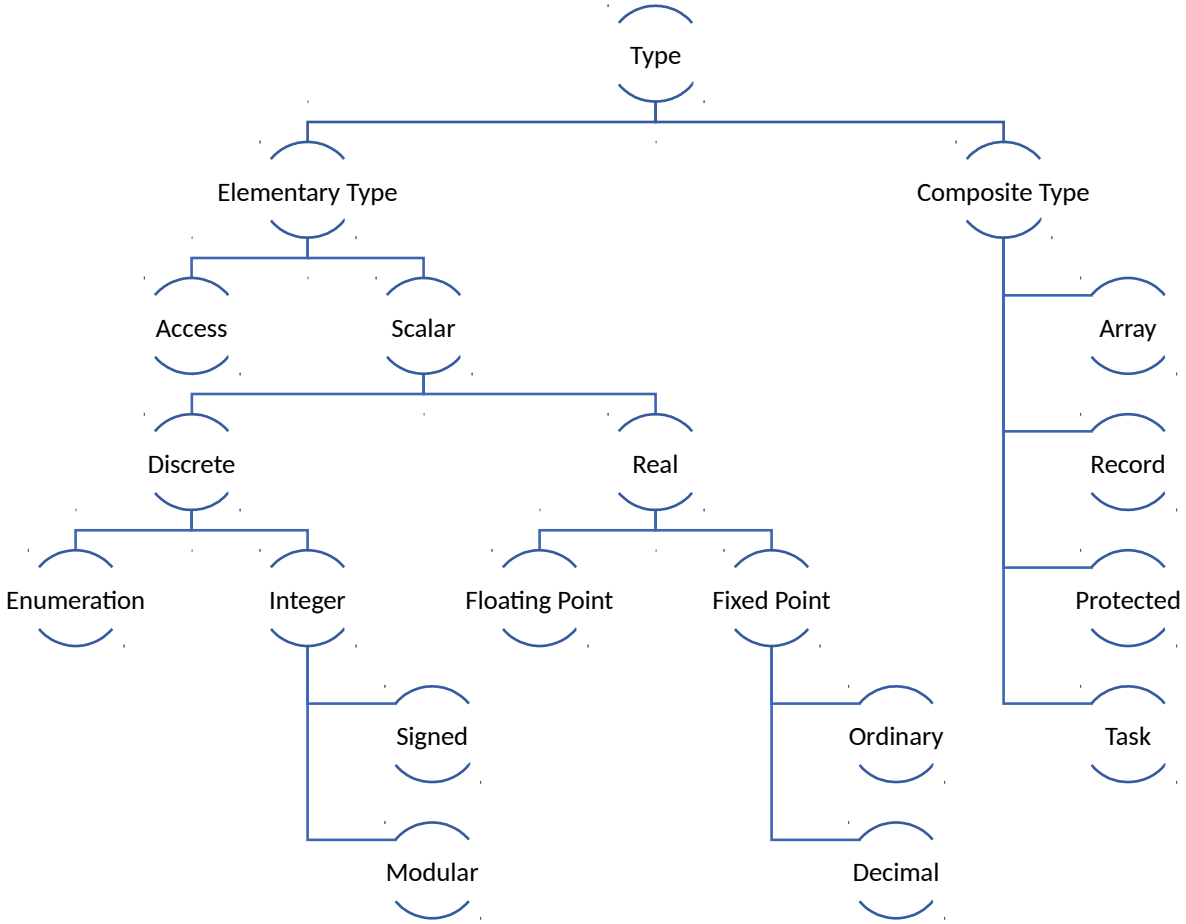
SWEN421 - Lecture 2
Data Types in Ada

Ada Types

- Ada has a very rich, strong type system
- Includes generics and subtypes
- No implicit type conversions (coercions) – check that!
- Can declare anything as a new type (cf. just objects in Java)

Ada type hierarchy



Defining types, type synonyms

- Can define a type name as a synonym for an existing type:
  - type apple is Integer;
- Integer types are defined by a range of values:
  - type percentage is range 0 .. 100;
- Enumeration type is defined by a list of values (names):
  - type Direction is (North, East, South, West);
- Float types are defined by number of significant digits, and range:
  - type Distance is digits 10;
  - type Temperature is digits 5 range -273.15 .. 1_000_000.0;

Defining new types from old, and subtypes

- type ExamScore is new Percentage;
- type TestScore is new Percentage range 0 .. 50;

- subtype Natural is Integer range 0 .. Integer'Last;
- subtype Positive is Integer range 1 .. Integer'Last;

- Can mix operations on subtypes, but not on new types.

- Which is these can you do arithmetic operations on?

Arrays

- Array index can be any discrete type – usually integer range or enumeration:
  - type A is array (Integer range 1 .. 10) of Integer;
  - type B is array (Character range ’a’ .. ‘z’) of Character;
  - type C is array (Direction) of Float;
  - type D is array (Integer range lo .. hi) of Integer;
- Bounds can be left to be supplied later (indefinite):
  - type E is array (Integer range <>) of Integer;
  - x : E(-100 .. 100);

Arrays

- Array operations act on the whole array!
  - A := B; -- copies the whole array (not a pointer)
  - if A = B then … -- compares the two arrays' elements (not their start address)
  - A & B concatenates two arrays
  - Can use array "slices": A := B(5 .. 10);

- Does this hold for other operations, like < and > ?

Records

- A record is a structure with named components of (possibly) different types (cf. struct in C, object with no methods in Java):
  - type Shape is record
    id: Integer;
    x, y: Float;
    end record;

- Elements accessed using dot notation:
  - s: Shape;
    begin
    s.id := 17;

Variant records

- A record type can have different forms, depending on a “discriminant” field (a kind of type union):
  - type Shape (kind: ShapeKind) is record
    x, y: Float;
    case kind is
      when Line =>
        u, v: Float;
      when Circle =>
        radius: Float;
      …
    end record;
  - s := Shape(circle);
    s.radius := 15.5; -- can only access fields for this variant!