Interazione con OOP

SCD

Anno accademico 2007/8
Corso di Sistemi Concorrenti e Distribuiti

Tullio Vardanega, tullio.vardanega@math.unipd.it

Inheritance anomaly – 1

- **L'interazione tra concorrenza e orientazione a oggetti (OO) produce effetti indesiderabili**
  - **Metodi che contengono codice di sincronizzazione (per mutua esclusione, sospensione, risveglio) possono essere non direttamente ereditabili senza preventiva analisi dettagliata e/o modifiche della classe sorgente**
    - S. Matsuoka and A. Yonezawa
      Analysis of inheritance anomaly in object-oriented concurrent programming languages
      In: G. Agha, P. Wegner, and A. Yonezawa (editors), **Research directions in concurrent object-oriented programming**, pp. 107-150. MIT Press, 1993
  - **Tale eventualità viola il principio di incapsulazione**

Inheritance anomaly – 2

- **Forme canoniche di sincronizzazione**
  - **Esecuzione condizionale**
    - Le operazioni effettuabili possono dipendere da
      - **Stato interno** della risorsa
      - **Storia** di esecuzione
      - **Parametri** della richiesta
    - La natura della risorsa può necessitare accesso esclusivo
  - **Controllo di ordinamento**
    - Le richieste non immediatamente soddisfacibili possono venire accodate

Inheritance anomaly – 3

- **L'anomalia ha luogo tipicamente all'introduzione di**
  - **Diversi criteri di ammissibilità**
    - La sottoclasse può aver bisogno di introdurre una corrispondenza diversa tra stati interni e operazioni ammissibili
  - **Modifica degli stati interni**
    - La sottoclasse può aver bisogno di introdurre nuovi stati interni, ignoti e non trattati dalla classe originaria
  - **Modifica delle dipendenze dalla storia d'esecuzione**
    - La sottoclasse può aver bisogno di imporre nuovi e diversi vincoli sulla ammissibilità di particolari richieste
- **Ciò richiede conoscenza precisa della logica originaria della classe originaria e la possibilità di modificarla**

Esempio

- ***Bounded buffer*** **con** `get()` **e** `put()`
  - Non vuoto → `get()`
  - Non pieno → `put()`
  - **Diversi criteri di ammissibilità**
    - Prelievo simultaneo di 2 articoli: aggiunta di metodo `get2()` e oscuramento dei metodi ereditati
      - Stato non originario → modifica della classe originaria
  - **Modifica degli stati**
    - Risorsa inaccessibile → ereditarietà multipla da oggetto con metodo `lock()`
    - Risorsa accessibile → ereditarietà multipla dal metodo `unlock()`
      - Stato non originario → nessun effetto a meno di modifica della classe originaria
  - **Modifica delle dipendenze della storia**
    - Preleva 1 articolo ogni N immissioni: aggiunta di metodo `Nget()` e oscuramento dei metodi ereditati
      - Stato non originario → modifica della classe originaria

Sulle interfacce – 1

- **Interfaccia come tipo astratto privo di membri**

```
type Int1 is interface;
procedure Op1(X: Int1) is abstract;

type DT is new Int1 with
  record
    ... -- membri aggiuntivi
  end record;
procedure Op1(NX: DT);
```

```
type LI is limited interface;
type SI is synchronized interface;
type TI is task interface;
type PI is protected interface;
```

- **Il tipo interfaccia può essere ulteriormente specializzato**

Sulle interfacce – 3

- **Caso 1: *synchronized interface***

```
type RW is limited interface;
procedure Write (Obj: out RW; X: in Item) is abstract;
procedure Read (Obj: in RW; X: out Item) is abstract;
```

L'esecuzione delle sue operazioni comporta sincronizzazione

```
type Sync_RW is synchronized interface and RW;
```

Il primo parametro dei metodi **Write** e **Read** è implicito!

```
protected type Prot_RW is new Sync_RW with
   [overriding]
   procedure Write (X: in Item);
   [overriding]
   procedure Read (X: out Item);
private
   V: Item;
end;
```


Sulle interfacce – 4

- **Caso 2: *task interface***

```
type TI is task interface;
procedure P(X: in TI) is abstract;
procedure Q(X: in TI; I: in Integer) is null;
```

```
task type TT1 is new TI with
   entry P; -- P and Q implemented by entries
   entry Q(I: in Integer);
end TT1;
```


Sulle interfacce – 5

- **Caso 3: *protected interface***

```
type Map is protected interface;
procedure Put (M: Map; K: Key; V: Value) is abstract;
```

```
protected A_Map is new Map with
   procedure Put (K: Key; V: Value);
...
end A_Map;
```


Sulle interfacce – 6

- **La gerarchia dei tipi interfaccia**
  - **Interface : il più generale**
  - **Limited interface : non ha operazioni predefinite di assegnazione e uguaglianza**
  - **Synchronized interface : la realizzazione dei suoi metodi comporta sincronizzazione**
    - Come canali tipati di *task* oppure metodi protetti
  - **Task interface : realizzabile solo attraverso canali tipati di *task***
  - **Protected interface : realizzabile solo attraverso metodi protetti**