Formal Definition

An *abstract data type* $\mathcal{D}$ is defined as a 5-tupel:
$\mathcal{D} = (N, P, Fs, Ts, Ax)$,
where its components are the following:

1. N is a string. This string is the *name* of the ADT.

2. P is the set of *type parameters.* Here, a type parameter is usually just a string, which denotes a type variable.

3. Fs is the set of *function symbols.* These function symbols denote the operations that are supported by this ADT. The function symbols itself are strings.

4. Ts is a set of *type specifications.* For every function symbol $f \in Fs$ the set Ts contains a type specification of the form
   $f : T_1 \times ... \times T_n \to S$,
   where $T_1, ..., T_n, S$ are names of data types. There are three types of these data types:

   (a) Predefined data types like *int* or *str*

   (b) Names of ADTs

   (c) Type parameters from the set P

   This type specification expresses the fact, that the function f has to be called as $f(t_1, ..., t_n)$, where the argument t_i is of type $T_i : \forall i \in \{1, ..., n\}$.
   Further, the result of the function f has to be of type S.
   Additionally, we must have either $T_1 = N \vee S = N$, where N is the name of this ADT. If we have $T_1 \neq N$, then f is called a *constructor* of $\mathcal{D}$.
   Otherwise, f is called a *method.*
   Iff f is a method, we usually write $N.f(...)$ to denote $f(N, ...)$

5. Ax is a set of *axioms* of $\mathcal{D}$. They are mathematical formulas, that specify the behavior of the ADT.