Содержание

[Цель работы: 2](#_Toc5205183)

[Спецификация ПО: 2](#_Toc5205184)

[Исходных набор тестов 2](#_Toc5205185)

[Ход выполнения регрессионного тестирования 3](#_Toc5205186)

[Приложение: 5](#_Toc5205187)

Цель работы:

Научиться производить регрессионное тестирование измененного программного обеспечения с использованием различных методов отбора тестов.

Спецификация ПО:

Вариант № 1.

Необходимо написать программу для выполнения расчета суммы получаемой студентом стипендии по результатам сдачи сессии. При сдаче сессии хотя бы с одной оценкой «удовлетворительно» либо сдаче сессии после установленного срока, студент стипендии не получает. При сдаче сессии вовремя и без оценок «удовлетворительно», студент получает стипендию, причем она рассчитывается индивидуально следующим образом:

1) при сдаче сессии только на оценки «хорошо», стипендия равна А рублей;

2) при сдаче сессии на оценки «хорошо» и «отлично», к сумме А рублей начисляется надбавка 25 %;

3) при сдаче сессии только на оценки «отлично», к сумме А рублей начисляется надбавка 50 %.

Исходные данные, вводимые пользователем:

1) оценка по каждой дисциплине из списка возможных дисциплин, а также указание того, вовремя или не вовремя сдана дисциплина;

2) значение А.

Исходных набор тестов

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Номера столбцов | | 1 | 2 | 3 | 4 | 5 |
| Причины | 1 | 0 | 1 | 1 | 1 | 1 |
| 2 | 1 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 1 |
| 4 | 1 | 1 | 0 | 0 | 0 |
| 5 | 0 | 0 | 1 | 0 | 0 |
| 6 | 0 | 0 | 0 | 1 | 0 |
| Вторичные причины | 11 | 1 | 0 | 0 | 0 | 0 |
| 12 | 1 | 0 | 0 | 0 | 0 |
| Следствия | 101 | 1 | 0 | 0 | 0 | 1 |
| 102 | 0 | 1 | 0 | 0 | 0 |
| 103 | 0 | 0 | 1 | 0 | 0 |
| 104 | 0 | 0 | 0 | 1 | 0 |

Ход выполнения регрессионного тестирования

Метод выборочного регрессионного тестирования называется **безопасным**, если при некоторых четко определенных условиях он не исключает тестов (из доступного набора тестов), которые обнаружили бы ошибки в измененной программе, то есть обеспечивает выбор всех тестов, обнаруживающих изменения. В соответствии с требованиями безопасного метода для повторного выполнения должны быть отобраны тесты все тесты, так как они участвуют в измененной части кода.

Процедура минимизации набора тестов ставит целью отбор минимального (в терминах количества тестов) подмножества, необходимого для покрытия каждого элемента программы, зависящего от изменений. Для проверки корректности программы используются только тесты из минимального подмножества. Для случая регрессионного тестирования функции из лабораторной работы данный метод ограничится *отбором* теста 1, так как он полностью покрывает измененные строки согласно таблице 1.

Значение методов, основанных на покрытии кода, состоит в том, что они гарантируют сохранение выбранным набором тестов требуемой степени покрытия элементов программы относительно некоторого критерия структурного покрытия, использовавшегося при создании первоначального набора тестов. Так, согласно методу стопроцентного покрытия измененного кода, для приведенного примера таблицы существует 1 способ отобрать 3 теста в соответствии с этим критерием. Это тесты 3, 4 и 5. 1 и 2 теста недостаточно.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| public static double scholarship(int[] score, boolean[] time, double A) { | 1 | 2 | 3 | 4 | 5 |
| for (int i = 0; i < time.length; i++) { | \* | \* | \* | \* | \* |
| if (!time[i]) return 0;} | \* | \* | \* | \* | \* |
| int four = 0, five = 0; | \* | \* | \* | \* | \* |
| for (int i = 0; i < score.length; i++) { | \* | \* | \* | \* | \* |
| if (score[i] < 4){ |  |  |  |  | \* |
| return 0; |  |  |  |  | \* |
| }else if (score[i] < 5){ |  |  | \* |  |  |
| four += 1; |  |  | \* |  |  |
| } else { |  |  | \* | \* |  |
| five += 1;}} |  |  | \* | \* |  |
| if (five != 0) return A; |  |  |  | \* |  |
| if(four != 0) return A \* 1.25; |  |  | \* | \* |  |
| return (double) A \* 1.5;}} |  |  |  | \* |  |

Таблица 1

Приложение:

Исходный код программы:

import java.util.Scanner;  
  
public class Main {  
 public static double *wage*;  
 public static String[] *subject* = new String[5];  
 public static int[] *score* = new int[5];  
 public static boolean[] *time* = new boolean[5];  
  
 public static void main(String[] args) {  
 System.*out*.println("Введите размер А:");  
 Scanner in = new Scanner(System.*in*);  
 *wage* = in.nextInt();  
 *subject*[0] = "IT";  
 *subject*[1] = "Math";  
 *subject*[2] = "Economy";  
 *subject*[3] = "Physics";  
 *subject*[4] = "Sport";  
 for (int i = 0; i < *subject*.length; i++) {  
 System.*out*.println(*subject*[i]);  
 *score*[i] = in.nextInt();  
 *time*[i] = in.nextInt() == 1;  
 }  
  
 double b = *scholarship*(*score*, *time*, *wage*);  
  
 for (int i = 0; i < *subject*.length; i++) {  
 System.*out*.print(*subject*[i] + " ");  
 System.*out*.println(*score*[i]);  
 }  
  
 }  
  
 public static double scholarship(int[] score, boolean[] time, double A) {  
 int four = 0, five = 0;  
 for (int i = 0; i < score.length; i++) {  
 if (score[i] < 4){  
 return 0;  
 }else if (score[i] < 5){  
 four += 1;  
 } else {  
 five += 1;  
 }  
 }  
 if (five == 0) return A;  
 if (four != 0) return A \* 1.25;  
 return (double) A \* 1.5;  
 }  
}

Исправленный код программы:

import java.util.Scanner;  
  
public class Main {  
 public static double *wage*;  
 public static String[] *subject* = new String[5];  
 public static int[] *score* = new int[5];  
 public static boolean[] *time* = new boolean[5];  
  
 public static void main(String[] args) {  
 System.*out*.println("Введите размер А:");  
 Scanner in = new Scanner(System.*in*);  
 *wage* = in.nextInt();  
 *subject*[0] = "IT";  
 *subject*[1] = "Math";  
 *subject*[2] = "Economy";  
 *subject*[3] = "Physics";  
 *subject*[4] = "Sport";  
 for (int i = 0; i < *subject*.length; i++) {  
 System.*out*.println(*subject*[i]);  
 *score*[i] = in.nextInt();  
 *time*[i] = in.nextInt() == 1;  
 }  
  
 double b = *scholarship*(*score*, *time*, *wage*);  
  
 for (int i = 0; i < *subject*.length; i++) {  
 System.*out*.print(*subject*[i] + " ");  
 System.*out*.println(*score*[i]);  
 }  
  
 }  
  
 public static double scholarship(int[] score, boolean[] time, double A) {  
 for (int i = 0; i < time.length; i++) {  
 if (!time[i]) return 0;  
 }  
 int four = 0, five = 0;  
 for (int i = 0; i < score.length; i++) {  
 if (score[i] < 4){  
 return 0;  
 }else if (score[i] < 5){  
 four += 1;  
 } else {  
 five += 1;  
 }  
 }  
 if (five != 0) return A;  
 if (four != 0) return A \* 1.25;  
 return (double) A \* 1.5;  
 }  
}