Task 1

a) match Java format strings

The regex is located in `java.util.Formatter` source code https://github.com/openjdk/jdk/blob/master/src/java.base/share/classes/java/util/Formatter.java. The variable is called `FORMAT_SPECIFIER`.

```
"%(\\d+\\$)?([-#+ 0,(\\<]*)?(\\d+)?(\\.\\d+)?([tT])?([a-zA-Z%])"
```

To get the output, I will collect all match begin and end positions in a Queue data structure. Then pass the queue and the entire text into a function. Each iteration will get the head of the queue. The first `if` is for the case when there is text after the last match left. The second `if` exists so that it will not add TEXT when FORMAT is the first part of the string or when two of them are next to each other.

```
public static void print(Queue<Format> lst, String text) {
  var strBuilder = new StringBuilder();
  var index = 0;
  while (index < text.length()) {
    var format = lst.poll();
    if (format == null) {
      strBuilder.append(String.format("TEXT(%s)",
          text.substring(index, text.length())));
      break;
    }
    if (format.begin != 0 && format.begin != index) {
      strBuilder.append(String.format(
          "TEXT(%s)", text.substring(index, format.begin)));
    }
    strBuilder.append(String.format("FORMAT(%s)",
        text.substring(format.begin, format.end)));
    index = format.end;
  }
  System.out.println(strBuilder);
}
```

b) writing ANTLR4 lexer rules for 12-hour clock

From reading the Wikipedia entry https://en.wikipedia.org/wiki/12-hour_clock. I came up with following lexer rules:

```
lexer grammar Aufgabe2Lexer;

Clock: WORD
     | TIME
     ;

fragment WORD: 'Midnight'
             | 'Noon'
        | '12' WS 'midnight'
        | '12' WS 'noon'
        ;

fragment TIME: HOUR SEPARATOR MINUTE WS UNIT;

fragment UNIT: 'a.m.'
               | 'p.m.'
               ;
```

```
fragment SEPARATOR: ':';

fragment HOUR: [1-9]
             | '1'[0-2]
       ;

fragment MINUTE: [0-9]
               | [0-5][0-9]
             ;

WS: [ \t\r\n]+ -> channel(HIDDEN);
```

Having the rules as fragment hides them in the output.

I also made a tree-sitter grammar just for fun:

```
module.exports = grammar({
  name: "Clock",
  extras: ($) => [/\s|\\\r?\n/],
  rules: {
    clock: ($) => choice($._word, $._time),
    _word: (_) =>
      choice("Midnight", "Noon", seq("12", choice("midnight", "noon"))),
    _time: ($) => seq($._hour, ":", $._minute, " ", $._unit),
    _hour: (_) => choice(/[1-9]/, /1[0-2]/, "12"), // need literal 12 otherwise it
will go to _word.
    _minute: (_) => choice(/[0-9]/, /[0-5][0-9]/),
    _unit: (_) => choice("a.m.", "p.m."),
  },
});
```

Having the rules prefixed with a _ also hides them in the output. The only difference between these two grammars is that the tree-sitter one does not care about whitespace ex.: `12noon` works with it.