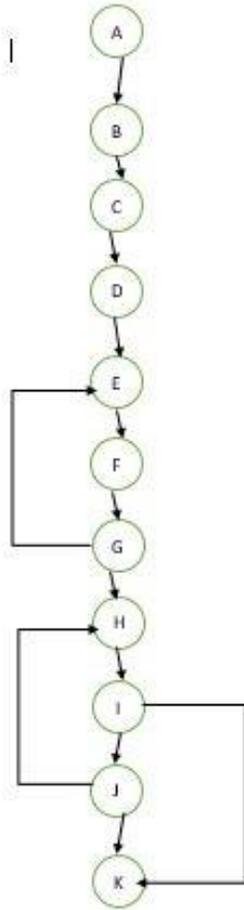


Muhis511, Muhammad Ismail, Lab: 6: Testing Planning

Part 1: White Box Testing

Jump search Control flow graph:



```
@Override
public <T extends Comparable<T>> int find(T[] array, T key) {
    int length = array.length; /* length of array */
    int blockSize = (int) Math.sqrt(length); /* block size to be jumped */

    int limit = blockSize;
    while (key.compareTo(array[limit]) > 0 && limit < array.length - 1) {
        limit = Math.min(limit + blockSize, array.length - 1);
    }

    for (int i = limit - blockSize; i <= limit; i++) {
        if (array[i] == key) {
            /* execute linear search */
            return i;
        }
    }
    return -1; /* not found */
}
```

Cyclomatic complexity:

$$V(G) = E - N + 2P$$

$$V(G) = 13 - 11 + 2*1$$

$$V(G) = 4$$

Paths:

1	ABCDEFGHIK
2	ABCDEFGHIJK
3	ABCDEFGEFGFEFGHIJK

Description:

Path 1, going through the path that found the required element.

Path 2, going through the path that don't find the required element.

Part 2: Black Box Testing & Planning

Identify Feature Sets (TD1):

In the insurance application basically it is the app that use to calculate insurance cost and deductible for client. It vary from client to client depending on how many cars they register for insurance also if they have red car that gone be change more. In an accident client get deductible calculated according to age and driving experience and no of accidents in a year.

Features:

- Calculate insurance cost for clients depending on their membership, no of accidents, driving license years, and age and car color.
- Update increase cost with change in no of accidents.
- Register cars, members, accidents in the system.

Assumptions:

Assume Db is created and working perfectly with the application.

Derive Test Conditions (TD2):

Input values are:

YearofBirth*	Integer	1990, 1986, 2020
YearofLicence**	Integer	2015, 2010, 1990
CarColor	String	Red, blue, white
IsGoldMember	bool	True, False
NoofAccidents	Integer	0, 3, 5

* YearofBirth convert to Age

**YearofLicence convert to Year of Licence

These are the inputs that effect the result of insurance application. I will create test cases by variation in above values.

Testing Conditions:

1. Age < 30
2. Age > 30
3. License > 5
4. License < 5
5. CarColor = Red
6. CarColor = Other
7. IsGoldMember = Yes
8. IsGoldMember = No
9. NoofAccidents = 0
10. NoofAccidents = 1
11. NoofAccidents = 2
12. NoofAccidents = 3
13. NoofAccidents = 4

Derive Test Coverage Items (TD3):

Test cases:

Test ID	Age	Licence	CarColor	IsGoldMember	NoOfAccidents	Coverage
1	35	4	Black	yes	0	2, 4, 6, 7, 9
2	25	3	Red	yes	1	1, 4, 5, 7, 10
3	37	6	Black & Red	No	2	2, 4, 5, 6, 8, 11
4	37	6	Black	Yes	4	2, 4, 6, 7, 13
5	26	7	Red	No	0	1, 3, 5, 8, 9
6	22	0	White	Yes	3	1, 4, 6, 7, 12
7	40	8	Blue,White,Red	No	0	2, 3, 5, 6, 8, 9

Derive Test Cases (TD4):

Test ID	Age	Licence	CarColor	IsGold Member	NoOf Accidents	Coverage	Insurance Cost	Deductible to client
1	35	4	Black	yes	0	2, 4, 6, 7, 9	500	5000
2	25	3	Red	yes	1	1, 4, 5, 7, 10	700	8000
3	37	6	Black & Red	No	2	2, 4, 5, 6, 8, 11	800	7500
4	37	6	Black	Yes	4	2, 4, 6, 7, 13	500	15000
5	26	7	Red	No	0	1, 3, 5, 8, 9	600	5000

6	22	0	White	Yes	3	1, 4, 6, 7, 12	600	12000
7	40	8	Blue, White, Red	No	0	2, 3, 5, 6, 8, 9	1000	5000

Assemble Test Sets (TD5):

Test Set (1-7)

Derive Test Procedures (TD6):

We can run above test cases on working database. And then we can add some data of clients, cars, membership etc. so we can run these tests to verify features of the system. We can use Junit to test all the above cases by implementing them in Junit tests.

Part 3: Integration testing

```
@Override
public int registerNewAccident(int clientId) {
    ClientProfile cl = clientDB.findById(clientId);
    int accidents = cl.getNumberOfAccidentsThisYear() + 1;
    cl.setNumberOfAccidentsThisYear(accidents);
    clientDB.updateClientProfile(clientId, cl);
    return accidents;
}
```

