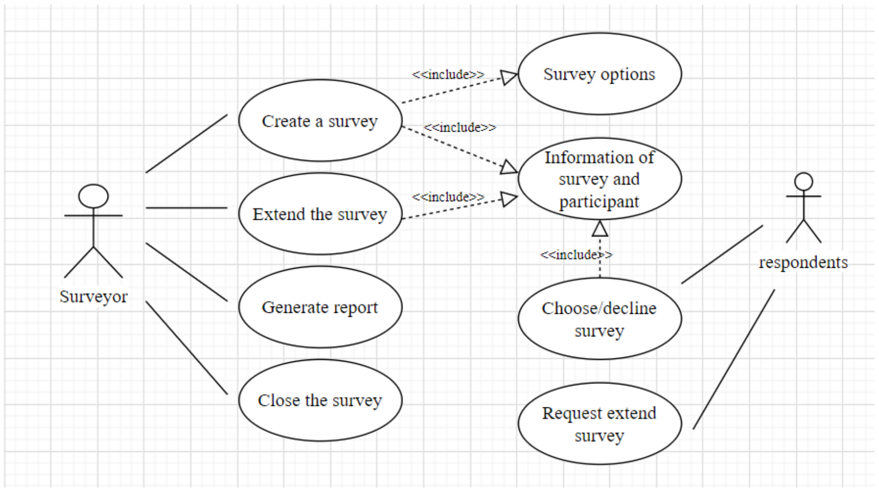
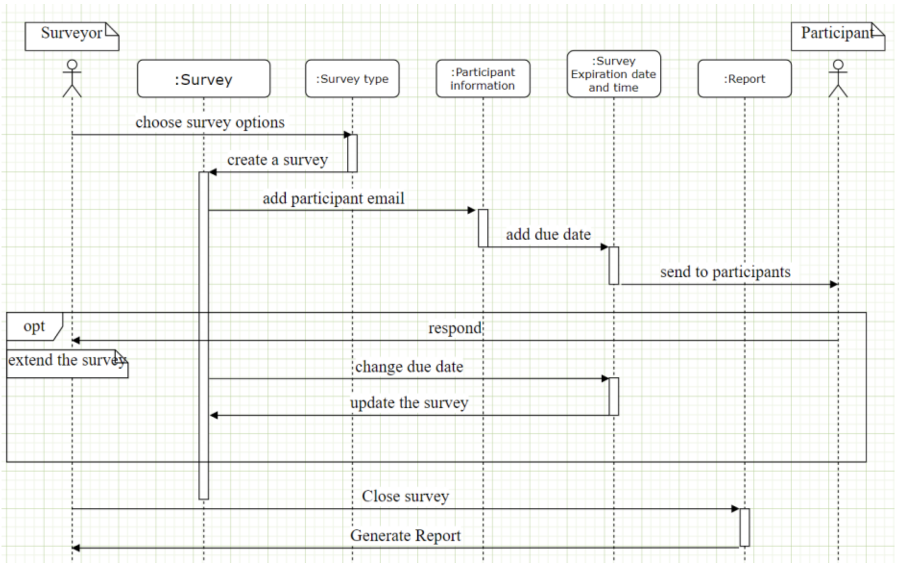
Task 1

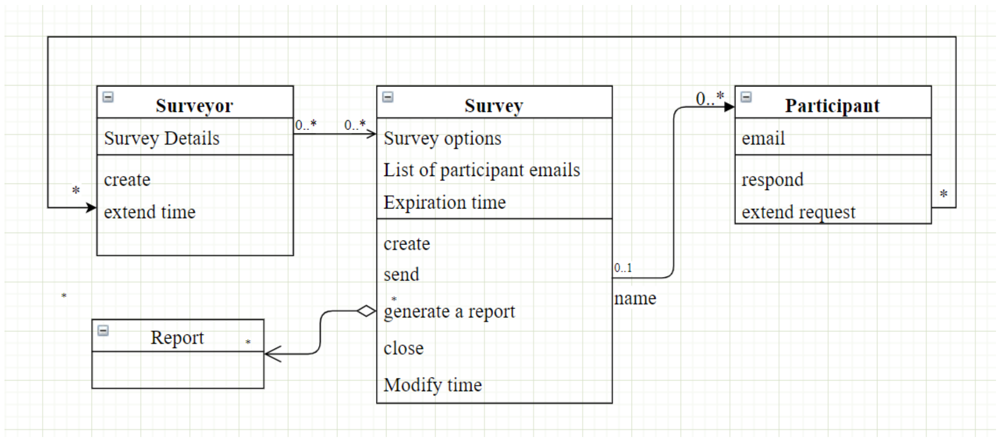
1.


Surveyor
Create a survey
Extend the survey
Generate report
Close the survey
<<include>>
<<include>>
<<include>>
<<include>>
Survey options
Information of survey and participant
Choose/decline survey
Request extend survey
respondents

2.


Surveyor
Participant
:Survey
:Survey type
:Participant information
:Survey Expiration date and time
:Report
choose survey options
create a survey
add participant email
add due date
send to participants
opt
respond
extend the survey
change due date
update the survey
Close survey
Generate Report

3.



Task 2

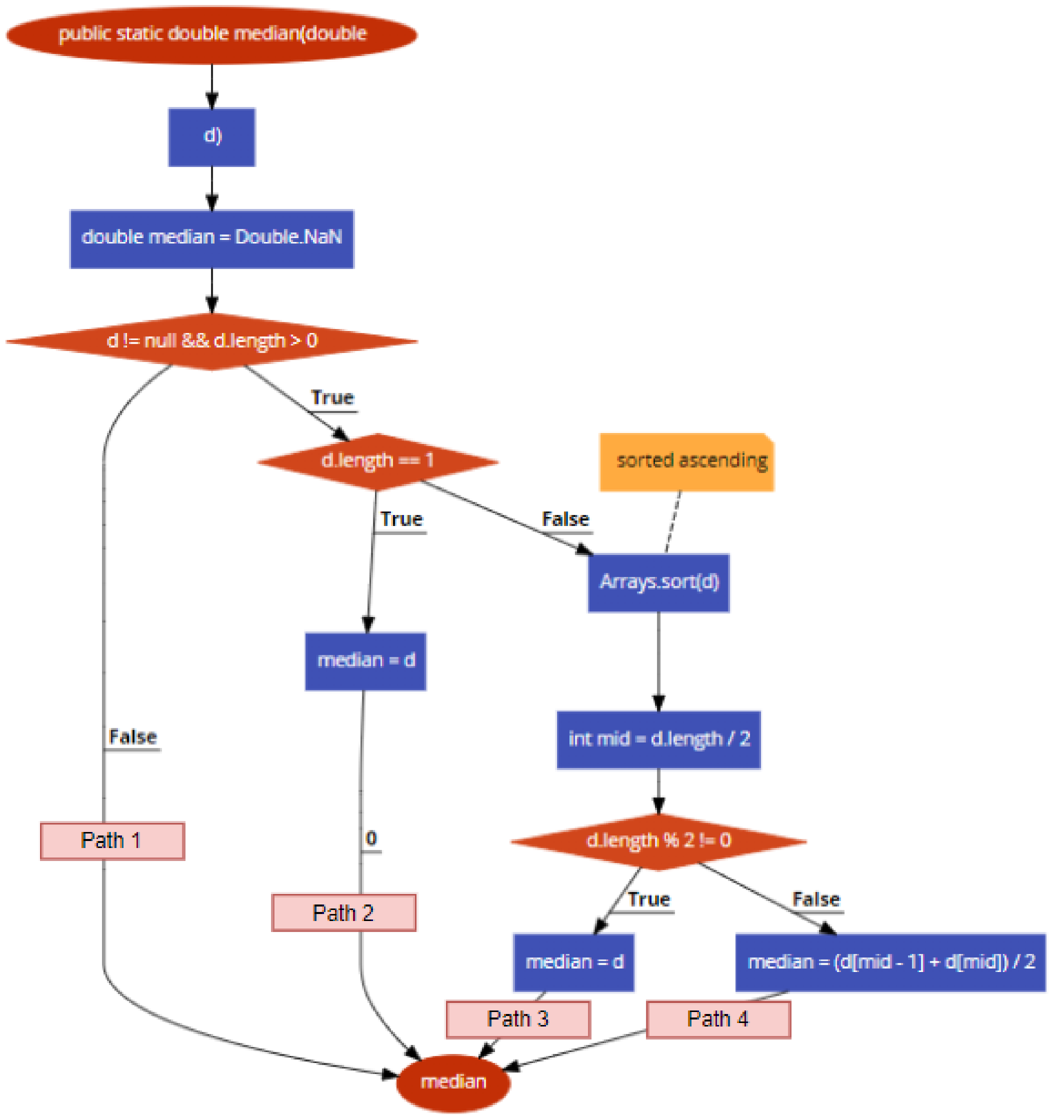
a).

The && operator can make code more efficient due to the fact that & operator evaluate all expressions while && operator only evaluate next expression if the current expression return true.

In our case, & operator evaluate (d!null) and (d.length>0)

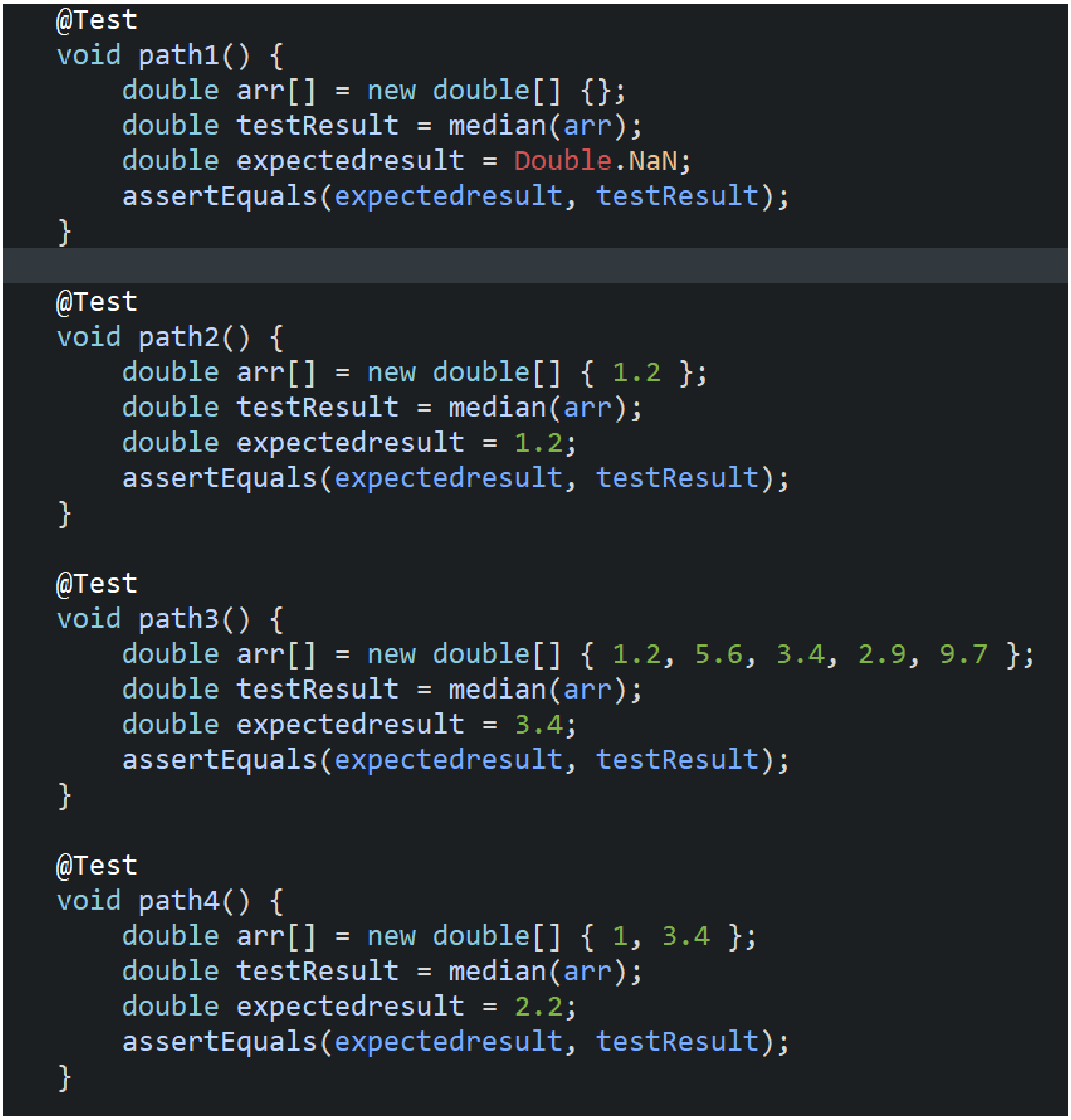
&& evaluate (d.length>0) if (d!null) returns true.

b).



c).

I have labeled Path 1,2,3,4 in the above graph.



```java
@Test
void path1() {
    double arr[] = new double[] {};
    double testResult = median(arr);
    double expectedresult = Double.NaN;
    assertEquals(expectedresult, testResult);
}

@Test
void path2() {
    double arr[] = new double[] { 1.2 };
    double testResult = median(arr);
    double expectedresult = 1.2;
    assertEquals(expectedresult, testResult);
}

@Test
void path3() {
    double arr[] = new double[] { 1.2, 5.6, 3.4, 2.9, 9.7 };
    double testResult = median(arr);
    double expectedresult = 3.4;
    assertEquals(expectedresult, testResult);
}

@Test
void path4() {
    double arr[] = new double[] { 1, 3.4 };
    double testResult = median(arr);
    double expectedresult = 2.2;
    assertEquals(expectedresult, testResult);
}
```