

Eiffel

First Example: Functions

```
class
  TIME_OF_DAY
    -- Absolute time within a day
feature
  hour: INTEGER is
    -- Hour of day, 00 to 23
  do
    Result := minutes // 60
  end -- hour

  minute: INTEGER is
    -- Minute in hour, 00 to 59
  do
    Result := minutes \% 60
  end -- minute
```

First Example: Procedures and Assertions

```
set (hh: INTEGER, mm: INTEGER) is
  require
    0 <= hh and hh < 24
    0 <= mm and mm < 60
  do
    minutes := hh * 60 + mm
  ensure
    hour = hh
    minute = mm
  end -- set
```

```
adjust (hh_by: INTEGER, mm_by: INTEGER) is
  -- Advance (+) or retard (-) either or both
  do
    minutes := minutes + hh_by * 60 + mm_by
    normalise
  end -- adjust
```

First Example: Variables and Visibility

```
feature {NONE}
  minutes: INTEGER
    -- Minutes since midnight

  normalise is
    -- Restore invariant after adjustment
  do
    minutes := minutes \% 1440
    if minutes < 0 then
      minutes := minutes + 1440
    end
  end -- normalise
```

First Example: Invariants and Object Creation

```
invariant
```

```
    0 <= hour and hour < 24
```

```
    0 <= minute and minute < 60
```

```
    0 <= minutes and minutes < 1440
```

```
end -- class TIME_OF_DAY
```

```
...
```

```
    finish_time: TIME_OF_DAY;
```

```
    ...
```

```
    create finish_time -- !!finish_time
```

Creators

```
class TIME_OF_DAY

creation
  set

feature
  ...

  lunchtime: TIME_OF_DAY
  ...
  create lunchtime.set (12, 30)
  -- !!lunchtime.set (12, 30)
```

Clone

```
ff_start, ent_start: TIME_OF_DAY
...
create ff_start -- !!ff_start
...
if starting_together then
    ent_start := clone (ff_start)
end
```

Rename

```
class
  HOUR12
      -- Hour component of time of day
inherit
  DIGIT12
      rename
          increment as advance,
          decrement as retard
      end
end -- class HOUR12
```


Export

```
class
  DIGITF

inherit
  DIGIT
    export
      {NUMBER} more_sig
    end

end -- class DIGITF
```

Redefinition

```
class HOUR12 -- Hour component of time of day
inherit
  DIGIT12
  redefine
    symbol
  end
feature
  symbol: STRING is -- 12, 1, 2... 10, 11
  do
    ...
  end -- symbol
end -- class HOUR12
```

Genericity

```
digits: ARRAY [DIGIT]
create digits.make (1, 3) -- !!digits.make(1,3)
```

bounded Genericity:

```
class
  SORTED_LIST [ET -> COMPARABLE]
  ...
```

Input and Output

```
class FILTER
creation run
feature
  run is -- Echo until end of file
  do
    from
      io.read_line
    until
      equal (io.last_string, "")
    loop
      io.put_string (io.last_string)
      io.put_new_line
      io.read_line
    end -- loop
  end -- run
end -- class FILTER
```

Another Example: Putting Things Together

```
class
  MON12
      -- Month of year
inherit
  DIGIT12
      redefine
          symbol, increment, decrement
      end
creation
  make,
  connect_day
```

Another Example: Direct Reuse

```
feature
```

```
  increment is
```

```
    -- As parent, but note change
```

```
  do
```

```
    Precursor
```

```
    changed
```

```
  end -- increment
```

```
  decrement is
```

```
    -- As parent, but note change
```

```
  do
```

```
    Precursor
```

```
    changed
```

```
  end -- decrement
```

Another Example: New Features

```
feature {NONE} -- new features, protected
  day: DIGITV -- Day of month, we control range
  changed is -- Month has changed: update day range
  do
    if day /= Void then
      inspect value + 1
      when 1, 3, 5, 7, 8, 10, 12 then
        day.set_range (1, 31)
      when 2 then
        day.set_range (1, 28) -- leap?
      when 4, 6, 9, 11 then
        day.set_range (1, 30)
      end
    end
  end
end -- changed
```

Another Example: Once

```
full_names: ARRAY [STRING] is
    -- Full month names
    once
        Result := <<"January", "February", "March",
            "April", "May", "June",
            "July", "August", "September",
            "October", "November", "December">>
    end
end -- class MON12
```


References

Simon Parker: Eiffel for Beginners

www.maths.tcd.ie/~odunlain/eiffel/eiffel_course/eforb.htm

(very comprehensive, but syntax outdated)

Bertrand Meyer: Object-Oriented Software Construction, 2nd Edition

Prentice Hall, ISBN 0-13-629155-4, 1997

(software development method in detail, still worth reading)

Web page of ISE Eiffel

docs.eiffel.com/book/method/method

(much information up to date)