Prova di esame del 22 settembre 2016

Esercizio 1) [10 punti]

Marcare le affermazioni che si ritengono vere. Ogni domanda può avere un qualunque numero naturale di affermazioni vere. Vengono **assegnati** 0.5 punti sia per ogni affermazione *vera che viene marcata* che per ogni affermazione *falsa che viene lasciata non marcata*. Analogamente vengono **sottratti** 0.5 punti sia per ogni affermazione *falsa che viene marcata* che per ogni affermazione *vera che viene lasciata non marcata*.

1. Se la classe *C* esporta la feature *x* verso *NONE* allora …
   a. Solo un'istanza di *C* può invocare in modo qualificato la feature *x* di un'altra istanza di *C*
   b. Solo la stessa istanza di *C* può invocare in modo non qualificato la feature *x* di sé stessa
   c. Nessuna istanza può invocare in modo qualificato la feature *x* di un'istanza di *C*
   d. Una qualunque istanza può invocare in modo qualificato la feature *x* di un'istanza di *C*

2. Sia f una feature *effective* introdotta nella classe *A*, siano *B* e *C* due classi che ereditano direttamente da *A* (cioè non attraverso altre classi) e che non modificano *f* in alcun modo. Sia *X* una classe che eredita direttamente sia da *B* che da *C*. Allora …
   a. … per usare f in X si deve attuare *rename* di almeno una delle due versioni ereditate
   b. … si può usare f in X anche senza modificare quanto ereditato
   c. … per usare f in X si deve attuare *rename* di entrambe le versioni ereditate
   d. … non si può in alcun caso usare f in X

3. Rinominare una feature *f* ereditata implica …
   a. … cambiare sia il nome che l'implementazione della feature
   b. … cambiare l'implementazione della feature senza necessariamente cambiarne il nome
   c. … portare lo stato della feature a *effective*
   d. … cambiare il nome della feature senza necessariamente cambiarne l'implementazione

4. …
   a. Se la clausola *until* del loop è falsa si esce dal loop
   b. L'invariante del loop deve essere vero anche immediatamente prima della prima iterazione
   c. La variante del loop deve sempre essere un intero positivo
   d. Se la clausola *until* del loop è vuota il programma non compila correttamente

5. Sia f una feature *effective* introdotta nella classe *A*, siano *B* e *C* due classi che ereditano direttamente da *A* (cioè non attraverso altre classi). Assumiamo che né *B* ne *C* modifichino *f* in alcun modo. Sia *X* una classe che eredita direttamente sia da *B* che da *C*. Allora per usare *f* in *X* esattamente col nome *f* è necessario che…
   a. …*X* effettui *rename* di almeno una delle due versioni ereditate
   b. …*X* effettui *deferred* di esattamente una delle due versioni ereditate
   c. …*X* effettui *undefine* di almeno una delle due versioni ereditate
   d. … *X* effettui *rename* di esattamente una delle due versioni ereditate

SOLUZIONE:

1. Se la classe *C* esporta la feature *x* verso *NONE* allora …
   a. Solo un'istanza di *C* può invocare in modo qualificato la feature *x* di un'altra istanza di *C*
   b. Solo la stessa istanza di *C* può invocare in modo non qualificato la feature *x* di sé stessa
   c. Nessuna istanza può invocare in modo qualificato la feature *x* di un'istanza di *C*
   d. Una qualunque istanza può invocare in modo qualificato la feature *x* di un'istanza di *C*

2. Sia f una feature *effective* introdotta nella classe *A*, siano *B* e *C* due classi che ereditano direttamente da *A* (cioè non attraverso altre classi) e che non modificano *f* in alcun modo. Sia *X* una classe che eredita direttamente sia da *B* che da *C*. Allora …
   a. … per usare f in X si deve attuare *rename* di entrambe le versioni ereditate
   b. … si può usare f in X anche senza modificare quanto ereditato
   c. … per usare f in X si deve attuare *rename* di almeno una delle due versioni ereditate
   d. … non si può in alcun caso usare f in X

3. Rinominare una feature *f* ereditata implica …
   a. … cambiare sia il nome che l'implementazione della feature
   b. … cambiare l'implementazione della feature senza necessariamente cambiarne il nome
   c. … portare lo stato della feature a *effective*
   d. … cambiare il nome della feature senza necessariamente cambiarne l'implementazione

4. …
   a. Se la clausola *until* del loop è falsa si esce dal loop
   b. L'invariante del loop deve essere vero anche immediatamente prima della prima iterazione
   c. La variante del loop deve sempre essere un intero positivo
   d. Se la clausola *until* del loop è vuota il programma non compila correttamente

5. Sia f una feature *effective* introdotta nella classe *A*, siano *B* e *C* due classi che ereditano direttamente da *A* (cioè non attraverso altre classi). Assumiamo che né *B* ne *C* modifichino *f* in alcun modo. Sia *X* una classe che eredita direttamente sia da *B* che da *C*. Allora per usare *f* in *X* esattamente col nome *f* è necessario che…
   a. …*X* effettui *rename* di almeno una delle due versioni ereditate
   b. …*X* effettui *deferred* di esattamente una delle due versioni ereditate
   c. …*X* effettui *undefine* di almeno una delle due versioni ereditate
   d. …*X* effettui *rename* di esattamente una delle due versioni ereditate

Esercizio 2) [10 punti]

La classe *INT_LINKABLE* modella un elemento di una lista che può rappresentare valori interi. La sua implementazione è la seguente:

```
class
   INT_LINKABLE
create
   make

feature -- accesso
   value : INTEGER
      -- L'intero memorizzato in questo elemento

   next : INT_LINKABLE
      -- Il successivo elemento della lista

feature -- operazioni fondamentali
   make (a_value : INTEGER)
      -- crea l'elemento
      do
         value := a_value
      ensure
         value = a_value
      end

   link_to (other: INT_LINKABLE)
        -- collega questo elemento con `other'
      do
         next := other
      ensure
         next = other
      end

   link_after (other: INT_LINKABLE)
        -- inserisce questo elemento dopo `other' conservando quello che c'era dopo di esso
      require
         other /= Void
      do
         link_to (other.next)
         other.link_to (Current)
      ensure
         other.next = Current
         other.next.next = old other.next
      end

end
```

La classe *INT_LINKED_LIST* modella una lista di interi. Una parte della sua implementazione è fornita qua sotto:

```
class
   INT_LINKED_LIST

feature -- accesso
   first_element: INT_LINKABLE
        -- Il primo elemento della lista

   last_element: INT_LINKABLE
        -- L'ultimo elemento della lista

   count: INTEGER
        -- Il numero di elementi della lista
```

Aggiungere alla classe *INT_LINKED_LIST* una feature *value_precedes* (*a_value, target*: *INTEGER*): *BOOLEAN* che restituisce vero se e solo se la lista contiene un elemento di valore *a_value* **prima** della prima occorrenza di *target* (non necessariamente subito prima).

Indicare anche contratti ed eventuali invarianti. Non è possibile modificare la classe *INT_LINKABLE.*

```
feature -- operazioni fondamentali
   value_precedes (a_value, target: INTEGER): BOOLEAN
        -- La lista contiene `a_value' prima della prima occorrenza di `target'?

      require
         ____________________________________________________________________________
         ____________________________________________________________________________
      local
         ____________________________________________________________________________
         ____________________________________________________________________________
      do
         ____________________________________________________________________________
         ____________________________________________________________________________
         ____________________________________________________________________________
         ____________________________________________________________________________
         ____________________________________________________________________________
         ____________________________________________________________________________
         ____________________________________________________________________________
         ____________________________________________________________________________
         ____________________________________________________________________________
         ____________________________________________________________________________
      ensure
         ____________________________________________________________________________
         ____________________________________________________________________________
end
```

SOLUZIONE:

Una possibile soluzione usa una feature separata *get_element*(*a_value*), che restituisce il puntatore alla prima occorrenza di *a_value*, e lo usa per determinare il risultato della query.

```
get_element (a_value: INTEGER): INT_LINKABLE
     -- Ritorna il primo elemento contenente `a_value', se esiste
   local
     current_element, pre_current: like first_element
   do
     from
       current_element := first_element;
       pre_current := Void
     invariant
       Result = Void implies (pre_current /= Void implies pre_current.value /= a_value)
     until
       (current_element = Void) or (Result /= Void)
     loop
       if current_element.value = a_value then
         Result := current_element
       end
       pre_current := current_element
       current_element := current_element.next
     end
   ensure
     (Result /= Void) implies Result.value = a_value
   end

value_precedes (a_value, target: INTEGER): BOOLEAN
     -- La lista contiene `a_value' prima della prima occorrenza di `target'?
   require
     esiste_bersaglio: get_element(target) /= Void
   local
     current_element, pre_current: like first_element
   do
     if get_element (target) /= first_element then
       from
         current_element := first_element
         pre_current := Void
       invariant
         not Result implies (pre_current /= Void implies pre_current.value /= a_value)
       until
         (current_element = get_element(target)) or Result
       loop
         if current_element.value = a_value then
           Result := True
         end
         pre_current := current_element
         current_element := current_element.next
       end
     end
   end
```

Esercizio 3) [10 punti]
Completare i contratti (pre-condizioni, post-condizioni, invarianti di classe) della classe *STOPWATCH_CONTROL* sotto descritta che modella un sistema software per controllare un cronometro conta-secondi in modo da rispecchiare la seguente specifica informale:

1. Si può operare sul cronometro attraverso il sistema software mediante i seguenti comandi: *avvio*, *arresto*, *giro*, *riprendi*, *azzera*. In corrispondenza di tali comandi il sistema software emette analoghi comandi verso il cronometro con la seguente semantica:
   - *avvio*: fa partire il cronometro dal valore corrente del tempo
   - *arresto*: ferma il cronometro al valore corrente del tempo
   - *giro*: blocca il visore del cronometro sul valore corrente del tempo mentre il cronometro continua a cronometrare
   - *riprendi*: riporta il visore del cronometro a visualizzare il valore corrente del tempo
   - *azzera*: riporta il valore del visore al valore iniziale di 0 secondi
2. Il sistema software (e quindi il cronometro) può essere in uno di questi stati:
   - *fermo*: il cronometro **non** sta misurando il tempo che scorre
   - *in_moto*: il cronometro sta misurando il tempo che scorre
3. I comandi *avvio* e *azzera* possono essere dati solo quando il sistema è nello stato *fermo*
4. I comandi *arresto*, *riprendi*, *giro* possono essere dati solo quando il sistema è nello stato *in_moto*

Non c'è relazione tra il numero di righe vuote ed il numero di contratti da scrivere. Non è necessario conoscere altro codice all'infuori di quello fornito.

```
deferred class
   STOPWATCH_CONTROL

feature {ANY}  accesso
   tempo_del_visore : INTEGER
      -- contiene il valore da visualizzare sul visore del cronometro.
   tempo_cronometrato : INTEGER
      -- contiene il valore corrente del tempo col cronometro.

feature {ANY}  stato
   fermo : BOOLEAN
      -- il cronometro non sta misurando il tempo che scorre.
   in_moto: BOOLEAN
      -- il cronometro sta misurando il tempo che scorre.

feature {ANY}  -- operazioni sul cronometro
   avvio
      -- Fa partire il cronometro dal valore corrente del tempo
      require
         ____________________________________________________________________
         ____________________________________________________________________
      deferred
      ensure
         ____________________________________________________________________
         ____________________________________________________________________
      end
```

```
   arresto
      -- Ferma il cronometro al valore corrente del tempo
      require
         _____________________________________________________________________________
         _____________________________________________________________________________
      deferred
      ensure
         _____________________________________________________________________________
         _____________________________________________________________________________
      end

   giro
      -- Blocca il visore del cronometro sul valore corrente del tempo mentre il cronometro
continua a cronometrare
      require
         _____________________________________________________________________________
         _____________________________________________________________________________
      deferred
      ensure
         _____________________________________________________________________________
         _____________________________________________________________________________
      end

   riprendi
      -- Riporta il visore del cronometro a visualizzare il valore corrente del tempo
      require
         _____________________________________________________________________________
         _____________________________________________________________________________
      deferred
      ensure
         _____________________________________________________________________________
         _____________________________________________________________________________
      end

   azzera
      -- Riporta il visore al valore iniziale di 0 secondi
      require
         _____________________________________________________________________________
         _____________________________________________________________________________
      deferred
      ensure
         _____________________________________________________________________________
         _____________________________________________________________________________
      end


invariant
_____________________________________________________________________________
_____________________________________________________________________________
_____________________________________________________________________________
```

SOLUZIONE:

```
deferred class
   STOPWATCH_CONTROL

feature {ANY}  accesso
   tempo_del_visore : INTEGER
      -- contiene il valore da visualizzare sul visore del cronometro.
   tempo_cronometrato : INTEGER
      -- contiene il valore corrente del tempo col cronometro.

feature {ANY}  stato
   fermo : BOOLEAN
      -- il cronometro non sta misurando il tempo che scorre.
   in_moto: BOOLEAN
      -- il cronometro sta misurando il tempo che scorre.

feature {ANY}  -- operazioni sul cronometro
   avvio
      -- Fa partire il cronometro dal valore corrente del tempo
      require
         fermo
      deferred
      ensure
         in_moto
      end

   arresto
      -- Ferma il cronometro al valore corrente del tempo
      require
         in_moto
      deferred
      ensure
         fermo
      end

   giro
      -- Blocca il visore del cronometro sul valore corrente del tempo mentre il cronometro
continua a cronometrare
      require
         in_moto
      deferred
      ensure
         in_moto
         tempo_del_visore = old tempo_cronometrato
      end

   riprendi
      -- Riporta il visore del cronometro a visualizzare il valore corrente del tempo
      require
         in_moto
      deferred
      ensure
         in_moto
         tempo_del_visore = tempo_cronometrato
      end
```

```
   azzera
      -- Riporta il visore al valore iniziale di 0 secondi
      require
         fermo
      deferred
      ensure
         fermo
         tempo_del_visore = 0
      end


invariant
   fermo XOR in_moto: (fermo and not in_moto) or (not fermo and in_moto)
```