Programmi disainist
Programmeerimiskeeled 2017
Vesal Vojdani
How To Design Programs

- “From Problem Analysis to Data Definitions
  Identify the information that must be represented and how it is represented in the chosen programming language. Formulate data definitions and illustrate them with examples.

- Signature, Purpose Statement, Header
  State what kind of data the desired function consumes and produces. Formulate a concise answer to the question what the function computes. Define a stub that lives up to the signature.

- Functional Examples
  Work through examples that illustrate the function’s purpose.”
Evil Hangman Setup

• Prompt the user for a word length, reprompting as necessary until she enters a number such that there's at least one word that's exactly that long. That is, if the user wants to play with words of length -42 or 137, since no English words are that long, you should reprompt her.

• Construct a list of all words in the English language whose length matches the input length.
• Print out how many guesses the user has remaining, along with any letters the player has guessed and the current blanked-out version of the word.
Game State

• Print out how many **guesses** the user has remaining, along with any **letters** the player has guessed and the current blanked-out version of **the word**.
Blanked Out?

- Oluline: alustame lihstalt ühe sõnaga!

- Oletame, et meil on sõna “kala” ja pakutud on Set('l', 'a')

- Kirjutame testi

  blankOutOthers("kala", Set('k', 'l')) == "k_l_"

- Ja implementatsioon…

- Proovime väljastada seisu!
Seisundi värskendamine

- Prompt the user for a single letter guess.
- Partition the words in the dictionary into groups by word family.
- Find the most common “word family” in the remaining words.
<table>
<thead>
<tr>
<th>Kandidaatsönad</th>
<th>Pakutud täht</th>
<th>Eristamatud rühmad</th>
</tr>
</thead>
<tbody>
<tr>
<td>{kala, maja, karu}</td>
<td>a</td>
<td>_ a _ a \rightarrow {kala, maja} _ a _ _ \rightarrow {karu}</td>
</tr>
<tr>
<td>{kala, maja, karu}</td>
<td>k</td>
<td>k _ _ _ \rightarrow {kala, karu} _ _ _ _ \rightarrow {maja}</td>
</tr>
</tbody>
</table>

Rühmitamine
Rühmitamine
Kuidas neid andmestruktuuridena esitada?

<table>
<thead>
<tr>
<th>Sõnad</th>
<th>Täht</th>
<th>Eristamatud rühmad</th>
</tr>
</thead>
<tbody>
<tr>
<td>{kala, maja, karu}</td>
<td>a</td>
<td>_ a _ a → {kala, maja}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>_ a _ _ → {karu}</td>
</tr>
<tr>
<td>{kala, maja, karu}</td>
<td>k</td>
<td>k _ _ _ → {kala, karu}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>_ _ _ _ → {maja}</td>
</tr>
</tbody>
</table>

Map[String, Set[String]]

Set[String]

Char
TDD järjekord

• Mõtleme endale näited läbi… (eelmine slaid)

• Mis on funktsiooni signatuur (kompileeruv stub ehk “???”)

```python
def findGroups(words: Set[String], c: Char): Map[String, Set[String]] = ???
```

• Kirjutame näited unit-testidena:

```python
assert(findGroups('a', Set("kala", "maja", "karu")) ==
    Map("_a_a" -> Set("kala", "maja"), "_a__" -> Set("karu")))
```

• Implementeeri (see on juba kõige lihtsam osa...)
Ja lõpuks: käigu tulemus

• If the word family doesn't contain any copies of the letter, subtract a remaining guess from the user.

• If the player has run out of guesses, pick a word from the word list and display it as the word that the computer initially “chose.”

• If the player correctly guesses the word, congratulate her.