ***Вихід*** |
| 1 | 1 | daysWorked : 0,0,0,0,0,0,0,0,0,0,0,0,0 dailyRate: 0 | 0 |
| 2 | 2 | visitWork : 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 |
| 3 | 3 | visitWork : 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 | 30 |
| 4 | 4 | visitWork : 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 -1 | 29 |

**Тести за методом причин та наслідків**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Причини та наслідки | Умовне позначення | Значення | Помітки про присутність | | |
| 1 | 2 | 3 |
| Причини | П1 | Відпрацьовано 0 днів в місяць | 1 |  |  |
| П2 | Присутні 1 та 0 |  | 1 |  |
| П3 | Заповнено все 1 |  |  | 1 |
| Наслідки | Н1 | Відсутні дні роботи | 1 |  |  |
| Н2 | Кількість заповнених 1 = кількості робочих днів за місяць |  | 1 |  |
| Н3 | Працівник був без вихідних |  |  | 1 |

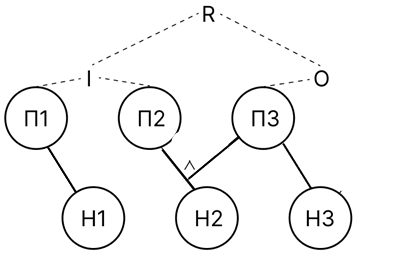


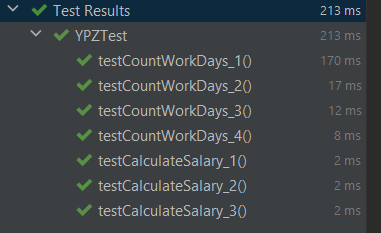
рис. 2, функціональна діаграма

**Тестування методом граничних умов**

|  |  |  |  |
| --- | --- | --- | --- |
| ***Помітка*** | ***Тест*** | ***Вхід*** | ***Вихід*** |
| 1 | 1 | visitWork : 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 | 15 |
| 2 | 2 | visitWork : 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 | 30 |

**Unit-тестування :**

import org.junit.jupiter.api.Assertions;  
import org.junit.jupiter.api.Test;  
  
public class YPZTest {  
 // Тест для обчислення зарплати з різною кількістю робочих днів  
 @Test  
 public void testCalculateSalary\_1() {  
 int[] daysWorked = {10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21};  
 int dailyRate = 100;  
 int expectedSalary = (dailyRate \* 10 + 300) + (dailyRate \* 11 + 300) + (dailyRate \* 12 + 300)  
 + (dailyRate \* 13 + 300) + (dailyRate \* 14 + 300) + (dailyRate \* 15 + 300)  
 + (dailyRate \* 16 + 300) + (dailyRate \* 17 + 300) + (dailyRate \* 18 + 300)  
 + (dailyRate \* 19 + 300) + (dailyRate \* 20 + 300) + (dailyRate \* 21);  
 assertEquals(expectedSalary, YPZ.calculateSalary(daysWorked, dailyRate));  
 }  
  
 private void assertEquals(int expectedSalary, int calculateSalary) {  
 }  
 // Тест для обчислення зарплати з відсутністю робочих днів  
 @Test  
 public void testCalculateSalary\_2() {  
 // Тест для мінімального вводу  
 int[] daysWorkedMin = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};  
 int dailyRateMin = 100;  
 int expectedSalaryMin = 0;  
 assertEquals(expectedSalaryMin, YPZ.calculateSalary(daysWorkedMin, dailyRateMin));  
 }  
 // Тест для обчислення зарплати з максимальною кількістю робочих днів в місяці  
 @Test  
 public void testCalculateSalary\_3() {  
 // Тест для максимального вводу  
 int[] daysWorkedMax = {31, 31, 31, 31, 31, 31, 31, 31, 31, 31, 31, 31};  
 int dailyRateMax = 100;  
 int expectedSalaryMax = dailyRateMax \* 31 \* 12;  
 Assertions.assertEquals(expectedSalaryMax, YPZ.calculateSalary(daysWorkedMax, dailyRateMax));  
 }  
 // Тест для підрахунку кількості робочих днів в місяці  
 @Test  
 public void testCountWorkDays\_1() {  
 int[] visitWork = {1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0};  
 int expectedWorkDays = 16;  
 assertEquals(expectedWorkDays, YPZ.countWorkDays(visitWork));  
 }  
 // Тест для підрахунку кількості робочих днів при відсутності робочих днів  
 @Test  
 public void testCountWorkDays\_2() {  
 // Тест для мінімального вводу  
 int[] visitWorkMin = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};  
 int expectedWorkDaysMin = 0;  
 Assertions.assertEquals(expectedWorkDaysMin, YPZ.countWorkDays(visitWorkMin));  
 }  
 // Тест для підрахунку кількості робочих днів при всіх робочих днях в місяці  
 @Test  
 public void testCountWorkDays\_3() {  
 // Тест для максимального вводу  
 int[] visitWorkMax = {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1};  
 int expectedWorkDaysMax = 30;  
 Assertions.assertEquals(expectedWorkDaysMax, YPZ.countWorkDays(visitWorkMax));  
 }  
 @Test  
 public void testCountWorkDays\_4() {  
 // Тест для випадку з випадковим розподілом 1 та 0  
 int[] visitWorkRandom = {1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0};  
 int expectedWorkDaysRandom = 15;  
 Assertions.assertEquals(expectedWorkDaysRandom, YPZ.countWorkDays(visitWorkRandom));  
 }  
 }

****

**Висновок:**

Тестування методами чорної скриньки - це ключовий етап в розробці програмного забезпечення. Воно спрямоване на перевірку функціональності програми без прямого доступу до внутрішньої структури коду. Головною метою цього виду тестування є переконання, що програма працює вірно відповідно до специфікацій та очікувань користувачів.

Тестування чорної скриньки виконується з точки зору зовнішнього користувача, і тестувальники не мають доступу до внутрішньої реалізації програми. Вони перевіряють, чи відповідає програма вхідним даним та генерує правильні вихідні результати. Це включає тестування різних сценаріїв використання та перевірку відповідності програми функціональним вимогам.

Під час тестування чорної скриньки можуть бути виявлені різноманітні проблеми, такі як інтерфейсні недоліки, неправильна реакція на вхідні дані або невідповідність функціональним вимогам. Результати тестів допомагають розробникам виправляти виявлені проблеми та забезпечувати високу якість та надійність програмного продукту.