SWEN90010: HIGH INTEGRITY SYSTEMS ENGINEERING
DEPARTMENT OF COMPUTING AND INFORMATION SYSTEMS
THE UNIVERSITY OF MELBOURNE

Workshop Ada #2

Introduction

The aim of this workshop is to extend your understanding of Ada and especially:

- Ada packages; and
- Ada records.

The intention is to work on a couple of small programs with records and how they can be used with packages. In doing so, you should get familiar with the way that Ada handles certain constructs. The tasks for the workshop are short. The remainder of the workshop can be used to start on the assignment.

Ada packages

Recall that a package in Ada is a module. A package consists of two parts:

1. A *specification* (the `.ads` file), which specifies publicly available constructs, including type definitions, data definitions, and sub-programs.
2. A *body* (the `.adb` file), which implements the specification. This is the encapsulated part of the package.

Ada specifications can declare types without defining them. That is, they declare type names, but not what the types are. For example:

```
type DateType is private;.
```

The details of the type are then defined in the *private* part of the package specification. Any calling program that uses a private type cannot see the internals of the type. It is up to the designer of the package to provide the necessary subprograms (getters and setters) to modify and read data from the private type. This is one way that encapsulation can be achieved with Ada.

Record types

Recall from the notes that in Ada, *records* can be used to group related information. The supporting source code for this workshop (available on the LMS) contains two packages that use record types: `Date` and `Time`.

Note that the types `DateType` and `TimeType` are both declared as `private`, which means that other packages cannot access the attributes of the records directly, and that getter and setter methods are used to read and modify the attributes respectively. The `TimeType` itself contains a nested `DateType` record.

The packages `TestDate` and `TestTime` provide examples of how to create variables of type `DateType` and `TimeType` respectively, and how to call subprograms for changing and reading the data in these types. This is analogous to creating instances of objects and calling methods, except that the subprograms are not bound to the record instances: they are statically bound.

Your Tasks

1. Compile and run the procedures in `TestDate` and `TestTime`. Note that the `TestTime` procedure should fail: the code for the `Time` package is incorrect.

2. This task is straightforward *if* one understands how to call sub-programs.

   The test in `TestTime` fails because the `Increment` procedure in the `Time` package fails to increment the date at the end of the day.

   Your task is to correct the `Time.Increment` procedure by incrementing the date correctly. To do this, you will need to understand how to call subprograms on records.

3. Once you have finished tasks 1 and 2, start working on the assignment — or continue working on the assignment if you have already started.