The Dictionary ADT

EECS 214, Fall 2018
The Dictionary ADT: values and operations

Looks like: \{a: 6, b: 7, c: 8\}

Signature:

```java
interface DICT[K, V]:
    def mem?(self, key: K) -> bool?
    def get(self, key: K) -> V
    def put(self, key: K, value: V): VoidC
    def del(self, key: K): VoidC
    def empty? (self): bool?
```
The Dictionary ADT: laws

\[
\emptyset .\text{empty}() \Rightarrow \top \\
\{k_1:v_1, \ldots\} .\text{empty}() \Rightarrow \bot \\
\{k_1:v_1, \ldots, k_i:v_i, \ldots\} .\text{mem}(k_i) \Rightarrow \top \\
\{k \neq k_i\} \{k_1:v_1, \ldots\} .\text{mem}(k) \Rightarrow \bot \\
\{k_1:v_1, \ldots, k_i:v_i, \ldots\} .\text{get}(k_i) \Rightarrow v_i \\
\{k \neq k_i\} \{k_1:v_1, \ldots\} .\text{get}(k) \Rightarrow \text{error!} \\
\{d = \{k_1:v_1, \ldots, k_i:v_i, \ldots\}\} .\text{put}(k_i, v) \{d = \{k_1:v_1, \ldots, k_i:v, \ldots\}\} \\
\{d = \{k_1:v_1, \ldots\} \land k \neq k_i\} .\text{put}(k, v) \{d = \{k_1:v_1, \ldots, k:v\}\} \\
\{d = \{k_1:v_1, \ldots\}\} .\text{del}(k_i) \{d = \{k_1:v_1, \ldots, k_{i-1}:v_{i-1}, k_{i+1}:v_{i+1}, \ldots\}\} \\
\{d = \{k_1:v_1, \ldots\} \land k \neq k_i\} .\text{del}(k) \{d = \{k_1:v_1, \ldots, \}\} \\
\]
Law breakdown: \textit{mem}?

If the key we are looking for is present, we get true:

\[
\{k_1:v_1, \ldots, k_i:v_i, \ldots \}.\text{mem?}(k_i) \Rightarrow \top
\]

If the key we are looking for is not equal to any of the keys in the dictionary, we get false:

\[
\{k \neq k_i\} \{k_1:v_1, \ldots \}.\text{mem?}(k) \Rightarrow \bot
\]
Law breakdown: *get*

If we try to lookup a key present in the dictionary, we get its associated value:

\[
\{k_1:v_1, \ldots, k_i:v_i, \ldots \}.get(k_i) \Rightarrow v_i
\]

If we try to lookup a key that isn’t among the dictionary’s keys—that’s the precondition \( k \neq k_i \)—then it returns a result that indicates that the key wasn’t found:

\[
\{k \neq k_i\} \quad \{k_1:v_1, \ldots \}.get(k) \Rightarrow error!
\]
Law breakdown: put

If we put a key that’s already present, its associated value gets replaced:

\[
\{d = \{k_1:v_1, \ldots, k_i:v_i, \ldots \}\} \ \text{d.put}(k_i, v) \ \{d = \{k_1:v_1, \ldots, k_i:v, \ldots \}\}\]

If we put a key that’s absent, the new key and value association is added:

\[
\{d = \{k_1:v_1, \ldots \} \land k \neq k_i\} \ \text{d.put}(k, v) \ \{d = \{k_1:v_1, \ldots, k:v\}\}\]
Law breakdown: \textit{del}

If we delete a key that’s present, it gets removed:

\[
d = \{k_1:v_1, \ldots \} \quad d.\text{del}(k_i) \quad d = \{k_1:v_1, \ldots, k_{i-1}:v_{i-1}, k_{i+1}:v_{i+1}, \ldots \}
\]

If we delete a key that’s absent, nothing happens:

\[
d = \{k_1:v_1, \ldots \} \land k \neq k_i \quad d.\text{del}(k) \quad d = \{k_1:v_1, \ldots, \}
\]
Next: a data structure for dictionaries