Section 02
Packages 01

Pedro Fernando Flores Palmeros

1 INTRODUCTION

In the previous section simple standalone subprograms were presented. It is easy to see that the programming style from the first section is not useful for scaling up for real-world applications. A better way to structure the code into modular and distinct units is needed.

Ada encourages the separation of programs into multiple packages and sub-packages, providing many tools to a programmer on a quest for a perfectly organized code-base.
Here is an example of a package in ADA

Code 1: week.ads

```
1 package Week is
2    Mon : constant String := "Monday";
3    Tue : constant String := "Tuesday";
4    Wed : constant String := "Wednesday";
5    Thu : constant String := "Thursday";
6    Fri : constant String := "Friday";
7    Sat : constant String := "Saturday";
8    Sun : constant String := "Sunday";
9 end Week;
```

The `Code 1` is package, the first aspect to have on mind is that the file extension has changed to `.ads` which is a `spect` ADA file. This do not do anything by itself, it has to be invoked from a `main` or other program.

Code 2: main.adb

```
1 with Ada.Text_IO; use Ada.Text_IO;
2 with Week;
3 -- References the Week package,
4 -- and ads a dependency from Main to Week
5
6 procedure Main is
7 begin
8    Put_Line("First day of the week is " & Week.Mon);
9 end Main;
```

The Code 2 snippet is the main code, in this code at line 8 is being invoked the week package and specifically the Monday day.

Packages let make your code modular, separating you programs into semantically significant unit. Additionally the separation of a package's specification from its body can reduce compilation time.

While the `with` clause indicates a dependency, you can see in the example that you still need to prefix the referencing of entities from the Week package by the name of the package. If we had included a `use` Week clause, then such a prefix would not have been necessary.

Accessing entities from a package uses the dot notation, A.B which is the same notation as the one used to access record files.

A `with` clause can *only* appear in the prelude of a compilation unit (before the reserved word such as `procedure`, that makrs the beginning of the unit). It is not allowed anywhere else. This rule is only needed for methodological reasons: the person reading your code should be able to see immediately which units the code depends on.

Code 3: main.adb

```
1 with Ada.Text_IO; use Ada.Text_IO;
2 with Week; use Week;
3 -- References the Week package,
4 -- and ads a dependency from Main to Week
5
6 procedure Main is
7 begin
8    Put_Line("First day of the week is " & Mon);
9 end Main;
```