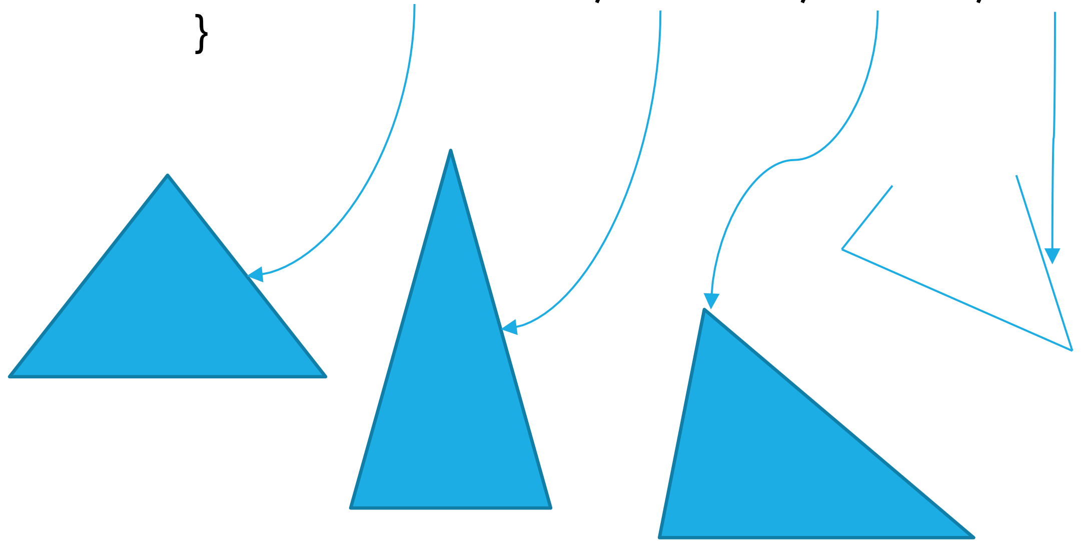
BLACKBOX TESTING

CS-HU 274 Lecture 4

GROUP WORK: TRIANGLE CLASSIFIER

```
enum TriangleType{
            EQUILATERAL, ISOSCELES, SCALENE, INVALID
}
```



```
static TtriangleType classify(int a, int b, int c)
```

What is the input domain for classify?

What are equivalent classes?

What are the boundaries between those classes?

List inputs for each equivalent class.

List inputs for boundary conditions.

Create test cases for those inputs in a text file.

Create JUnit test cases for those inputs.

Today

FINISHING THE TEST SPECIFICATION TABLE

For each specification create at least two test cases.

- "Inside" the equivalence partition (change in values in one critical side does not change the triangle's classification. For example, for a=5, b=5, c=3, changing c to 2 or to 4 keeps the classification the same
- One boundary test case (does not exist for Equilateral case). For example, the test case a=2, b=2, and c=1, changing c to 2 would make it equilateral and changing c to 0 would make it invalid.

At least15 test cases total.

JUNIT TEST CASES TRIANGLE CLASSIFIER

Update the repository

- week2 source folder
- week2_code contains an example on how to use the class
- wee2_code a Junit test case with two test cases
- files folder includes classifier.jar – contains the compiled class, so no code source is given
- classifier.jar is added to the project's classpath.

Remember inputs are not test cases!

Create test cases for those inputs in a text file.

Create JUnit test cases for those inputs.

This is a correct implementation so if a test case fails check the test case first.

JUNIT TEST CASES FOR TRIANGLE CLASSIFIER

How good are your test cases?

- Can they expose failures in incorrect implementation?
- Update the project with triangleFaulty.jar, uncomment the files in the main method.
- Regenerate/modify your test cases to execute classify method of TriangleClassifierFaulty class.
- How many of your test cases fail?
- Can you figure out what could be the faults in the code?

TEST-DRIVEN DEVELOPMENT

Goal is to rapidly crate code that is testable and correct.

Commonly used in agile development

Idea:

- Developers writes a JUnit test suite first (for a given interface)
- Programs until all test cases pass

Cycle: RED, GREEN, REFACTOR

- In the beginning all test cases fail (JUnit is in RED)
- Implements functionality to pass tests (JUnit is getting more GREEN)
- Tests are all pass (JUnit is in GREEN)
- Then the developer can optimize the code (i.e., REFACTOR)

TEST-DRIVEN DEVELOPMENT PRACTICE

Using test-cases for TriangleClassifier write your own triangle classifier program

- Implement functionality one a time
  1. INVALID case
  2. EQUILATERAL case
  3. ISOSCELES case
  4. SCALENE case
- After each implementation, document how many test cases pass and how many fail.
- Did any of your test cases fail?
- Did test cases help you with the implementation?

ASSIGNMENT 1

Similar what we did in class but on a larger scale:

- You will be given the program's jar files.
- Design category partition for the program's input.
- Identify boundary values.
- Design test cases for the program.
- Any test cases pass/fail?
- Document your experience.

Submit to Canvas, files are in the repository

Due before the next class.

The advice of the week:

make sure not import all java.util classes, e.g., java.util.*