Kehittää ohjelmointitehtävien ratkaisemisessa tarvittavia metakognitioita!

eli ...

Hyvä kaava sanoo enemmän kuin,
. . . tuhat riviä koodia!
. . . sata riviä tekstiä!
. . . kymmenen diagrammia!

Abstraktit tietotyypit (ADT):

Rakenteensa ja operaatioidensa suhteen täsmällisesti kuvattuja ohjelmistokomponentteja, “sortit” ja operaatiot.

- Esimerkiksi

  pino esim. operaatioilla *Create, Top, Pop, Push.*

  binääripuu esim. operaatioilla *Create, Insert, IsIn.*

  hierarkkinen taulukko esim. operaatioilla *Create, IncrLevel, DecrLevel, Add, IsIn, Retrieve.*

- Abstraktius = geneerisyys = alkioiden tyyppiä tai toteutustapaa (kieltä) ei kiinnitetä

- ADT:n *algebrallinen* spesifikaatio:

  Operaatioiden muoto (syntaksi) *eksplisiittisesti*,
  tulkinta (semantiikka) *implisiittisesti* (operaatioiden yhteisvaikutus).

  type *IntStack*;
  functions

$Create$:	$\rightarrow IntStack$
$Push$: $IntStack \times Int$	$\rightarrow IntStack$
Pop : $IntStack$	$\rightarrow IntStack$
Top : $IntStack$	$\rightarrow Int$
$IsEmpty$: $IntStack$	$\rightarrow Boolean$

  axioms

  $IsEmpty(Create) = True$
  $IsEmpty(Push(s, i)) = False$
  $Pop(Create) = Create$
  $Pop(Push(s, i)) = s$
  $Top(Create) = 0$
  $Top(Push(s, i)) = i$

  end *IntStack*;

Abstraktit tietotyypit (ADT):

- Modulaarista rakennetta voidaan tukea esim. seuraavasti:

  type Stack[Item];
  imports True, False **from** Boolean, = **from** Item;
  exports Create, Push, Pop, Top, IsEmpty;

- ADT:n rakenne: *signature* (allekirjoitus) ja *equation* (identiteetit)

  1 **type** *Nat*;
  2 **functions**
  3 $Null :$ $\rightarrow Nat$
  4 $Succ : Nat$ $\rightarrow Nat$
  5 $Add : Nat \times Nat$ $\rightarrow Nat$
  6 **axioms**
  7 $Add(i, Null) = i$
  8 $Add(i, Succ(j)) = Succ(Add(i, j))$
  9 **end** *Nat*;

- *suljettuja* (closed) lausekkeita:

  $Null$

  $Succ(Succ(Add(Succ(Null), Succ(Null))))$

  $Add(Succ(Null), Succ(Succ(Null)))$

- *avoimia* lausekkeita:

  i

  $Add(Succ(i), j)$

Algebrallinen ADT:

- $Nat-$sortti:

```
1  type Nat;
2  functions
3        Null :                    → Nat
4        Succ : Nat                → Nat
5        Add : Nat × Nat           → Nat
6  axioms
7        Add(i, Null) = i
8        Add(i, Succ(j)) = Succ(Add(i, j))
9  end Nat;
```

- Päättelyketju aksioomien avulla:

$$Add(Succ(Null), Succ(Succ(Null)))$$
$$\Downarrow$$
$$\ldots$$
$$\Downarrow$$
$$Succ(Succ(Succ(Null)))$$

- Tulkinnan (semantiikan) liittäminen ADT:hen:

  1. alkuperäinen semantiikka: $Nat = \mathbb{N} = \{0, 1, 2, \ldots\}$ ja

  $$Null_{\mathbb{N}} = 0, \quad Succ_{\mathbb{N}}(n) = n + 1, \quad Add_{\mathbb{N}}(n, m) = n + m.$$

  Tällöin

  $$n + 0 = 0, \quad \text{ja} \quad n + (m + 1) = (n + m) + 1,$$

  $\Rightarrow$ “luonnollinen” algebra!

  2. lopullinen semantiikka: $T = \{0\}$ ja

  $$Null_T = 0, \quad Succ_T(0) = 0, \quad Add_T(0, 0) = 0,$$

  $\Rightarrow$ triviaali algebra!

Algebrallisen ADT:n funktioluokat:

- *IntStack*−sortin allekirjoitus:

  type *IntStack*;
  functions

Create :	$\rightarrow$ *IntStack*
Push : *IntStack* $\times$ *Int*	$\rightarrow$ *IntStack*
Pop : *IntStack*	$\rightarrow$ *IntStack*
Top : *IntStack*	$\rightarrow$ *Int*
Size : *IntStack*	$\rightarrow$ *Int*
IsEmpty : *IntStack*	$\rightarrow$ *Boolean*

- Funktioiden jako luokkiin:

  1. Konstruktorit (*constructors*): *arvojoukkona* määriteltävänä oleva tyyppi.

  1.1 Peruskonstruktorien avulla *muodostetaan* kaikki ADT:n määrittämät tietorakenteet.

  1.2 Ylimääräisten konstruktorien avulla muutetaan rakennetta.

  2. Havainnoijat (*observers*): arvojoukkona muu kuin määriteltävä tyyppi.

  2.2 Ylimääräiset havainnoijat voidaan ilmaista muiden havainnoijien avulla ($\neg\ (\neg\ P) = P$).

  2.1 Perushavainnoijia ei voida ilmaista muiden havainnoijien avulla.

Kuinka muodostaa algebrallisen ADT:n aksioomat?

type *IntStack*;
functions

Create :	$\rightarrow$ *IntStack*
Push : *IntStack* $\times$ *Int*	$\rightarrow$ *IntStack*
Pop : *IntStack*	$\rightarrow$ *IntStack*
Top : *IntStack*	$\rightarrow$ *Int*
Size : *IntStack*	$\rightarrow$ *Int*
IsEmpty : *IntStack*	$\rightarrow$ *Boolean*

1. Määritä peruskonstruktorit
2. Määritä ylimääräiset konstruktorit ja liitä niistä jokainen *kaikkiin* peruskonstruktoreihin sopivan aksiooman avulla:

   $Pop(Create) =$ "Mikä pino saadaan poistettaessa tyhjän pinon päällimmäinen alkio?"

   $Pop(Push(s,i)) =$ "Mitä tapahtuu, kun pinoon lisätään ja poistetaan alkio?"
3. Jokainen perushavainnoija liitetään aksiomaattisesti peruskonstruktoreihin:

   $Top(Create) =$ "Mikä alkio saadaan nyhjäistyä tyhjän pinon päältä?"

   $Top(Push(s,i)) =$ "Mikä alkio löytyy pinon päältä, kun sinne on lisätty alkio *i*?"

   $Size(Create) =$ "Minkä kokoinen on tyhjä pino?"

   $Size(Push(s,i)) =$ "Mitä tapahtuu pinon koolle, kun sinne lisätään alkio *i*?"
4. Lopuksi jokainen ylimääräinen havainnoija liitetään sopivan aksiooman avulla perushavainnoijiin:

   $IsEmpty(s) =$ "Mikä ehto pinon koon suhteen kuvaa tyhjää pinoa?"