Disjoint Unions

- Pascal: variant record
- C/C++: union
- ML: union type
- Ada: discriminated record
What is a disjoint union?

• “Another kind of composite value is the disjoint union, whereby a value is chosen from one of several (usually different) sets.”

  David Watt

  *Programming Language Design Concepts* page 27
S+T

Values are tagged to indicate which set they are from:

\[ S+T = \{ \text{left}(x) \mid x \in S \} \cup \{ \text{right}(y) \mid y \in T \} \]

Cardinality? \( #(S+T) = #S + #T \)
datatype number = Exact of int | Inexact of real;

datatype person = King
    | Peer of string*string*int
    | Knight of string
    | Peasant of string;

Values: King, Peer("Earl", "Carlisle", 7),
        Knight("Gawain"), Peasant("Jack Cade")
The main purpose of a disjoint union is to bring together values from different sets in a way that you can determine from which set a given value is drawn.

This type of structure is used very commonly in OO programming.

Polymorphism feeds off of this idea: implicit selection based on type.
OO disjoint union example

- **Employee**
  - Employee(Position initialPosition)
  - int getSalary()

- **Position**
  - int getSalary()

- **SalesPerson**
  - int getSalary()
  - int getHoursWorked()
  - int getHourlyRate()

- **RegionalManager**
  - int getSalary()
  - int getBaseSalary()
  - int getProfitSharingBonus()

- **Executive**
  - int getSalary()
  - int getBaseSalary()
  - int getProfitSharingBonus()
  - int getExecutiveBonus()