
Eiffel: Beispiel 1a

```
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
```

Eiffel: Beispiel 1b

```
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
```

Eiffel: Beispiel 1c

```
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
```

Eiffel: Beispiel 1d

```
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;
```

```
    ...
```

```
    !!finish_time
```

Eiffel: Creation

```
class TIME_OF_DAY

creation
    set

feature
    ...

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

Eiffel: Clone

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

Eiffel: Rename

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

Eiffel: Export

```
class
  DIGITF

inherit
  DIGIT
    export
      {NUMBER} more_sig
    end

end -- class DIGITF
```

Eiffel: Überschreiben

```
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
```

Eiffel: Generizität

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

auch gebundene Generizität:

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

Eiffel: 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
```

Eiffel: Beispiel 2a

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

Eiffel: Beispiel 2b

```
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
```

Eiffel: Beispiel 2c

```
feature {NONE} -- new features, private
  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 -- changed
```

Eiffel: Beispiel 2d

```
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
```