



TECHNISCHE  
UNIVERSITÄT  
WIEN

# Welcome.TU.code

Dealing With Errors, Exceptions, Recursion

<https://github.com/votacom/it4refugees>

# Agenda

- Short Recap
- Exception Handling
- Recursion

# Recap Questions

**“Who can explain what a variable is?”**

# Recap Questions

**“Who can explain the difference between an declaration and initialisation of an variable?”**

# Recap Questions

**“Who can explain what an array is?”**

# Recap Questions

**“Who can explain how to use an array?”**

# Recap Questions

**“Who can explain when if-statements are used?”**

# Recap Questions

**“Who can name three different "loop" variations and explain how they work?”**



# Recap Questions

**“Who can explain what a function is? why do we use functions?”**

# Recap Questions

**“Who can explain how to declare a function?”**

# General Types of Errors in Java

- Syntax Errors
- Semantic Errors
- Runtime Errors

# Syntax Errors

Occurs due to incorrect grammar

- Spelling mistakes
- Missing semicolons
- Improperly matches parentheses

# Semantic Errors

This types of Errors indicate an improper use of the Java programming language.

- use of a non-initialized variable
- type incompatibility

# Runtime Errors

- Occur during execution of an programm
- “***Exception Handling***” deals with this types of errors

# Exceptions

- occur if something goes wrong
- often give a hint on the problem
- can be caught
- can be manually thrown
- can be created

# Types of Exceptions

- RuntimeException/Unchecked Exceptions
  - ArrayIndexOutOfBoundsException
  - NullPointerException
  - etc.
- Checked Exceptions (require catch block)
  - IOException
  - etc.



# Errors

- errors you can't really do anything about at runtime
- typically ignored in code, tried to avoid as good as possible
- e.g.: `StackOverflowError`

# Handle Exceptions

- try-Block: what you try to do
- catch-clause: which exceptions you want to catch, what to do with the information of these
- finally Block: always executed, even if exception occurs, used to close open file etc.
- or instead of all of this: 'throws' clause in function header

# Live Example

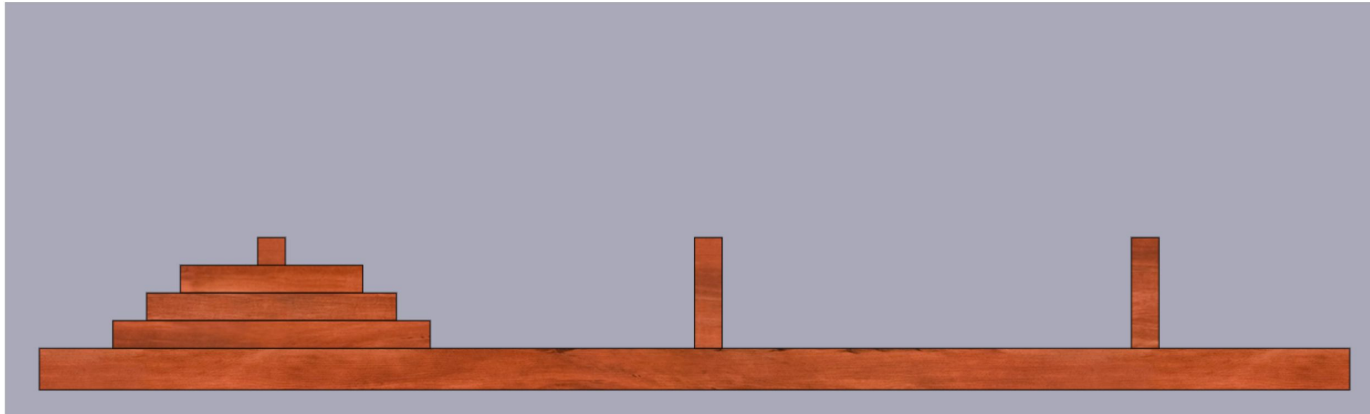
```
public static void main(String[] args) {  
    try {  
        //something causing an exception  
    } catch(Exception e) {  
        //what to do with the information of the exception  
    } finally {  
        //what you always want to do  
    }  
}
```

# Recursion

- calling your function in the same function again
- simplifying code for specific problems

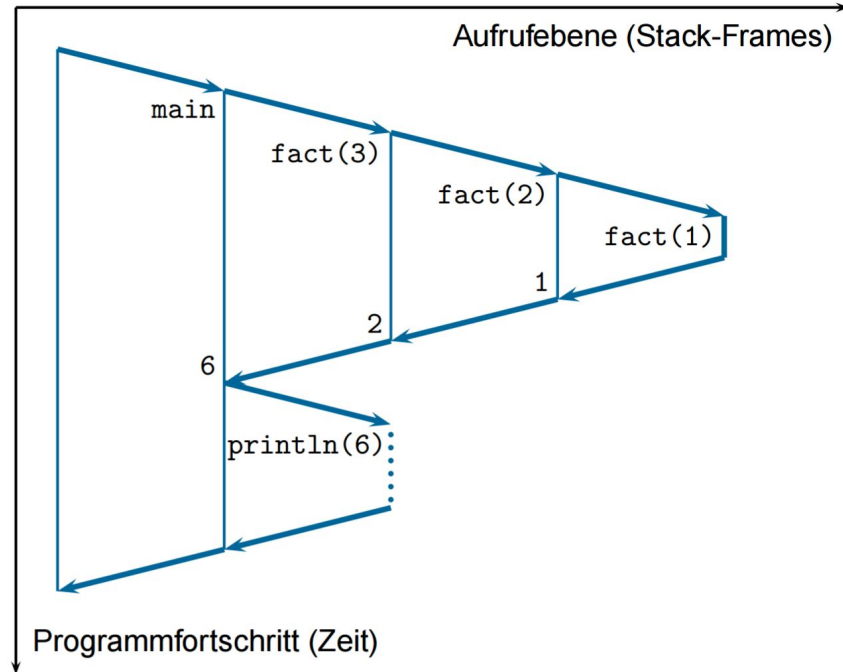
# Live Examples

- Towers of Hanoi

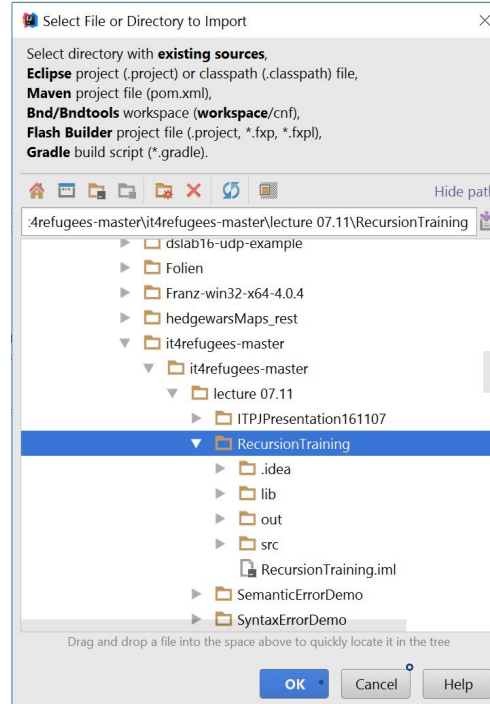
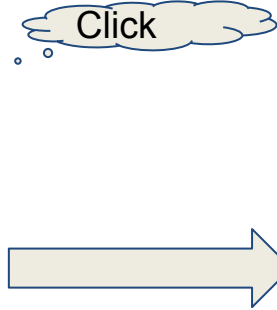
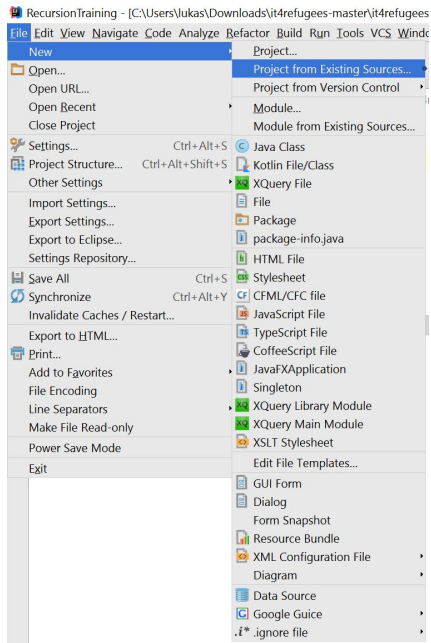


# Problems when using Recursion

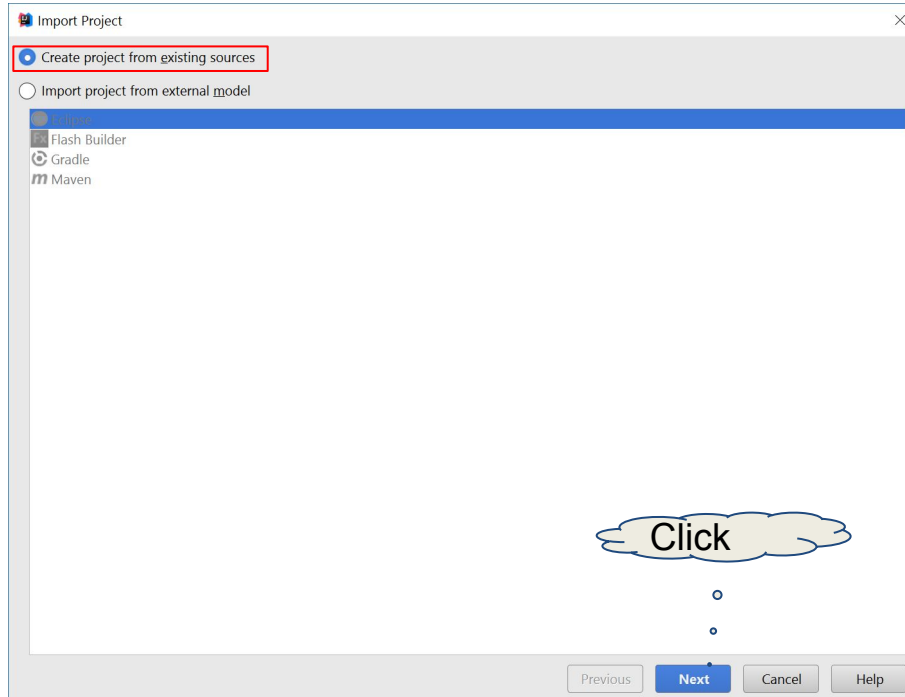
StackOverflowErrors



# How To Homework 1

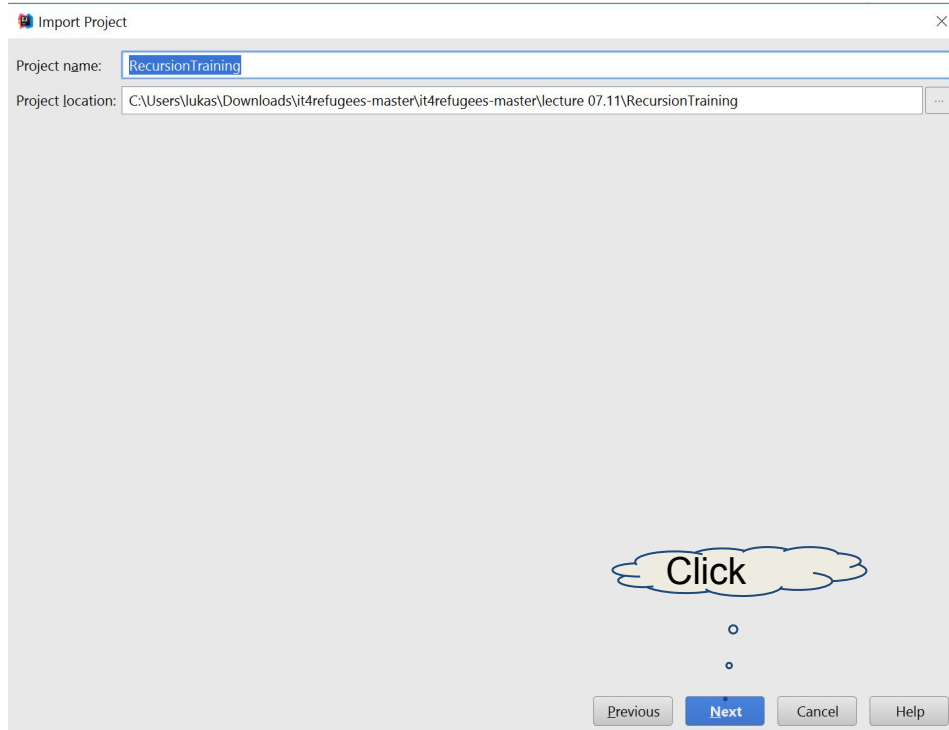


# How To Homework 2

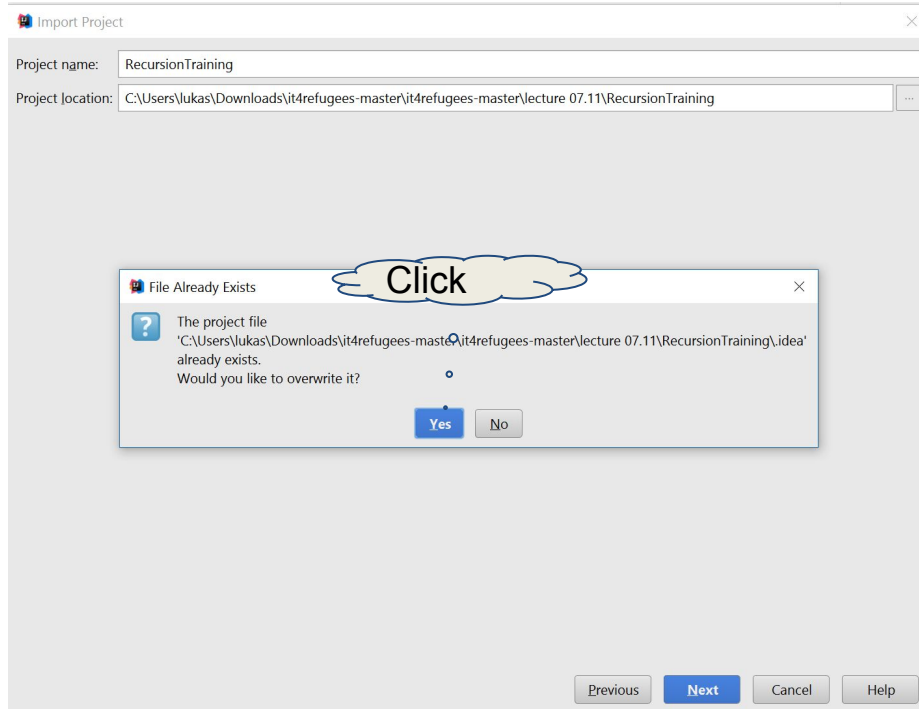




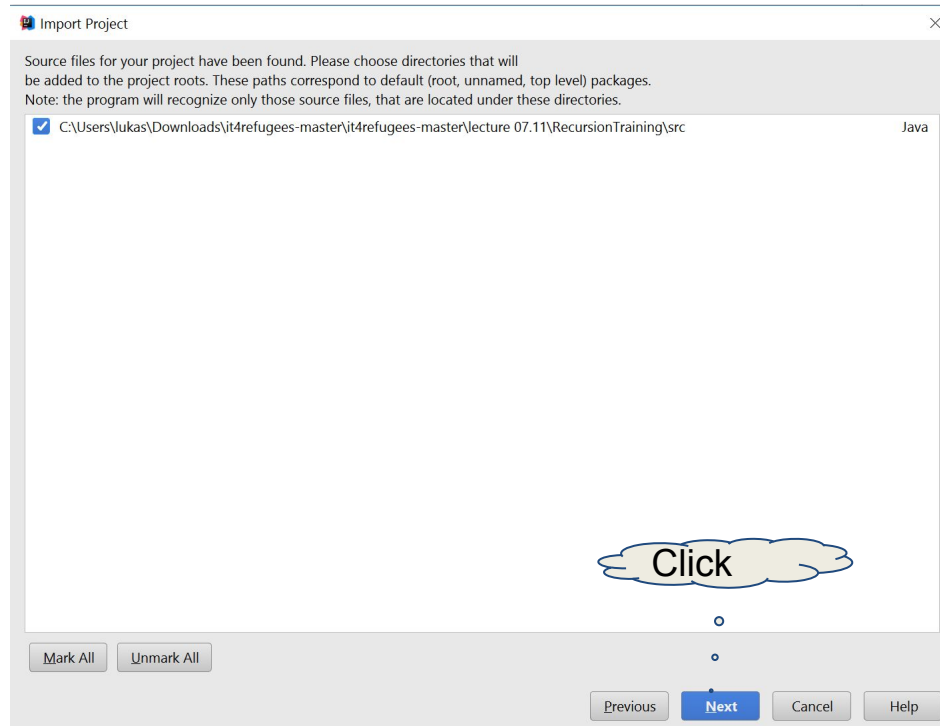
# How To Homework 3



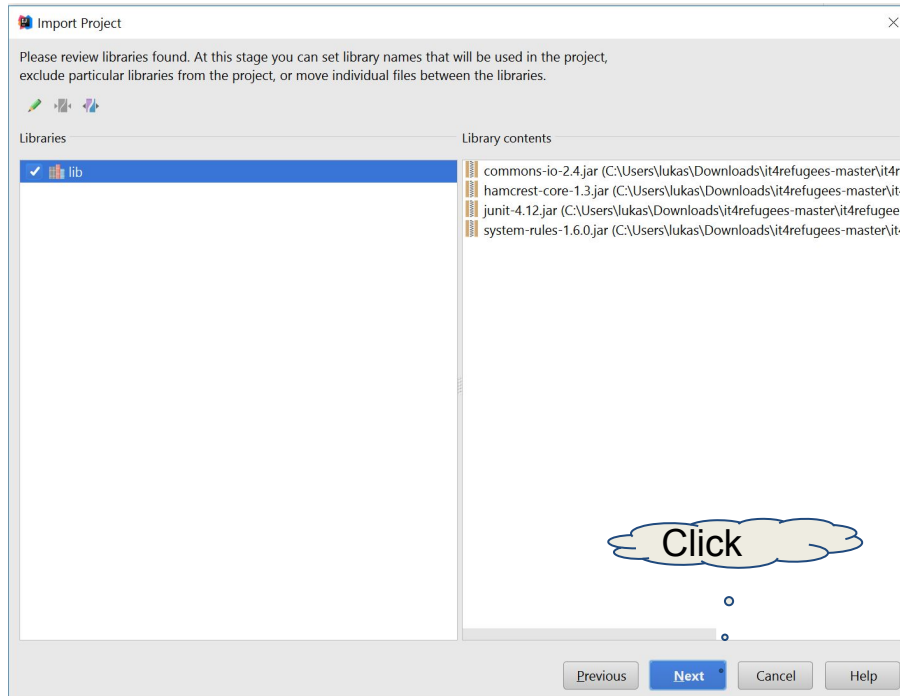
# How To Homework 4



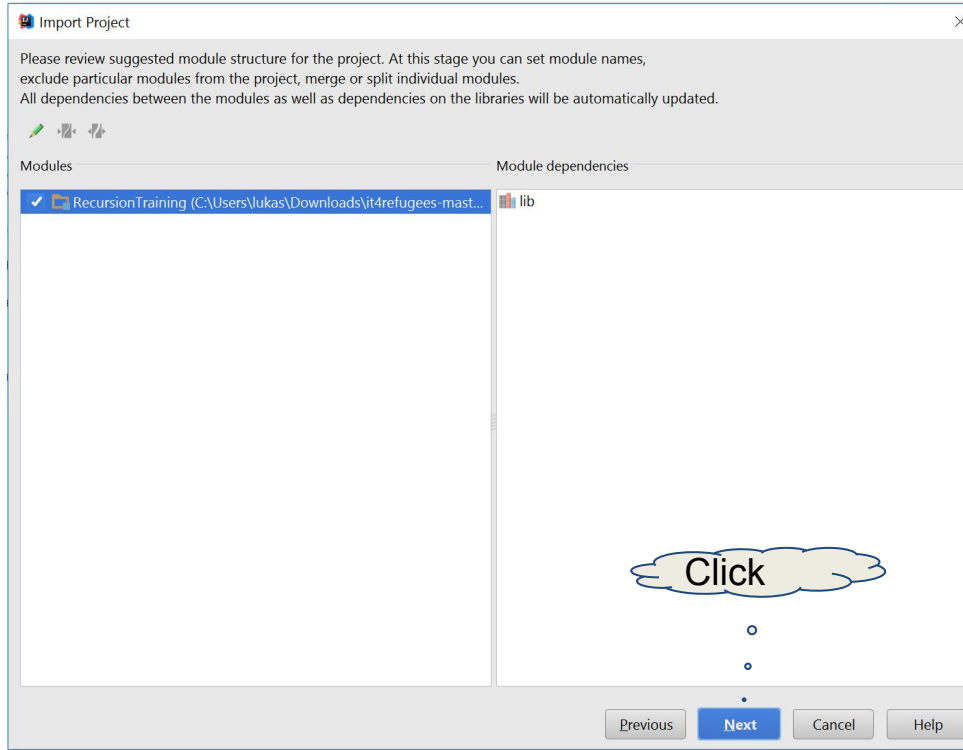
# How To Homework 5



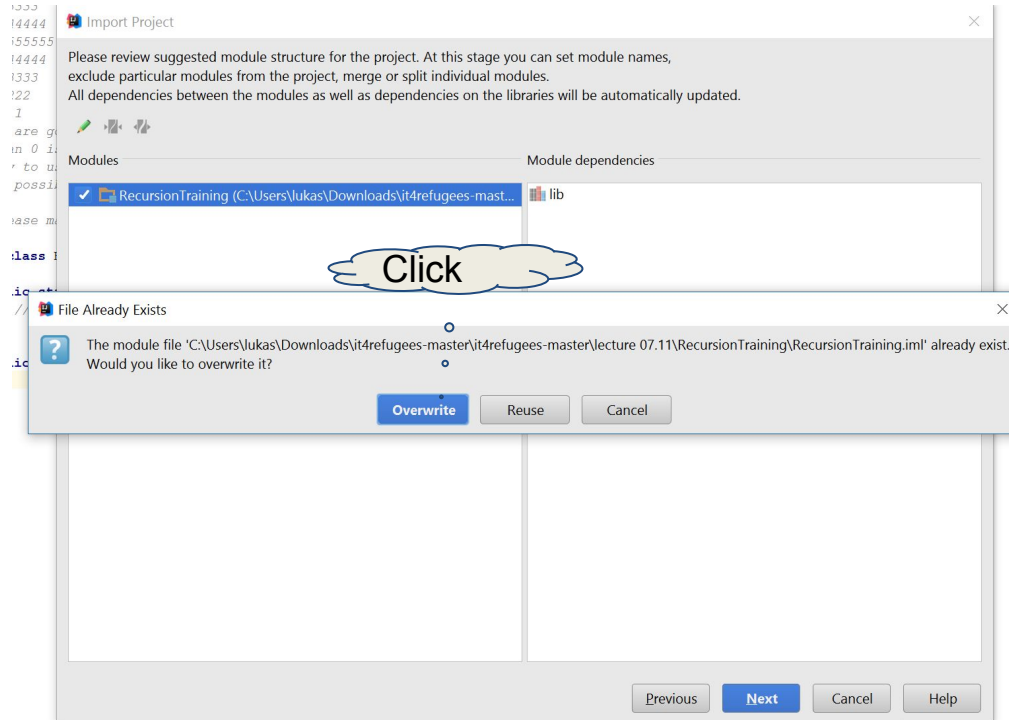
# How To Homework 6



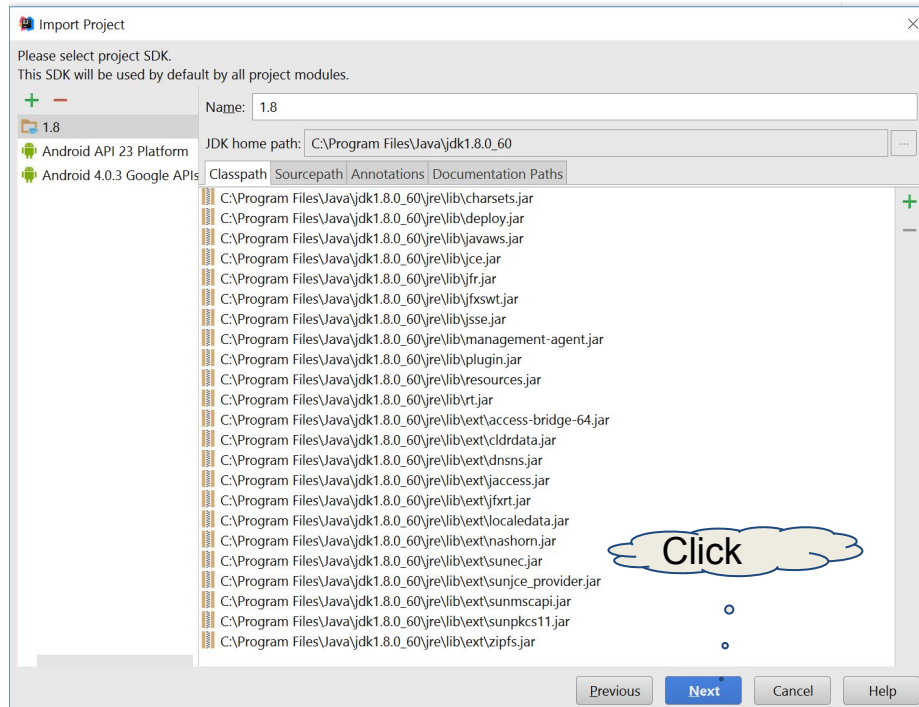
# How To Homework 7



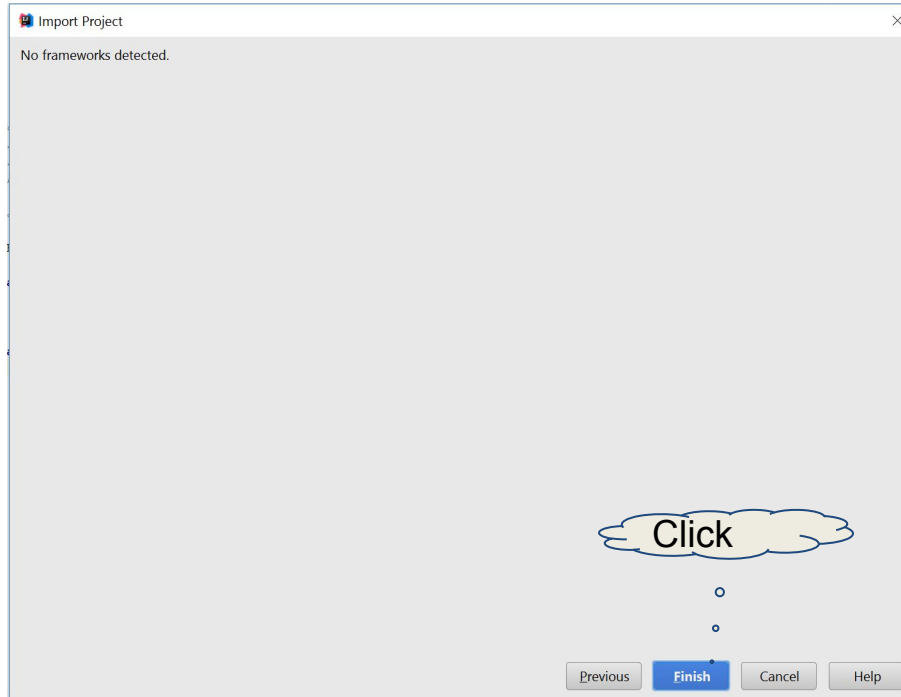
# How To Homework 8



# How To Homework 9

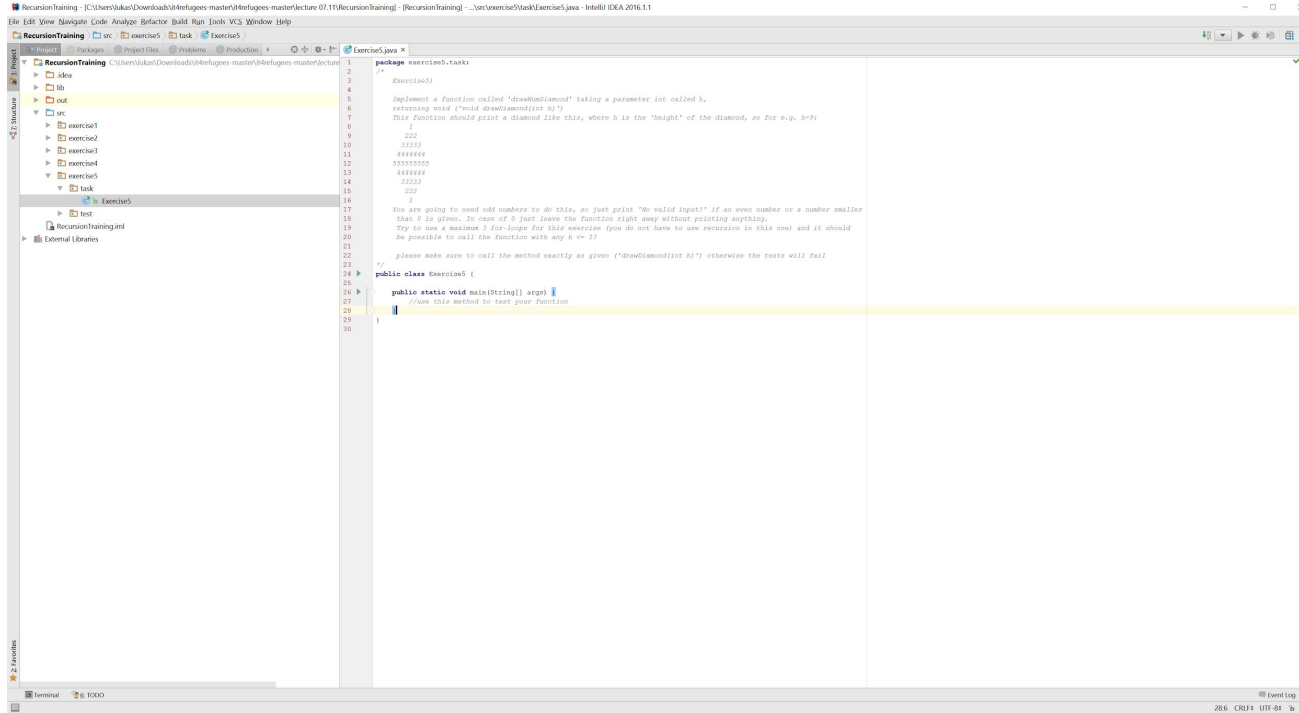


# How To Homework 10





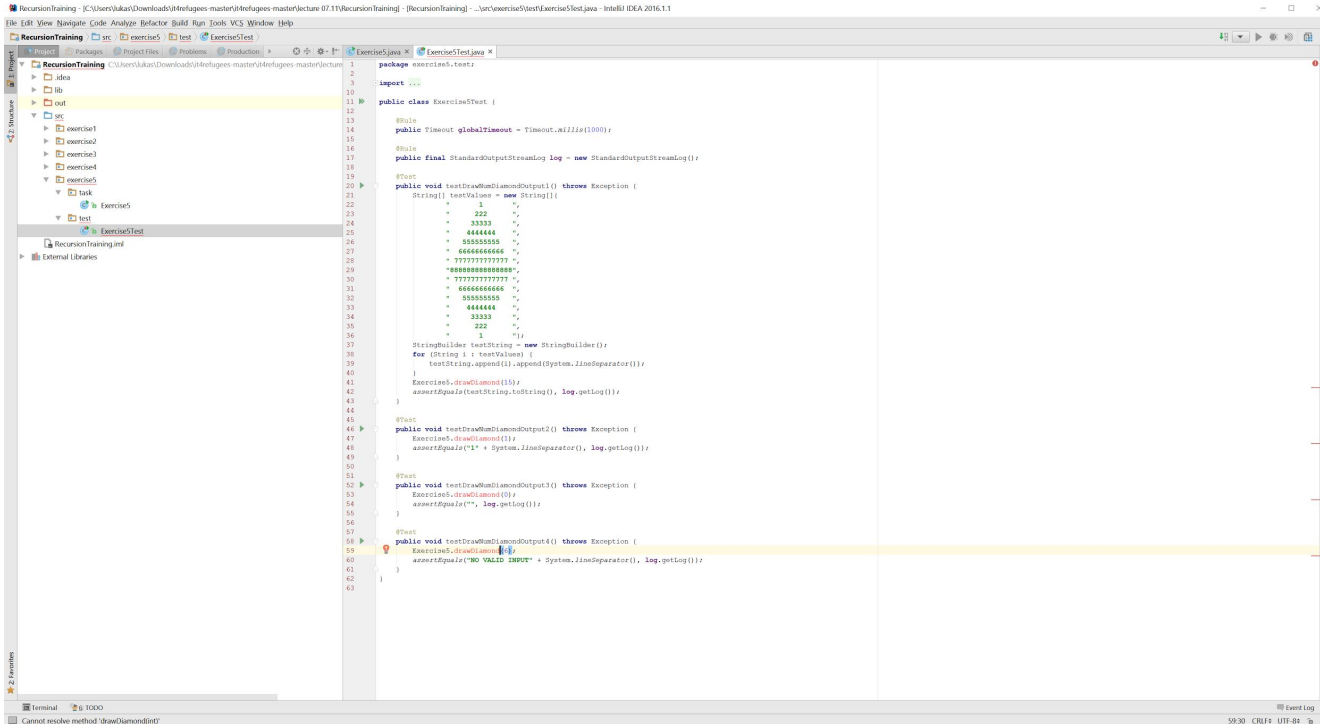
# Homework - only Exercise5



```
1 package exercises.task5;
2
3 /**
4  * Exercise5
5  *
6  * Implement a function called 'drawDiamond' taking a parameter int called h,
7  * returning void ('void drawDiamond(int h)').
8  * This function should print a diamond like this, where h is the 'height' of the diamond, so for e.g. h=5:
9  *
10  *      1
11  *     11
12  *    111
13  *   1111
14  *  11111
15  *
16  * You are going to need odd numbers to do this, so just print 'No valid input!' if an even number or a number smaller
17  * than 0 is given. In case of 0 just leave the function right away without printing anything.
18  * Try to use a maximum 3 for-loops for this exercise (you do not have to use recursion in this one) and it should
19  * be possible to call the function with any h >= 17
20  *
21  * please make sure to call the method exactly as given ('drawDiamond(int h)') otherwise the tests will fail
22  */
23
24 public class Exercise5 {
25     public static void main(String[] args) {
26         //run this method to test your function
27     }
28 }
29
30 }
```

don't forget  
putting  
whitespace right  
of the numbers!  
otherwise the  
test will fail ;)

# How To Run Tests



The screenshot shows an IDE window with a project named 'RecursionTraining'. The left sidebar shows a file tree with 'test' and 'Exercise5Test' selected. The main editor displays the code for 'Exercise5Test.java'. The code includes a package declaration, imports, a class definition, and several test methods. The last test method, 'testDrawDiamond4', is highlighted in yellow and contains a red error icon. The error message at the bottom of the IDE states: 'Cannot resolve method 'drawDiamond4()''.

```
1 package exercises.test;
2
3 import java.util.*;
4
5 public class Exercise5Test {
6
7     @Before
8     public Timeout globalTimeout = Timeout.seconds(1000);
9
10    @Before
11    public final StandardOutputStream log = new StandardOutputStream();
12
13    @Test
14    public void testDrawDiamondOutput1() throws Exception {
15        String[] testValues = new String[] {
16            "1",
17            "222",
18            "3333",
19            "44444",
20            "5555555",
21            "666666666",
22            "7777777777",
23            "8888888888888",
24            "9999999999999",
25            "6666666666",
26            "55555555",
27            "444444",
28            "3333",
29            "22",
30            "1"
31        };
32        StringBuilder testString = new StringBuilder();
33        for (String s : testValues) {
34            testString.append(s).append(System.lineSeparator());
35        }
36        Exercises.drawDiamond(15);
37        assertEquals(testString, log.getLog());
38    }
39
40    @Test
41    public void testDrawDiamondOutput2() throws Exception {
42        Exercises.drawDiamond(1);
43        assertEquals("1", System.lineSeparator(), log.getLog());
44    }
45
46    @Test
47    public void testDrawDiamondOutput3() throws Exception {
48        Exercises.drawDiamond(3);
49        assertEquals("1", log.getLog());
50    }
51
52    @Test
53    public void testDrawDiamondOutput4() throws Exception {
54        Exercises.drawDiamond(4);
55        assertEquals("NO VALID INPUT", System.lineSeparator(), log.getLog());
56    }
57
58    }
59
60
61
62
63
```

Select  
Exercise5Test

→ will show error  
until you  
implement  
method  
'drawDiamond' in  
the correct way!

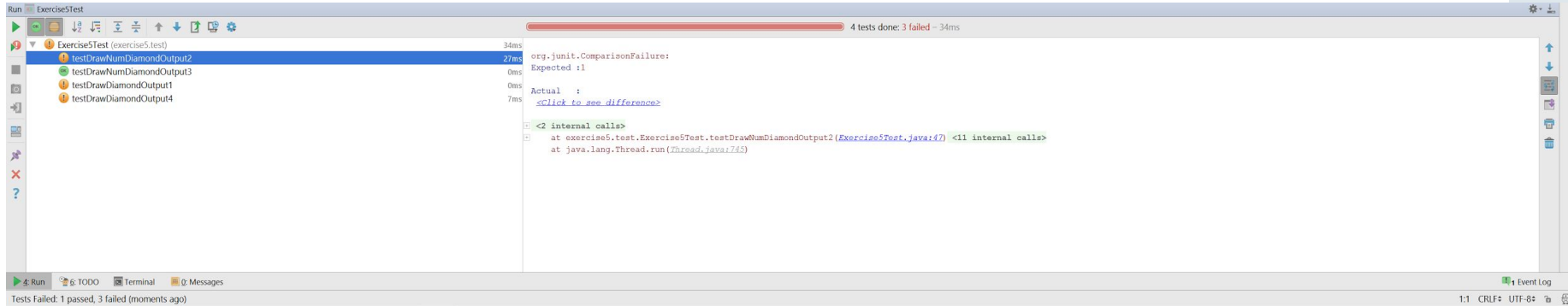
# How To Run Tests

When implemented, select Exercise5Test and select "Run Exercise5Test()'

[illegible]

# How To Run Tests

If the orange Circles turn green you finished the exercise and you have solved the exercise!



# How To Run Tests

If you click <click to see difference> a new window opens, showing in grey what has been expected, and what was your output! You can use it to check why your test failed!

