

ITP2200

Introduction to Software Testing

Bogdan Marculescu

Lecture 4

Logic Coverage

Input Space Partitioning

So, more programming this week?



What do we remember from last time?



We mentioned “Coverage”

```
4  public class TriangleClassifier {  
5      @  
6          public static String classify(int a, int b, int c) {  
7              if (a <= 0 || b <= 0 || c <= 0) {  
8                  return "NOT_A_TRIANGLE";  
9              }  
10             if (a == b && b == c) {  
11                 return "EQUILATERAL";  
12             }  
13             int max = Math.max(a, Math.max(b, c));  
14             if ((max == a && max - b - c >= 0) ||  
15                 (max == b && max - a - c >= 0) ||  
16                 (max == c && max - a - b >= 0)) {  
17                 return "NOT_A_TRIANGLE";  
18             }  
19             if (a == b || b == c || a == c) {  
20                 return "ISOSCELES";  
21             } else {  
22                 return "SCALENE";  
23             }  
24         }  
25     }  
26 }  
27 }  
28 }  
29 }  
30 }
```

```
@Test  
public void testScalene() throws Exception {  
    int a = 5;  
    int b = 7;  
    int c = 9;  
  
    String result = TriangleClassifier.classify(a, b, c);  
    assertTrue(result.equalsIgnoreCase( anotherString: "SCALENE"));  
}  
  
@Test  
public void testIsosceles() throws Exception{  
    int a = 5;  
    int b = 5;  
    int c = 7;  
  
    String result = TriangleClassifier.classify(a, b, c);  
    assertTrue(result.equalsIgnoreCase( anotherString: "ISOSCELES"));  
    assertFalse(result.equalsIgnoreCase( anotherString: "SCALENE"));  
}
```

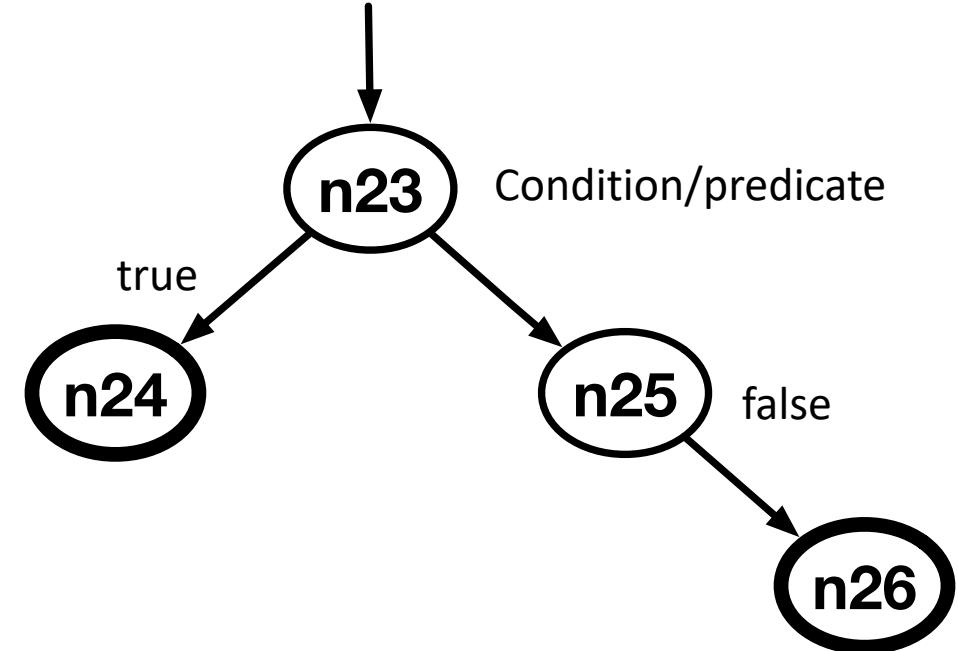
Code is at

<https://github.com/bogdanmarculescu/itp2200>

We mentioned “Coverage”

```
if (a == b || b == c || a == c) {  
    return "ISOSCELES";  
} else {  
    return "SCALENE";  
}
```

Code is at
<https://github.com/bogdanmarculescu/itp2200>

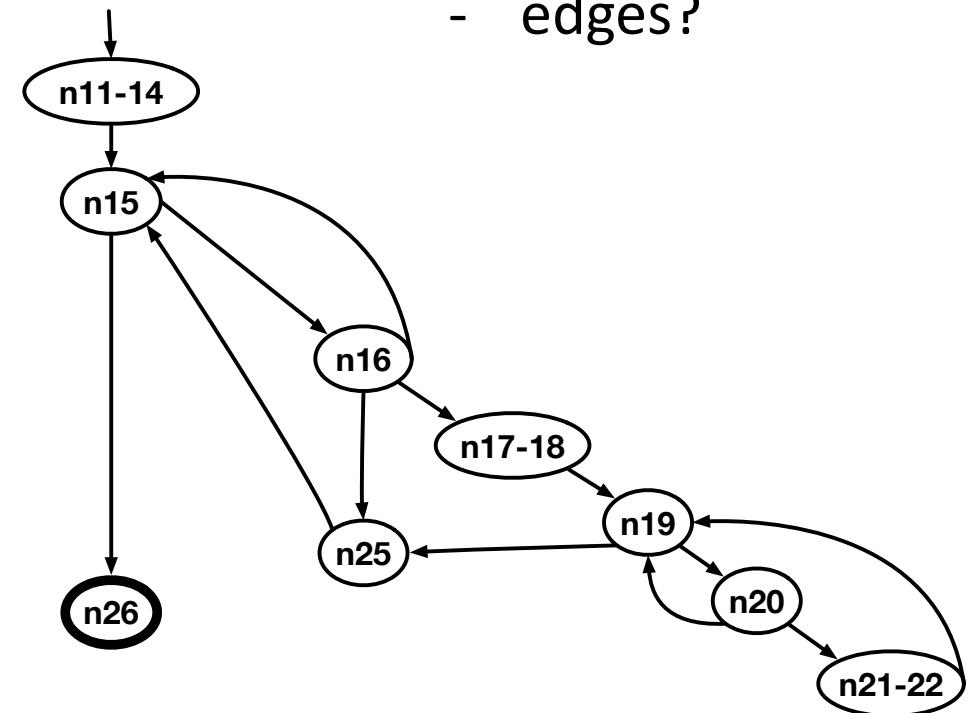


Code!!!

```
public class PatternMatcher {  
    public static int pat (char[] subject, char[] pattern)  
    {  
        // Post: if pattern is not a substring of subject, return -1  
        // else return (zero-based) index where the pattern (first)  
        // starts in subject  
        final int NOTFOUND = -1;  
        int iSub = 0, rtnIndex = NOTFOUND; boolean isPat = false;  
        int subjectLen = subject.length;  
        int patternLen = pattern.length;  
        while (isPat == false && iSub + patternLen - 1 < subjectLen) {  
            if (subject[iSub] == pattern[0]) {  
                rtnIndex = iSub; // Starting at zero  
                isPat = true;  
                for (int iPat = 1; iPat < patternLen; iPat++) {  
                    if (subject[iSub + iPat] != pattern[iPat]) {  
                        rtnIndex = NOTFOUND; isPat = false;  
                        break; // out of for loop  
                    } }  
                iSub++; }  
        return (rtnIndex); }  
}
```

How do we improve coverage?

- nodes?
- edges?



Logical predicates and clauses

Predicates (and their expression in java)

- Negation: !variable
- And: var1 && var2
- Or: var1 || var2
- Xor: var1 ^ var2
- A condition evaluates to **true** or **false**
- **If, while, for** – all contain conditions
- E.g.:
isPat == false && iSub + patternLen -1 < subjectLen

```
public class PatternMatcher {  
    public static int pat (char[] subject, char[] pattern)  
    {  
        // Post: if pattern is not a substring of subject, return -1  
        // else return (zero-based) index where the pattern (first)  
        // starts in subject  
        final int NOTFOUND = -1;  
        int iSub = 0, rtnIndex = NOTFOUND; boolean isPat = false;  
        int subjectLen = subject.length;  
        int patternLen = pattern.length;  
        while (isPat == false && iSub + patternLen - 1 < subjectLen) {  
            if (subject [iSub] == pattern [0]) {  
                rtnIndex = iSub; // Starting at zero  
                isPat = true;  
            for (int iPat = 1; iPat < patternLen; iPat++) {  
                if (subject[iSub + iPat] != pattern[iPat]) {  
                    rtnIndex = NOTFOUND; isPat = false;  
                    break; // out of for loop  
                } }  
            iSub++; }  
        return (rtnIndex); }
```

Logical predicates and clauses

Pred => isPat == false && x < y

Predicate coverage: pred is true, pred is false
(isPat == false and x < y , isPat == true)

Clause Coverage

isPat == true, x > y

isPat == false, x < y

Combinatorial coverage:

All possible combinations (8 in all)

BUT (it isn't x < y, is it?)

It's iSub + patternLen - 1 < subjectLen

```
public class PatternMatcher {  
    public static int pat (char[] subject, char[] pattern)  
    {  
        // Post: if pattern is not a substring of subject, return -1  
        // else return (zero-based) index where the pattern (first)  
        // starts in subject  
        final int NOTFOUND = -1;  
        int iSub = 0, rtnIndex = NOTFOUND; boolean isPat = false;  
        int subjectLen = subject.length;  
        int patternLen = pattern.length;  
        while (isPat == false && iSub + patternLen - 1 < subjectLen) {  
            if (subject[iSub] == pattern[0]) {  
                rtnIndex = iSub; // Starting at zero  
                isPat = true;  
                for (int iPat = 1; iPat < patternLen; iPat++) {  
                    if (subject[iSub + iPat] != pattern[iPat]) {  
                        rtnIndex = NOTFOUND; isPat = false;  
                        break; // out of for loop  
                    } }  
                iSub++; }  
        return (rtnIndex); }  
}
```

Logical predicates and clauses

```
for (int iPat = 1; iPat < patternLen; iPat++)
```

Can be rewritten:

```
int iPat = 1;  
while( iPat < patternLen){  
    x = y/pattern[iPat];  
    iPat++;  
}
```

```
public class PatternMatcher {  
    public static int pat (char[] subject, char[] pattern)  
    {  
        // Post: if pattern is not a substring of subject, return -1  
        // else return (zero-based) index where the pattern (first)  
        // starts in subject  
        final int NOTFOUND = -1;  
        int iSub = 0, rtnIndex = NOTFOUND; boolean isPat = false;  
        int subjectLen = subject.length;  
        int patternLen = pattern.length;  
        while (isPat == false && iSub + patternLen - 1 < subjectLen) {  
            if (subject [iSub] == pattern [0]) {  
                rtnIndex = iSub; // Starting at zero  
                isPat = true;  
            }  
            for (int iPat = 1; iPat < patternLen; iPat++) {  
                if (subject[iSub + iPat] != pattern[iPat]) {  
                    rtnIndex = NOTFOUND; isPat = false;  
                    break; // out of for loop  
                } }  
            iSub++; }  
        return (rtnIndex); }  
}
```

Logical predicates and clauses

```
for (int iPat = 1; iPat < patternLen; iPat++)
```

Can be rewritten:

```
int iPat = 1;  
while( iPat < patternLen && pattern[iPat]) {  
    x = y/pattern[iPat];  
    iPat++;  
}
```

-> Controllability comes in

```
public class PatternMatcher {  
    public static int pat (char[] subject, char[] pattern)  
    {  
        // Post: if pattern is not a substring of subject, return -1  
        // else return (zero-based) index where the pattern (first)  
        // starts in subject  
        final int NOTFOUND = -1;  
        int iSub = 0, rtnIndex = NOTFOUND; boolean isPat = false;  
        int subjectLen = subject.length;  
        int patternLen = pattern.length;  
        while (isPat == false && iSub + patternLen - 1 < subjectLen) {  
            if (subject [iSub] == pattern [0]) {  
                rtnIndex = iSub; // Starting at zero  
                isPat = true;  
                for (int iPat = 1; iPat < patternLen; iPat++) {  
                    if (subject[iSub + iPat] != pattern[iPat]) {  
                        rtnIndex = NOTFOUND; isPat = false;  
                        break; // out of for loop  
                    } }  
            }  
            iSub++; }  
        return (rtnIndex); }  
}
```



Høyskolen
Kristiania



Questions so far?

Back to triangles – value selection

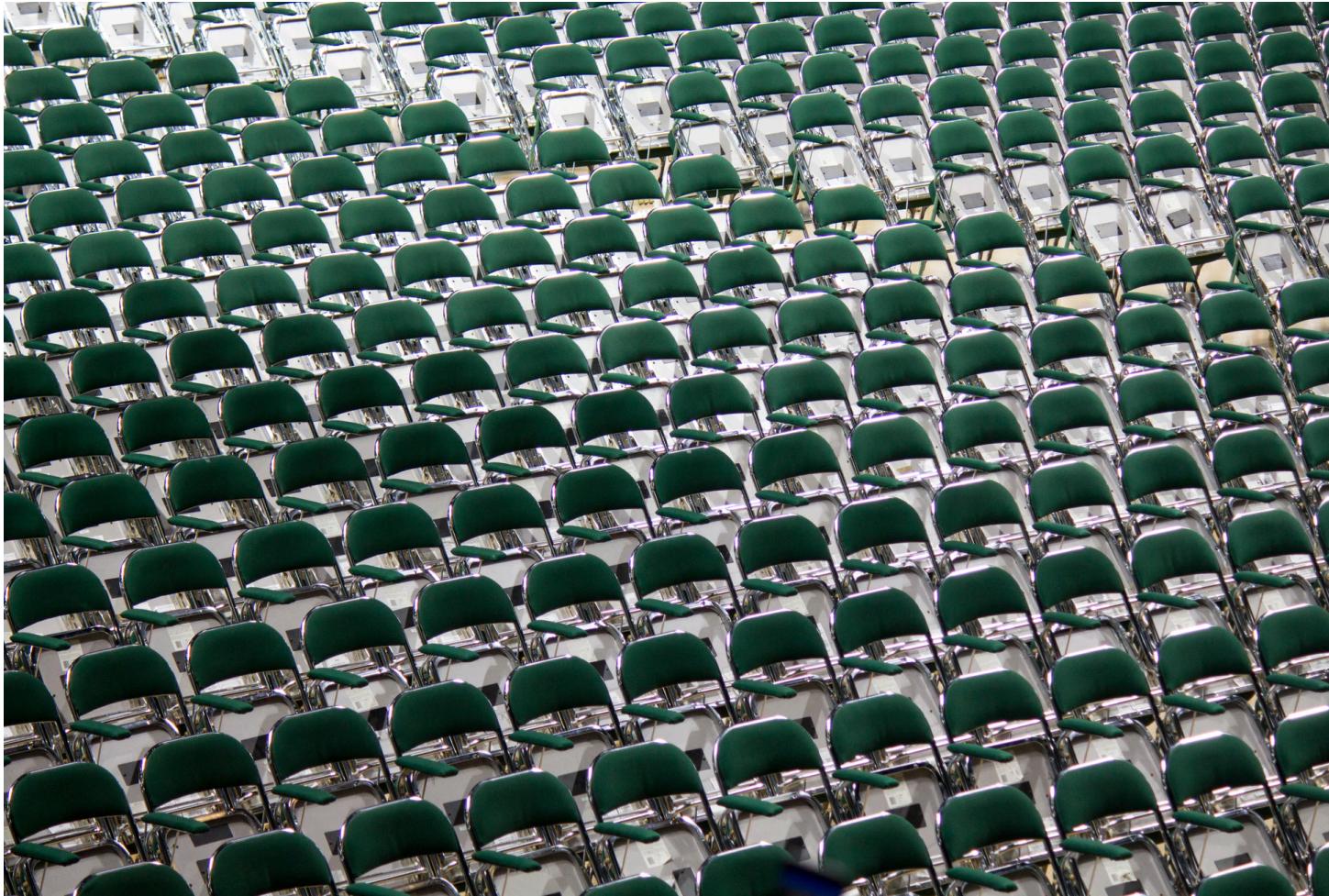
```
5  public class TriangleClassifier {  
6      @ ...  
7      public static String classify(int a, int b, int c) {  
8          if (a <= 0 || b <= 0 || c <= 0) {  
9              return "NOT_A_TRIANGLE";  
10         }  
11  
12         if (a == b && b == c) {  
13             return "EQUILATERAL";  
14         }  
15  
16         int max = Math.max(a, Math.max(b, c));  
17  
18         if ((max == a && max - b - c >= 0) ||  
19             (max == b && max - a - c >= 0) ||  
20             (max == c && max - a - b >= 0)) {  
21             return "NOT_A_TRIANGLE";  
22         }  
23  
24         if (a == b || b == c || a == c) {  
25             return "ISOSCELES";  
26         } else {  
27             return "SCALENE";  
28         }  
29     }  
30 }
```

8: $a \leq 0 \mid\mid b \leq 0 \mid\mid c \leq 0$
12: $a == b \&\& b == c$
18: $(\max == a \&\& \max - b - c \geq 0)$
19: $(\max == b \&\& \max - a - c \geq 0)$
20: $(\max == c \&\& \max - a - b \geq 0)$
18a : 18 $\mid\mid$ 19 $\mid\mid$ 20

24a: $a == b$
24b: $b == c$
24c: $a == c$

24: 24a $\mid\mid$ 24b $\mid\mid$ 24c

Input Space Partitioning



Domain: the space of all possible input values

Blocks: disjoint (i.e. non-overlapping) areas of the domain that are “equivalent” in some way

Partitioning: the process of splitting the domain into blocks

Input Space Partitioning

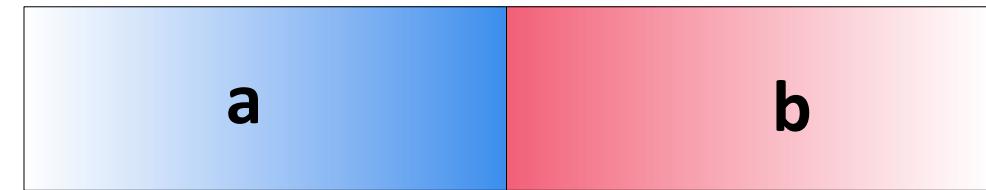
Equivalent

- similar **behaviour, output,**
- By some **criterion**

Equivalence classes:

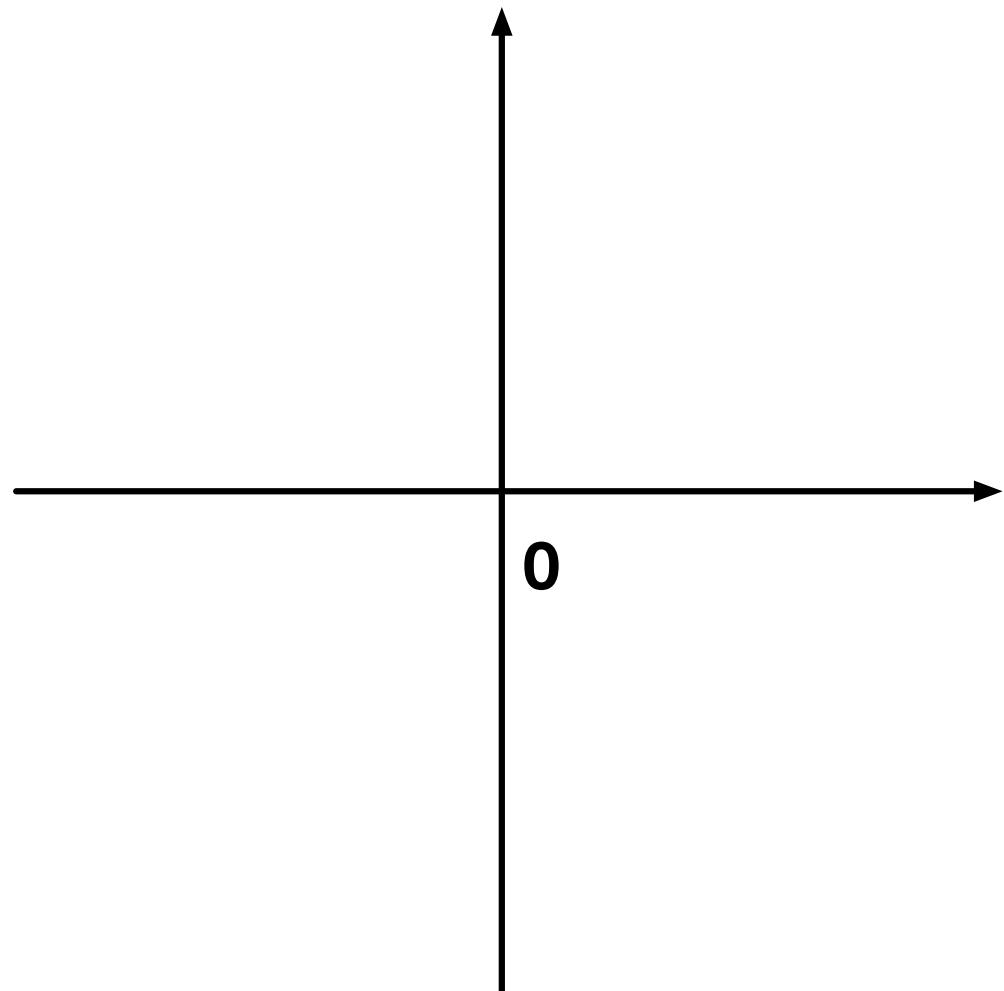
Sections of the input space (i.e. collections of inputs) that result in equivalent behaviour from the system

```
public static int categorize(int input){  
    int result = 0;  
    if (input > 0) {  
        result = 1; //Return 1 if the number is positive  
    }  
    else {  
        if (input < 0) result = -1; //Return -1 if the number is negative  
    }  
    return result;  
  
public static int absoluteValue(int input){  
    if(input >= 0) return input;  
    else return -input;  
}
```



Høyskolen
Kristiania

Input Space Partitioning

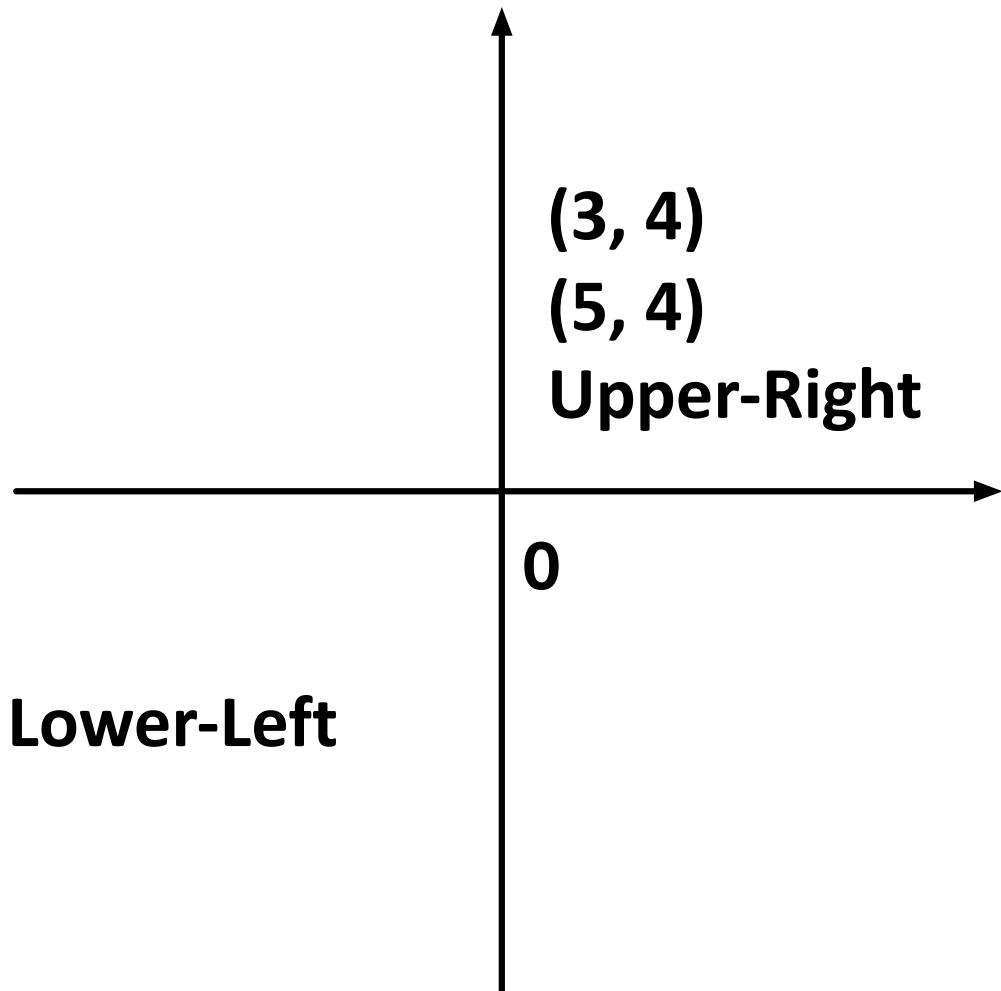


```
20 @
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39 }
```

```
public static String quadrants(int x, int y){
    if(x >= 0){
        if(y>= 0){
            return "Upper-Right";
        }
        else{
            return "Lower-Right";
        }
    }
    else {
        if(y>=0){
            return "Upper-Left";
        }
        else{
            return "Lower-Left";
        }
    }
}
```

(3, 4)
(5, 4)

Input Space Partitioning



```
20 @
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39 }
```

```
public static String quadrants(int x, int y){
    if(x >= 0){
        if(y>= 0){
            return "Upper-Right";
        }
        else{
            return "Lower-Right";
        }
    }
    else {
        if(y>=0){
            return "Upper-Left";
        }
        else{
            return "Lower-Left";
        }
    }
}
```

Back to triangles – equivalence classes

```
5  public class TriangleClassifier {  
6      @ ...  
7      public static String classify(int a, int b, int c) {  
8          if (a <= 0 || b <= 0 || c <= 0) {  
9              return "NOT_A_TRIANGLE";  
10         }  
11  
12         if (a == b && b == c) {  
13             return "EQUILATERAL";  
14         }  
15  
16         int max = Math.max(a, Math.max(b, c));  
17  
18         if ((max == a && max - b - c >= 0) ||  
19             (max == b && max - a - c >= 0) ||  
20             (max == c && max - a - b >= 0)) {  
21             return "NOT_A_TRIANGLE";  
22         }  
23  
24         if (a == b || b == c || a == c) {  
25             return "ISOSCELES";  
26         } else {  
27             return "SCALENE";  
28         }  
29     }  
30 }
```

What is the input domain?

How can we partition it?

Back to triangles – equivalence classes

```
5  public class TriangleClassifier {  
6      @ ...  
7      public static String classify(int a, int b, int c) {  
8          if (a <= 0 || b <= 0 || c <= 0) {  
9              return "NOT_A_TRIANGLE";  
10         }  
11  
12         if (a == b && b == c) {  
13             return "EQUILATERAL";  
14         }  
15  
16         int max = Math.max(a, Math.max(b, c));  
17  
18         if ((max == a && max - b - c >= 0) ||  
19             (max == b && max - a - c >= 0) ||  
20             (max == c && max - a - b >= 0)) {  
21             return "NOT_A_TRIANGLE";  
22         }  
23  
24         if (a == b || b == c || a == c) {  
25             return "ISOSCELES";  
26         } else {  
27             return "SCALENE";  
28         }  
29     }  
30 }
```

What is the input domain?

How can we partition it?

Given an example of test values per partition:

Equivalence Classes



Equivalence Classes in Testing

Valid values:

Sub-partition:

Boundaries:

Normal use:

Invalid values:

Balance:

Missing partitions:

Overlapping partitions:





Høyskolen
Kristiania



Questions so far?

Exercise for the Seminar



Rooms:

A3-01

A4-03

A5-18

Exercise for the Seminar



Note: you can save your existing work in a separate branch

git branch -b mywork

Self-organized.

In repo:

Ex04

For the methods in classes

TriangleClassifier

MathsAndStats

- Discuss the logical **predicates** (and what values would make them true and false)
- Define **input domain** and **partition** it (describe what blocks you find)
- Write at least one test case for each **equivalence class**

Exercise for the Seminar



Note: you can save your existing work in a separate branch

```
git branch -b mywork
```

Self-organized.

In repo:

Ex04

For the methods in classes

- Discuss the logical **predicates** (and what values would make them true and false)
- Define **input domain** and **partition** it (describe what blocks you find)
- Write at least one test case for each **equivalence class**

Advanced: PatternMatcher

Exercise for the Seminar



When you're done,

Reflect on how this differs from the paths discussed last time.

Did the examples you tested last time cover all the equivalence classes?

Note: you can save your existing work in a separate branch
`git branch -b mywork`