Ada.O5: Hantering av poster och enkla paket

I denna uppgift kommer du att skapa ett paket med rutiner som kan hantera ett datum.

Mål

Du skall efter denna uppgift kunna skapa egna paket med datatyper och rutiner som senare kan användas i ett större program. Undantag skall kunna hanteras både i och utanför paketet.

Uppgift

Du skall skapa ett paket för att hantera ett datum. Uppgiften är uppdelad i delar för att du ska undvika alltför många saker på en gång. Man bör alltid försöka dela upp stora problem i mindre.

På filen `test_date.adb` finns ett huvudprogram som använder delarna ifrån ditt paket. Det enda du ska behöva göra är att lägga in inkluderingen av ditt paket i detta huvudprogram för att det ska fungera enligt körexemplen nedan (när du implementerat alla delar). Ett tips är att kommentera ut de delar du inte implementerat ännu så att du kan testa ditt paket allt eftersom du arbetar.

När du skickar in denna uppgift för rättning ska du skicka med `.adb`- och `.ads`-filen för ditt paket samt denna modifierade `test_date.adb` så vi kan testköra ditt program.

KRAV (detta gäller samtliga delar): Alla huvud- och underprogram får ha som mest ca 20 rader kod mellan "begin" och "end". Tänk över uppdelningen av underprogram, vissa underprogram kanske kan anropas från flera ställen i koden.

Del A:

I Ada.O4 skapade du datatypen `Date_Type` samt underprogrammen för denna datatyp. Dessa ska du lägga in i ett paket där datatypen blir privat och de relevanta underprogrammen blir publika.

Tips! Skapa en ny mapp för Ada.O5 och kopiera in din Ada.O4-lösning till denna mapp. Använd sedan kopian när du arbetar på denna del. På så sätt har du kvar Ada.O4-lösningen utan risk för att du ändrar/tar bort något av originalet.

Del B:

Lägg till de publika funktionerna `Next_Date` och `Previous_Date` i paktetet. Dessa underprogram tar en `Date_Type`-parameter som indata (t.ex. 1997-01-01). `Next_Date` ska returnerar nästa datum (t.ex. 1997-01-02) som en `Date_Type`. `Previous_Date` ska returnera föregående datum (t.ex. 1996-12-31) som en `Date_Type`.

Funktionerna ska ta hänsyn till skottår. Generellt sett är det skottår var fjärde år. Dock finns det några undantag. Det är inte skottår då årtalet är jämnt delbart med 100. Dock är det skottår om årtalet är jämnt delbart med 400. Om det är skottår har februari 29 dagar istället för 28.

Del C:

Lägg till de publika operatorerna ”<”, ”=” och ”>” i paketet. Dessa operatorer tar två `Date_Type`-parametrar som indata.

Operatorerna ska returnera `True` eller `False` beroende på huruvida jämförelsen logiskt sett stämmer eller ej. Självklart ska duplicering av kod undvikas.

Du skapar nu operatoröverlagringar som kan användas på samma sätt som motsvarande operatorer för heltal eller flyttal, men i detta fall gäller de för värden av typen `Date_Type`.

Körexempel 1:

```
Mata in datum 1: 2020-01-01
Datumet var: 2020-01-01
Nästa dag: 2020-01-02
Föregående dag: 2019-12-31
Mata in Datum 2: 2019-12-31
Datum 1 > Datum 2? Ja.
Datum 1 < Datum 2? Nej.
Datum 1 = Datum 2? Nej.
```

Körexempel 2:

```
Mata in datum 1: 2000-04-40
Felaktig dag.
```

Körexempel 3:

```
Mata in datum 1: 2000-02-28
Datumet var: 2000-02-28
Nästa dag: 2000-02-29
Föregående dag: 2000-02-27
Mata in Datum 2: 2010-01-01
Datum 1 > Datum 2? Nej.
Datum 1 < Datum 2? Ja.
Datum 1 = Datum 2? Nej.
```

Körexempel 4:

```
Mata in datum 1: 2000-03-01
Datumet var: 2000-03-01
Nästa dag: 2000-03-02
Föregående dag: 2000-02-29
Mata in Datum 2: 2000-03-01
Datum 1 > Datum 2? Nej.
Datum 1 < Datum 2? Nej.
Datum 1 = Datum 2? Ja.
```

Körexempel 5:

```
Mata in datum 1: 12345-3-BB
Felaktigt format.
```

Körexempel 6:

```
Mata in datum 1: 2009:02:13
Felaktigt format.
```

Körexempel 7:

```
Mata in datum 1: 2#1#-01-02
Felaktigt format.
```

Körexempel 8:

```
Mata in datum 1: 9001-12-23
Felaktigt år.
```

Körexempel 9:

```
Mata in datum 1: 1904-13-01
Felaktig månad.
```

Körexempel 10:

```
Mata in datum 1: 1995-02-29
Felaktig dag.
```