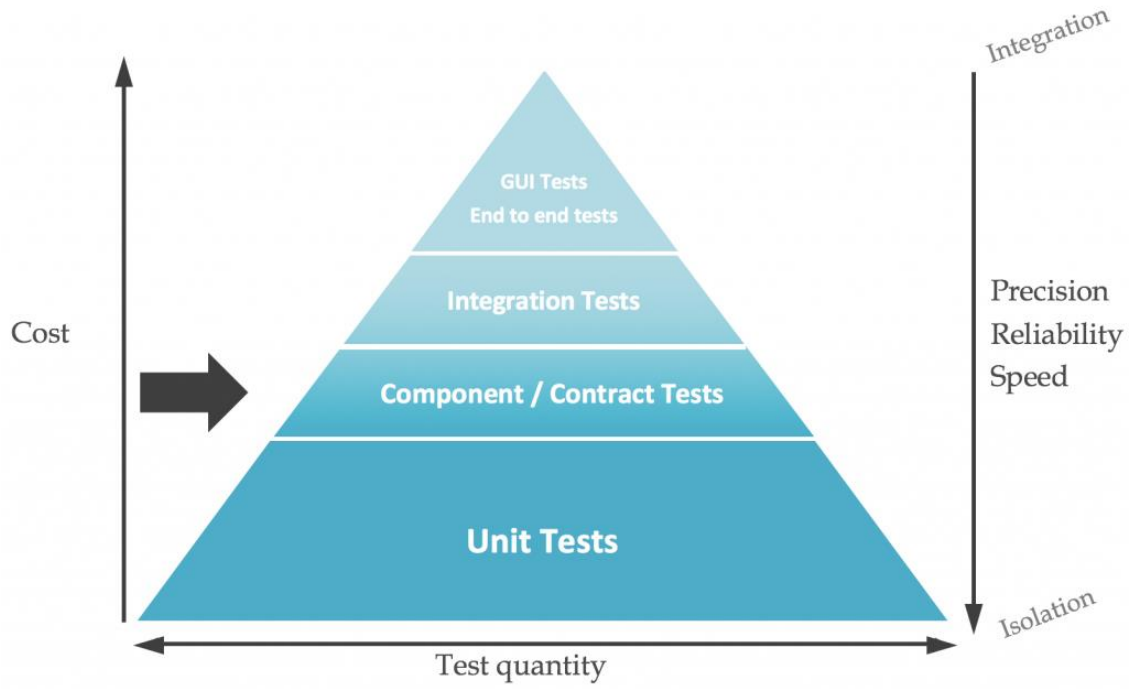
Vienetų testavimas ir kodo padengimas testais

Dr. Asta Slotkienė

Unit test

- Test individual pieces in isolation
- Focused, low-level; test individual methods or pieces of methods.
- Control: easier **to ensure that each piece of code is tested**



Unit test

- A unit test is an **automated piece of code** that invokes a unit of work in the system. And a unit of work can span a **single method**, a whole **class** or **multiple classes** working together to achieve one single logical purpose that can be verified.

```
public class MyUnit {
        public String concatenate(String one, String two){
        return one + two;
} }
```

```
import org.junit.Test;
import static org.junit.Assert.*;
public class MyUnitTest {

@Test
public void testConcatenate() {
         MyUnit myUnit = new MyUnit();
         String result = myUnit.concatenate("one", "two");
         assertEquals("onetwo", result);
} }
```

Python vieneto testo pavyzdys

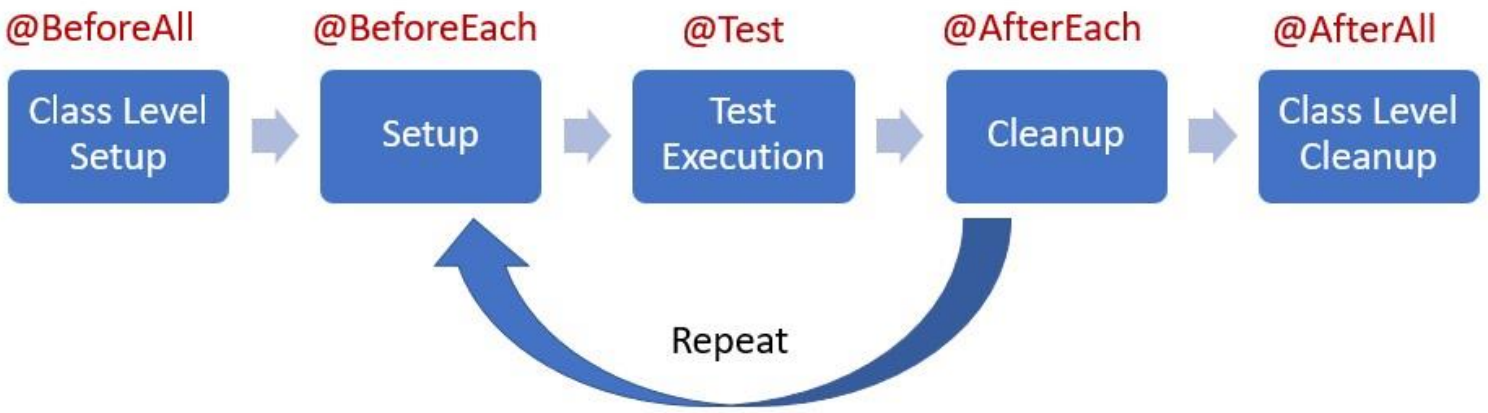
```
import unittest

class Calculator:
    def __init__(self):
        pass
    def add(self, a, b):
        return a + b

class TestCalculator(unittest.TestCase):
    def test_add(self):
        self.calc = Calculator()
        result = self.calc.add(4, 7)
        expected = 11
        self.assertEqual(result, expected)
```

Unit test framework

- Python:
  - https://docs.python.org/3.9/library/unittest.html#test-cases
- Java
  - https://junit.org/junit5/docs/current/user-guide/

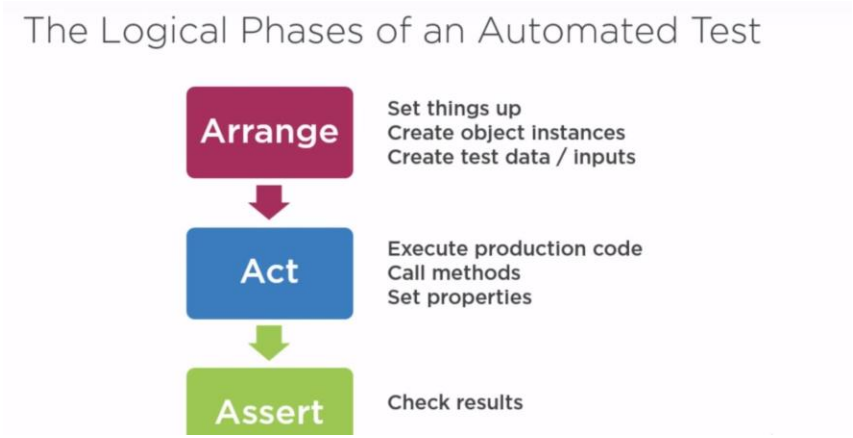


Assert...

- assertEqual(a,b) a == b
- assertNotEqual(a,b) a != b
- assertTrue(x) - bool(x) is True
- assertFalse(x) - bool(x) is False
- assertIs(a, b) a is b
- assertIsNot(a, b) a is not b
- assertIsNone(x) x is None
- assertIsNotNone(x) x is not None
- assertIn(a, b) a in b
- assertNotIn(a, b) a not in b
- assertGreater(a, b) a > b
- assertLess(a, b) a < b

Unit test pattern

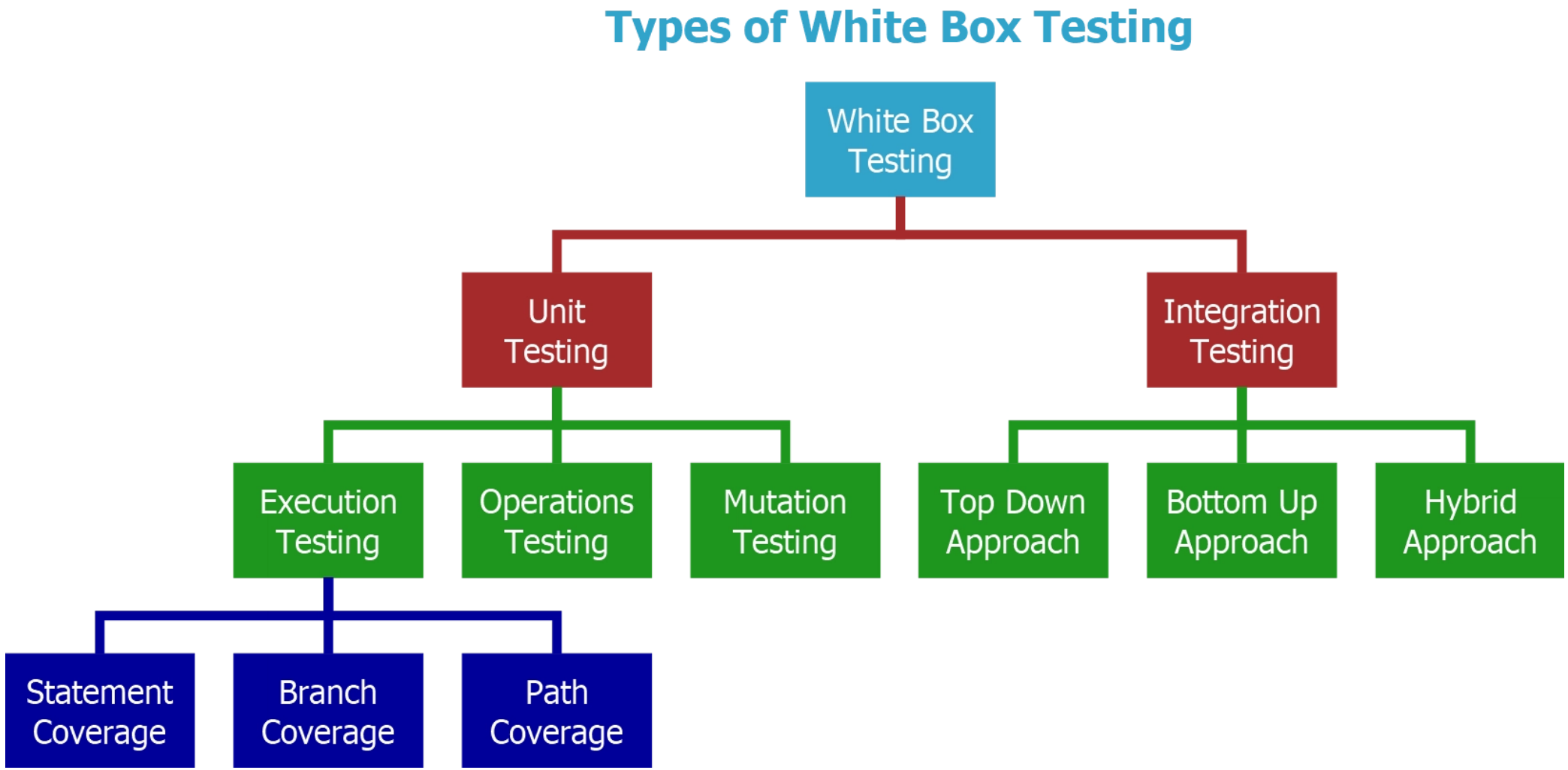
- **The Arrange-Act-Assert pattern**




```
@Test
  void testPlus() {
    //Arrange
    var cash = new Cash(3);
    //Act
    cash.plus(4);
    //Assert
    assertEquals(7, cash.count());
  }
```

https://dzone.com/articles/unit-testing-patterns-common-patterns-to-follow-fo

Baltos dėžės testavimas



Įvesties->Išvesties testavimas: kur problema?

```
public static int numZero(int[] x) {
 int count=0;
 for (int i=1; i<x.length; i++){
     if (x[i]==0)
          count++;
      }
 return count;
}
```

Įvestis: x=[5, 10, 0]

Tikėtinas rezultatas=?

Gautas rezultatas=?

Įvesties->Išvesties testavimas

```
public static int numZero(int[] x) {
  int count=0;
  for (int i=1; i<x.length; i++){
      if (x[i]==0)
           count++;
       }
  return count;
}
```

Įvestis: x=[5, 10, 0]

Tikėtinas rezultatas=1

Gautas rezultatas=1

Įvesties->Išvesties testavimas

```
public static int numZero(int[] x) {
  int count=0;
  for (int i=1; i<x.length; i++){
      if (x[i]==0)
            count++;
       }
 return count;
}
```

Input: x=[5, 10, 0]

Expected result=1

Actual result=1

Error=YES

Failure=NO

Įvesties->Išvesties testavimas

```
public static int numZero(int[] x) {
  int count=0;
  for (int i=1; i<x.length; i++){
      if (x[i]==0)
           count++;
       }
  return count;
}
```

Įvestis: x=[0, 5, 10]

Tikėtinas rezultatas=?

Gautas rezultatas=?

Įvesties->Išvesties testavimas

```
public static int numZero(int[] x) {
  int count=0;
  for (int i=1; i<x.length; i++){
      if (x[i]==0)
          count++;
       }
  return count;
}
```

Įvestis: x=[0, 5, 10]

Tikėtinas rezultatas=1

Gautas rezultatas=?

Įvesties->Išvesties testavimas

```
public static int numZero(int[] x) {
  int count=0;
  for (int i=1; i<x.length; i++){
      if (x[i]==0)
           count++;
       }
  return count;
}
```

Įvestis: x=[0, 5, 10]

Tikėtinas rezultatas=1

Gautas rezultatas=0

Failure=YES

Vieneto testavimo algortimas

1) Pasirinkti kodo dalį
2) Pasirinkti testinį duomenį/is
3) Numatyti tikėtiną rezultatą
4) Vykdyti testą
5) Palyginti gautus rezultatus su tikėtinais

Kodo padengimas testais

- Kodo padengimas testais tai procentinis rodiklis, parodantis, kokia kodo dalis yra ištestuojama testais (unit tests)

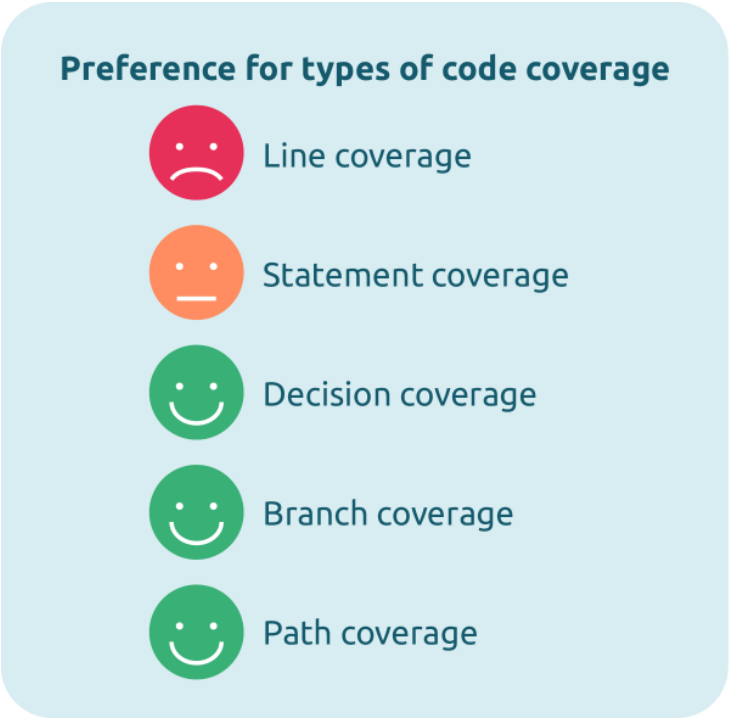
- *Code coverage* is the extent to which a given test suite executes the source code of the software.

Testiniai atvejai

- Perėjimai, galimi sprendimo keliai (Control flow)
  - Grafai
- Duomenų srautai (Data flow)
  - Įvesties duomenys
  - Konfigūraciniai duomenys
  - Generuojami duomenys

Baltos dėžės testavimo technikos

- Sakinių (statement, Node) padengimas
- Išsišakojimų padengimas (branch=!desicions)
- Ciklo padengimai
- Sąlygų padengimai
- Sprendimo kelių padengimas (Path)
- Įvairios kombinacijos



Padengimas sakiniais (statements)

- Skaičiuojama kiek kodo sakinių padengė įvykdymas testas
- Kodo padengiamumas=įvykdytų sakinių kiekis/visų sakinių kiekis
- Kodo padengiamumas=3/5 ->60%

```
metai = int(input().strip())
if metai<85:
    print(metai, "draudziamas")
else:
    print(metai, "nedraudziamas")
```

```
56
56 draudziamas
```

Programos išsišakojimų (branch) padengimas

- Išsišakojimų padengimas=įvykdyti išsisakojimai/visų išsisakojimų kiekis

```
sk = int(input().strip())
daliklis= int(input().strip())
if daliklis!=0:
  rez=sk/daliklis
  print("Rezulatas", rez)
else:
  print("Dalyba negalima")
print ("Ivesti duomenys dalybai:", "skaicius ", sk, " daliklis ", daliklis)
```

Programos išsišakojimų (branch) padengimas


```
sk = int(input().strip())
daliklis= int(input().strip())
if daliklis!=0:
  rez=sk/daliklis
  print("Rezulatas", rez)
else:
  print("Dalyba negalima")
print ("Ivesti duomenys dalybai:", "skaicius ", sk, " daliklis ", daliklis)
```

Programos išsišakojimų(branch) padengimas

```
result = (15, 5)
expected = 3
Padengiamumas>3/5 =60%
```


```python
sk = int(input().strip())
daliklis= int(input().strip())
if daliklis!=0:
  rez=sk/daliklis
  print("Rezulatas", rez)
else:
  print("Dalyba negalima")
print ("Ivesti duomenys dalybai:", "skaicius ", sk, " daliklis ", daliklis)
```

Programos išsišakojimų(branch) padengimas

```
result = (15, 0)
expected = “Dalyba negalima”
Padengiamumas>2/5 =40%
```


```
sk = int(input().strip())
daliklis= int(input().strip())
if daliklis!=0:
  rez=sk/daliklis
  print("Rezulatas", rez)
else:
  print("Dalyba negalima")
print ("Ivesti duomenys dalybai:", "skaicius ", sk, " daliklis ", daliklis)
```

Programos išsišakojimų(branch!=decisions) padengimas

```
1 sprendimas: 1->2>3 -> 50%
2 sprendimas: 4->5   -> 50%
```


```python
sk = int(input().strip())
daliklis= int(input().strip())
if sk>0:
  if daliklis!=0:
    rez=sk/daliklis
    print("Rezulatas", rez)
  else:
    print("Dalyba negalima")
print ("Ivesti duomenys dalybai:", "skaicius ", sk, " daliklis ", daliklis)
```

Užduotis

- Įvestis-> 100, 100
- Rezultatas-> Klaida, 200
- Sakinių padengimas ?/7
- Išsišakojimų padengimas
- Sprendimų padengimas

```
x= int(input().strip())
y= int(input().strip())
if   x - 100 <= 0 :
   if   y - 100 <= 0 :
      if   x + y - 200 == 0:
         print("Klaida")
print(x + y);
```

Užduotis

- TA1: 5 (int), 100 (int)
- TA2: neInt, 5 (int)
- TA3: 5 (int), neInt
- TA4: neInt, neInt

```
x= int(input().strip())
y= int(input().strip())
if  x - 100 <= 0 :
  if  y - 100 <= 0 :
    if  x + y - 200 == 0:
      print("Klaida")
print(x + y);
```

	TA1	TA2	TA3	TA4
X (sveikas skaičius)	T	K	T	K
X (sveikas skaičius)	T	T	K	K
Tikėtinas rezultatas	Apskaičiuojama	Error	Error	Error

Ar pakanka juodos dėžės technikos?

TA1: 100, 100

TA2: neInt, 5

TA3: 5, neInt

TA4: neInt, neInt

Sakinių padengimas 7/7=100%

```
x= int(input().strip())
y= int(input().strip())
if  x - 100 <= 0 :
  if  y - 100 <= 0 :
    if  x + y - 200 == 0:
      print("Klaida")
print(x + y);
```

Ar pakanka juodos dėžės technikos?

TA1: 5, 100

TA2: neInt, 5

TA3: 5, neInt

TA4: neInt, neInt

- Sakinių padengimas 6/7=85%

```
x= int(input().strip())
y= int(input().strip())
if  x - 100 <= 0 :
  if  y - 100 <= 0 :
    if  x + y - 200 == 0:
      print("Klaida")
print(x + y);
```

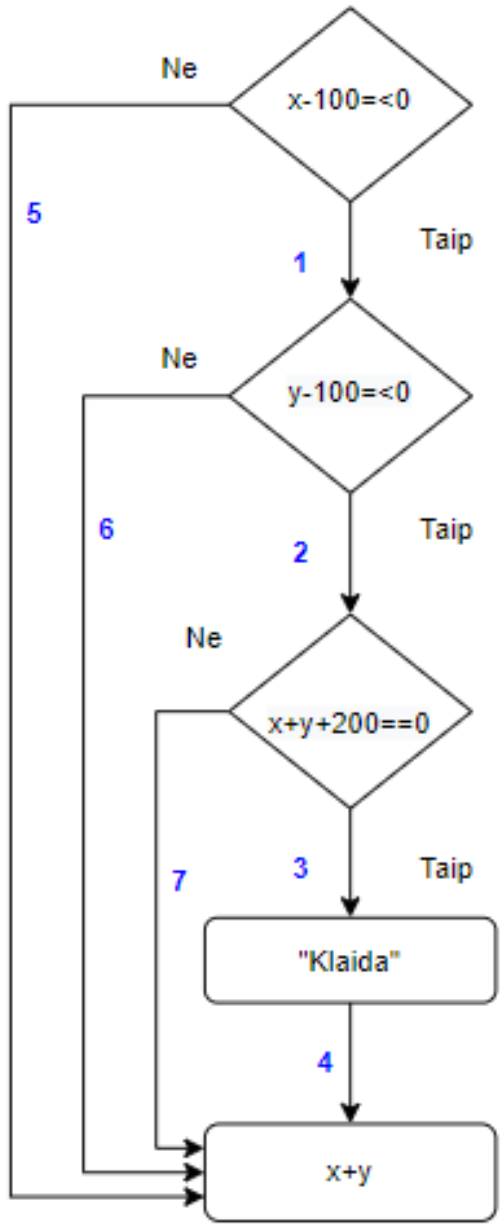
Nepakanka!

Sprendimas - > Taikyti Išsišakojimų ir sprendimų padengimą

Užduotis

- Įvestis-> 100, 100
- Sakinių padengimas: 7/7
- Išsišakojimų padengimas:
- Sprendimų padengimas:

```
x= int(input().strip())
y= int(input().strip())
if  x - 100 <= 0 :
  if  y - 100 <= 0 :
    if  x + y - 200 == 0:
      print("Klaida")
print(x + y);
```



	100, 100	5,100	-5, 200	200, 5
x-100=<0	100-100=0	5-100=-95<0	-5-100=-105<0	200-100=100>0
y-100=<0	100-100=0	100-100=0	200-100=100>0	
x+y+200==0	100+100-200=0	-95+100-200<>0		
"Klaida"	Klaida			
x+y	200	105	195	205

Užduotis

- Įvestis-> 100, 100
- Sakinių padengimas: 7/7
- Išsišakojimų padengimas: 4/7
- Sprendimų padengimas: 1/4

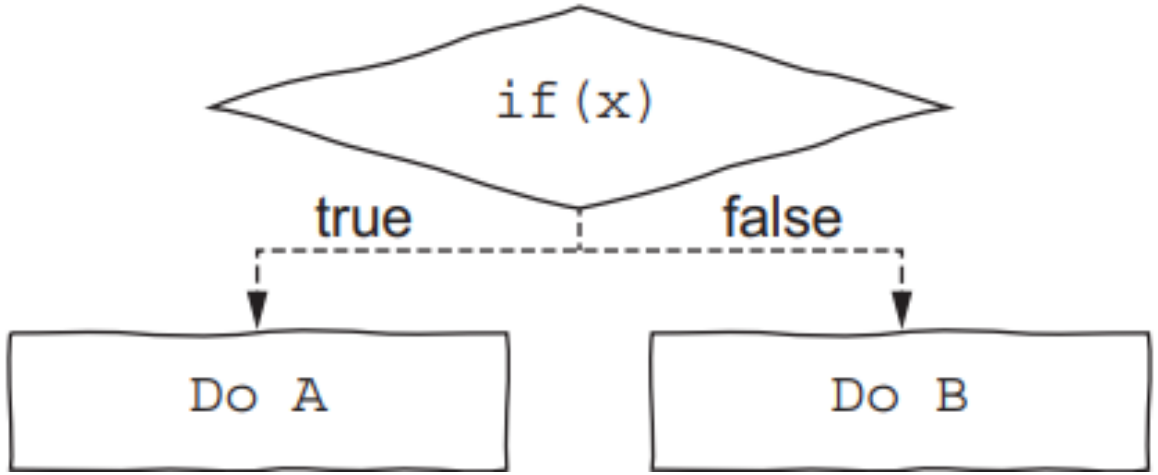
```
x= int(input().strip())
y= int(input().strip())
if  x - 100 <= 0 :
  if  y - 100 <= 0 :
    if  x + y - 200 == 0:
      print("Klaida")
print(x + y);
```

Užduotis

```
x= int(input().strip())
y= int(input().strip())
if  x - 100 <= 0 :
  if  y - 100 <= 0 :
    if  x + y - 200 == 0:
      print("Klaida")
print(x + y);
```

Kodo padengiamumas	100, 100	5,100	-5, 100	200,5
Sakinių padengimas	100% (7/7)	86% (6/7)	71% (5/7)	28% (2/7)
Išsišakojimų padengimas	57% (4/7)	57% (3/7)	57% (4/7)	57% (1/7)
Sprendimų padengimas	25% (1/4)	25% (1/4)	25% (1/4)	25% (1/4)

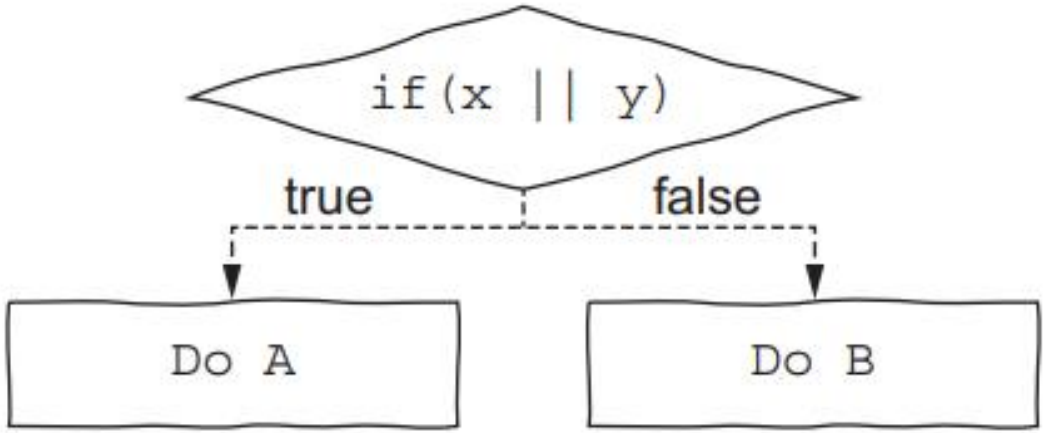
Branch coverage



Suppose a test T1 makes the `if` true.

- Line coverage: 2/3 = 66.6%
- Branch coverage: 1/2 = 50%

Condition + Branch coverage

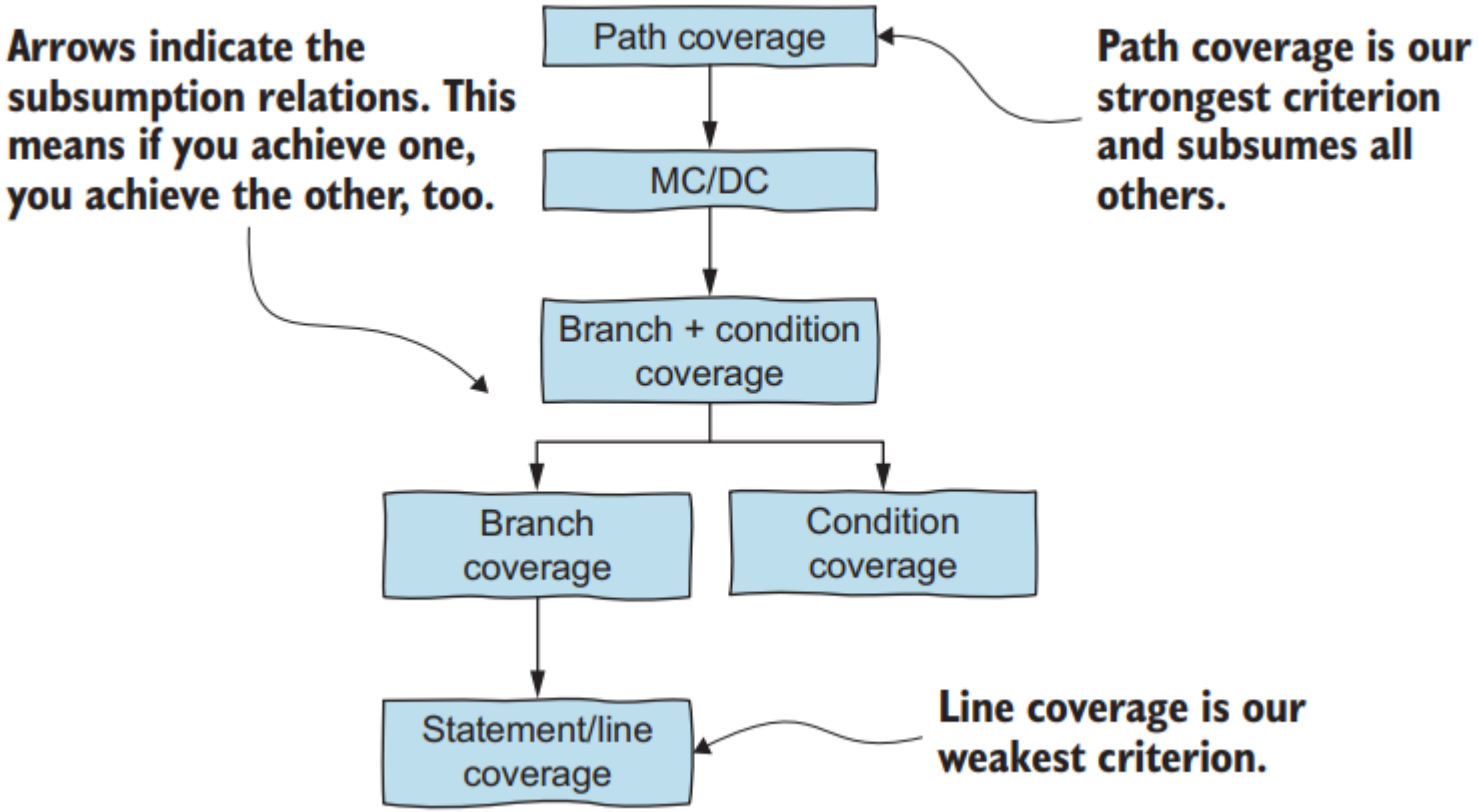


Imagine a test T1 where `x = true`.

- Line coverage: 2/3 = 66.6%
- Branch coverage: 1/2 = 50%
- Branch + condition coverage: (1 + 2)/(2 + 4) = 50%

$$\text{c+b coverage} = \frac{\text{branches covered} + \text{conditions covered}}{\text{number of branches} + \text{number of conditions}} \times 100\%$$

The different coverage criteria



Advanced Condition Coverage

- Condition/Decision Coverage (C/DC)
  - as DC plus: **every condition in each decision is tested in each possible outcome**
- Modified Condition/Decision coverage (MC/DC)
  - as above plus, **every condition shown to independently affect a decision outcome** (by varying that condition only)
  - Def: A condition independently affects a decision when, by flipping that condition's outcome and holding all the others fixed, the decision outcome changes
  - this criterion was created at Boeing and is required for aviation software according to RCTA/DO-178B
- Multiple-Condition Coverage (M-CC)
  - **all possible combinations of condition outcomes within each decision is checked**

Modified condition/decision coverage

MC/DC

*"**Each condition** in a decision has been shown to independently affect that **decision's outcome.** A condition is shown to independently affect a decision's outcome by varying just that condition while holding fixed all other possible conditions".*

Modified condition/decision coverage *MC/DC*

Complete requirements for an MC/DC test suite:

- All program **entry and exit points covered**
- **Each decision exercised on both branches**
- Each condition takes **both values**
- Each condition is shown to affect its enclosing decision

if (a and b) or c

Test Case	a	b	c	outcome
1	True	True	True	True
2	True	True	False	True
3	True	False	True	True
4	True	False	False	False
5	False	True	True	True
6	False	True	False	False
7	False	False	True	True
8	False	False	False	False

if (a and b) or c: **from a value**

Test Case	a	b	c	outcome
1	True	True	True	True
2	True	True	False	True
3	True	False	True	True
4	True	False	False	False
5	False	True	True	True
6	False	True	False	False
7	False	False	True	True
8	False	False	False	False

Test Case	**a**	b	c	outcome
2	True	True	False	True
6	False	True	False	False

if (a and b) or c: **from b value**

Test Case	a	b	c	outcome
1	True	True	True	True
2	True	True	False	True
3	True	False	True	True
4	True	False	False	False
5	False	True	True	True
6	False	True	False	False
7	False	False	True	True
8	False	False	False	False

Test Case	a	**b**	c	outcome
2	True	True	False	True
4	True	False	False	False

if (a and b) or c: **from c value**

Test Case	a	b	c	outcome
1	True	True	True	True
2	True	True	False	True
3	True	False	True	True
4	True	False	False	False
5	False	True	True	True
6	False	True	False	False
7	False	False	True	True
8	False	False	False	False

Test Case	a	b	**c**	outcome
3	True	False	True	True
4	True	False	False	False

if (a and b) or c

Required N+1 test cases: T2, T3, T4. T6

Test Case	a	b	c	outcome
1	True	True	True	True
2	True	True	False	True
3	True	False	True	True
4	True	False	False	False
5	False	True	True	True
6	False	True	False	False
7	False	False	True	True
8	False	False	False	False

Example

- Counts the number of words in a string that end with either “r” or “s”

- Given a sentence, the program should count the number of words that end with either “s” or “r”. A word ends when a non-letter appears. The program returns the number of words

```
public class CountWords {
        public int count(String str) {
                int words = 0;
                char last = ' ';
                 for (int i = 0; i < str.length(); i++) {
                        if (!isLetter(str.charAt(i)) && (last == 's' || last == 'r')) {
                                words++;
                                }
                        last = str.charAt(i); }
                        if (last == 'r' || last == 's') {
                         words++;
}
                return words;
} }
```

Unit testai 1

```
@Test
void twoWordsEndingWithS() {
	int words = new CountLetters().count("dogs cats");
	assertThat(words).isEqualTo(2);
}
@Test
void noWordsAtAll() {
	int words = new CountLetters().count("dog cat");
	assertThat(words).isEqualTo(0);
}
```

Code Coverage

```
public class CountWords {
    public int count(String str) {
        int words = 0;
        char last = ' ';
        for (int i = 0; i < str.length(); i++) {
            if (!Character.isLetter(str.charAt(i)) && (last == 's' || last == 'r')) {
                words++;
            }
            last = str.charAt(i);
        }
        if (last == 'r' || last == 's')
            words++;
        return words;
    }
}
```

Unit testai 12

```
@Test
void wordsThatEndInR() {
	int words = new CountWords().count("car bar");
	assertThat(words).isEqualTo(2);
 }
```

Code Coverage

```
public class CountWords {
    public int count(String str) {
        int words = 0;
        char last = ' ';
        for (int i = 0; i < str.length(); i++) {
            if (!Character.isLetter(str.charAt(i)) && (last == 's' || last == 'r')) {
                words++;
            }
            last = str.charAt(i);
        }
        if (last == 'r' || last == 's')
            words++;
        return words;
    }
}
```

Modified condition/decision coverage

Test case	`isLetter`	`last == s`	`last == r`	`decision`
T1	true	true	true	true
T2	true	true	false	true
T3	true	false	true	true
T4	true	false	false	false
T5	false	true	true	false
T6	false	true	false	false
T7	false	false	true	false
T8	false	false	false	false

https://www.youtube.com/watch?v=DivaWCNohdw

https://en.wikipedia.org/wiki/Modified_condition/decision_coverage

isLetter: T1 opposite T5: (T1;T5)

Test case	isLetter	last == s	last == r	decision
T1	true	true	true	true
T2	true	true	false	true
T3	true	false	true	true
T4	true	false	false	false
T5	false	true	true	false
T6	false	true	false	false
T7	false	false	true	false
T8	false	false	false	false

last == s or last == r : isLetter: opposite (T2;T6)

Test case	isLetter	last == s	last == r	decision
T1	true	true	true	true
T2	true	true	false	true
T3	true	false	true	true
T4	true	false	false	false
T5	false	true	true	false
T6	false	true	false	false
T7	false	false	true	false
T8	false	false	false	false

last == s or last == r : isLetter: opposite (T3;T7)

Test case	isLetter	last == s	last == r	decision
T1	true	true	true	true
T2	true	true	false	true
T3	true	false	true	true
T4	true	false	false	false
T5	false	true	true	false
T6	false	true	false	false
T7	false	false	true	false
T8	false	false	false	false

last == s or last == r : isLetter: opposite

~~(T4;T8): both the same outcome~~

Test case	isLetter	last == s	last == r	decision
T1	true	true	true	true
T2	true	true	false	true
T3	true	false	true	true
T4	true	false	false	false
T5	false	true	true	false
T6	false	true	false	false
T7	false	false	true	false
T8	false	false	false	false

MC/DC coverage: N + 1 tests
isLetter:
opposite (T1;T5), (T2;T6), (T3;T7)

Test case	isLetter	last == s	last == r	decision
T1	true	true	true	true
T2	true	true	false	true
T3	true	false	true	true
T4	true	false	false	false
T5	false	true	true	false
T6	false	true	false	false
T7	false	false	true	false
T8	false	false	false	false

MC/DC coverage: N + 1 tests

isLetter:

opposite (T1;T5), (T2;T6), (T3;T7)->T6 or T7

Test case	isLetter	last == s	last == r	decision
T1	true	true	true	true
T2	true	true	false	true
T3	true	false	true	true
T4	true	false	false	false
T5	false	true	true	false
T6	false	true	false	false
T7	false	false	true	false
T8	false	false	false	false

MC/DC coverage: N + 1 tests

T2, T3, T4, T6 or T7

Test case	`isLetter`	`last == s`	`last == r`	`decision`
T1	true	true	true	true
T2	true	true	false	true
T3	true	false	true	true
T4	true	false	false	false
T5	false	true	true	false
T6	false	true	false	false
T7	false	false	true	false
T8	false	false	false	false

Papildomai

- Reinforced Condition/Decision Coverage (RC/DC)
- ISTQB Technical Test Analyst | 2.4 Modified Condition/Decision Coverage (MC/DC) Testing
  - https://www.youtube.com/watch?v=9i9xpxn6pzM