Programming Languages

G22.2110
A brief history of Ada

- 1975-1979: strawman-steelman requirements, Pascal is base
  - Packages, generics, tasks, derived types
- 1983 : ANSI Standard for Ada83, first validated compiler
- 1990-1994 : Revision Process
  - Objects, dynamic dispatching, polymorphism, protected types
- 1995 : First validated GNAT compiler
- 1995-2004: Revision Process
- 2005 (Expected) new ISO Ada05 Standard
  - Interfaces for multiple inheritance, mutually-recursive types
Design principles

- software engineering: programming in the large
- readability over writability
- rigorous standard for portability
- precise semantics
- built-in concurrency
- strong typing: all unsafe code is explicit
Ada, C++ and Java

- big languages win
- strong typing: Ada, C++, Java
- generics: Ada $\Rightarrow$ C++ $\Rightarrow$ Java 1.5
- concurrency: Ada $\Rightarrow$ Java
- packages: Ada $\Rightarrow$ C++ and Java
  - ahead of its time in design
  - out of step in implementation
  - between language generations
Key Goals

- readability: between COBOL and APL
- static typing: find as many errors as possible during compilation
- programming in the large: package it and offer interfaces
- exception handling: hope for the best; be prepared for the worst
- data abstraction: can’t compare apples and oranges
- object orientation: inheritance and late binding
- tasking: threads (walk and chew gum at the same time)
- generic units: similar functionality across types (parametric polymorphism, Ada-style)
The canonical example

with Text_Io; use Text_Io;

procedure example is
begin
  Put_Line ("Hello, world.");
end;
package Math_Functions is
    function Sqr (X: Float) return Float;
    function Exp (Base: Float;
                    Exponent: Float) return Float;
end Math_Functions;
Using the package

```ada
with Math_Functions;
with Ada.Text_IO;  use Ada.Text_IO;

procedure Example2 is
   Val: Float;
begin
   Get (Val);
   Put ("Sqrt("); Put (Val); Put ("\)=\)");  
   Put (Math_Functions.Sqrt (Val));
   New_Line;
end;
```
Implementing the package

package body Math_Functions is
    Epsilon: constant := 1.0e-7;
    function Sqrt (X: Float) return Float is
        Result: Float := X / 2.0;
        begin
            while Abs(Result*Result - X) > Epsilon loop
                Result := 0.5 * (X / Result + Result);
            end loop;
            return Result;
        end Sqrt;
    ...
Summary of Package Features

- package contains data and functions
- can be accessed by writing “with” (and avoid qualification by writing “use”)
- full program is normally a bunch of packages and a “main” procedure that starts the program.
- Define interface by giving signatures (aka specification) and then define body separately
- functions have in parameters and a return value
- procedures can have parameters of any mode (in, out, or in out) but no return value