Тестування програми – це процес виконання програми з метою виявлення помилок. Тестом називається набор вхідних і вихідних даних, які відповідають специфікації.

До найбільш ефективних методів тестування можна віднести комбінацію стратегій чорного ящика і білого ящика. Стратегія «чорного ящика» передбачає вивчення і тестування зовнішньої поведінки функціонуючої програми; стратегія «білого ящика» передбачає вивчення і тестування внутрішньої архітектури і логіки програми на основі її вихідних кодів.

Тестування методом «білого ящика»

«Білий ящик» ­– тестування коду на предмет логіки роботи програми та коректності її роботи з точки зору компілятора тієї мови, на якій її написано.

Техніка тестування за принципом «білого ящика» (техніка, якою керує логіка програми) дозволяє перевірити внутрішню структуру програми. Виходячи з цієї стратегії тестувальник отримує тестові дані шляхом аналізу логіки роботи програми.

Найбільш поширеними методами тестування за технікою «білого ящика» є наступні:

покриття операторів;

покриття рішень;

покриття умов;

покриття рішень та умов;

комбінаторне покриття умов.

Було вирішено, що найбільш доцільним буде проведення тестування розроблених методів за допомогою покриття умов. Метод покриття умов полягає в такому підборі тестів, коли кожна умова (елементарне судження в умовних операторах) приймає як істинне так і хибне значення.

Вхідними даними для тестування програми методами білого ящика є специфікації функції та текст програми. Таким чином далі буде надаватися:

текст методу, який тестуємо;

специфікація;

тестування методом покриття умов.

Метод checkTime()

Специфікація: метод перевіряє коректність уведеної дати відправлення (якщо параметр time містить місяць і число, повертає true і змінює поточний стан параметра date, інакше повертає false);

Вхідні дані:

time – дата для перевірки (строкове значення);

date – об’єкт класу Date, до якого буде занесено розпізнану дату.

Вихідні дані:

ознака коректності отриманої дати (логічне значення).

Текст методу:

public static boolean checkTime(String time, Date date) {

boolean res = false;

int number = -1;

int month = -1;

int maxDays = -1;

String[] words = time.split(" ");

for (String i : words) {

for (Month m : months) {

if (m.isSuitable(i)) {

month = m.getID();

maxDays = m.getMaxDays();

}

}

}

for (String i : words) {

try {

number = Integer.parseInt(i);

} catch (NumberFormatException e) {}

}

if ((month != -1) && (number > 0) && (number < maxDays)) {

res = true;

Calendar calendar = Calendar.getInstance();

calendar.set(calendar.get(Calendar.YEAR), month, number);

date.setTime((calendar.getTime()).getTime());

}

return res;

}

Розроблені тести

Тест 1

Вхід: time = “26 июня”, date = new Date()

Тест 2

Вхід: time = “32 чактобря”, date = new Date()

Тест 3

Вхід: time = “май -1”, date = new Date()

Результати тестування за методом покриття умов наведено у таблиці 4.1.

Таблиця 4.1 – Результати тестування методу checkTime()

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Тест № | month != -1 | | number > 0 | | number < maxDays | |
| + | - | + | - | + | - |
| 1 | + |  | + |  | + |  |
| 2 |  | + | + |  |  | + |
| 3 | + |  |  | + | + |  |

Метод handleMessage

Специфікація: обробляє повідомлення, що надходять від клієнта (містяться у параметрі message) та повертає власні повідомлення.

Вхідні дані:

message – прийняте повідомлення у форматі JSON (строкове значення).

Вихідні дані:

власне повідомлення у форматі JSON (строкове значення).

Текст методу:

…

Розроблені тести

Тест 1

Вхід: message = “{‘type’: ‘check\_city’, ‘text’: ‘Павлоград’}”

Тест 2

Вхід: message = “{‘type’: ‘check\_time’, ‘text’: ‘’}”

Тест 3

Вхід: message = “{‘type’: ‘find\_trains’, ‘text’: ‘’}”

Результати тестування за методом покриття умов наведено у таблиці 4.2.

Таблиця 4.2 – Результати тестування методу handleMessage()

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Тест № | type.equals  (“check\_city”) | | type.equals  (“check\_time”) | | type.equals  (“find\_trains”) | | rs.next() | |
| + | - | + | - | + | - | + | - |
| 1 | + |  |  | + |  | + | + |  |
| 2 |  | + | + |  |  | + |  | + |
| 3 |  | + |  | + | + |  |  | + |

Метод processMessage()

Специфікація:

Вхідні дані:

message – прийняте повідомлення у форматі JSON (строкове значення).

Метод не повертає значень.

Текст методу:

…

Розроблені тести

Тест 1

Вхід: message = “{‘type’: ‘check\_city\_response’, ‘text’: ‘yes’}”, currentState = “city\_confirm”

Тест 2

Вхід: message = “{‘type’: ‘check\_city\_response’, ‘text’: ‘no’}”, currentState = “city\_input”

Тест 3

Вхід: message = “{‘type’: ‘check\_city\_response’, ‘text’: ‘yes’}”, currentState = “arrival\_input”

Тест 4

Вхід: message = “{‘type’: ‘check\_time\_response’, ‘text’: ‘yes’}”

Тест 5

Вхід: message = “{‘type’: ‘find\_trains\_response’, ‘text’: ‘{}’}”

Тест 6

Вхід: message = “{‘type’: ‘find\_trains\_response’, ‘text’: ‘{‘departure’: ‘’, ‘arrival’: ‘’, ‘time’: ‘’, ‘cost’: ‘’}’}”

Результати тестування за методом покриття умов наведено у таблиці 4.3.

Таблиця 4.3 – Результати тестування методу handleMessage()

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Тест № | answer.type== “check\_city\_response” | | answer.type== “check\_time\_response” | | answer.type== “find\_trains\_response” | | currentState== “city\_confirm” | | currentState== “city\_input” | | currentState== “arrival\_input” | | str == “” | | answer.text == “yes” | |
| + | - | + | - | + | - | + | - | + | - | + | - | + | - | + | - |
| 1 | + |  |  | + |  | + | + |  |  | + |  | + |  |  | + |  |
| 2 | + |  |  | + |  | + |  | + | + |  |  | + |  |  |  | + |
| 3 | + |  |  | + |  | + |  | + |  | + | + |  |  |  |  | + |
| 4 |  | + | + |  |  | + |  |  |  |  |  |  |  |  | + |  |
| 5 |  | + |  | + | + |  |  |  |  |  |  |  | + |  |  |  |
| 6 |  | + |  | + | + |  |  |  |  |  |  |  |  | + |  |  |

Тестування методом «чорного ящика»

У цьому методі програма розглядається як «чорний ящик», тобто її алгоритм невідомий. Мета тестування – визначення обставин, за яких поведінка програми не відповідає специфікації. Для знаходження усіх помилок у програмі необхідно виконати вичерпне тестування, тобто тестування на всіх можливих наборах даних. Для більшості програм це неможливо, тому застосовують «розумне» тестування, обмежуючись невеликою підмножиною з усіх можливих наборів даних. При цьому необхідно вибирати найбільш підходящі підмножини – ті, в яких імовірність знаходження помилок найбільша.

Найбільш поширеними методами тестування за технікою «чорного ящика» є наступні:

еквівалентне розбиття;

аналіз граничних значень;

аналіз причинно-наслідкових зв’язків;

припущення про помилку.

Було вирішено, що найбільш доцільним буде проведення тестування розроблених методів методом еквівалентного розбиття. Метод еквівалентного розбиття полягає у розбитті всіх тестів на класи еквівалентності. Якщо один тест виявляє помилку, то і інший виявить, та навпаки.

Метод checkTime()

Специфікація: метод перевіряє коректність уведеної дати відправлення (якщо параметр time містить місяць і число, повертає true і змінює поточний стан параметра date, інакше повертає false);

Вхідні дані:

time – дата для перевірки (строкове значення);

date – об’єкт класу Date, до якого буде занесено розпізнану дату.

Вихідні дані:

ознака коректності отриманої дати (логічне значення).

Класи еквівалентності

|  |  |  |
| --- | --- | --- |
| Вхідні умови | Правильні класи | Неправильні класи |
| Місяць | Назви місяців російською мовою у називному або родовому відмінках | Будь-які інші слова |
| Число | Від 1 до максимальної к-сті днів у визначеному місяці | Менше 1 або більше до максимальної к-сті днів у визначеному місяці |

Розроблені тести

Тест 1

Вхід: time = “26 июня”, date = new Date()

Вихід: true

Тест 2

Вхід: time = “май 31”, date = new Date()

Вихід: true

Тест 3

Вхід: time = “выходной”, date = new Date()

Вихід: false

Тест 4

Вхід: time = “-1 июня”, date = new Date()

Вихід: false

Тест 5

Вхід: time = “май 32”, date = new Date()

Вихід: false