# Esercizi: Tipi Astratti in Java e Ocaml

Sommario: 31 Maggio, 2017

Semplici Esercizi su:

- Uso dei costrutti
- Stato Concreto: AF ed I
- Additionals
- Riuso ed Estensioni

# Esercizi: Coda Polimorfa Mutable - Un differente algoritmo

## Esercizio (1)

*Stesso esercizio che in 1 di Esercizidel30-5-17 ma utilizzando uno stato concreto con un accesso diretto al last della coda. Complessità della remove diventa costante invece che lineare (come in MuQueueADTX).*
*Ci sono però altri problemi: Ad ogni add/remove dovremmo propagare i valori corretti di size e di last su tutti i componenti (ma dobbiamo visitare l'intera struttura e questo ci porta ad un costo analogo a quello della soluzione di Esercizio 1).*
*Soluzione di compromesso: Aggiorniamo size e last solo nel primo compo- nente. Gli altri componenti hanno campi size e last non significativi. Tuttavia quando il calcolo con tali componenti interni richiede il valore effettivo della size e del last definiamo le operazioni private realSize e realLast in grado di calcolare i valori richiesti.*
*Si chiede di:*
*(b) Definire un ADT, MuQueueADTXV, per l'API, MuQueueAPI, vedi Esercizio 1;*
*(c) Definire gli additionals;*
*(d) Fornire AF e I per MuQueueADTXV;*

# Esercizi: Specializzazione e Riuso di Codice

## Esercizio (2)

*La classe MuQueueADTX non pone vincoli sulla dimensione delle code nè sugli oggetti che possono accordarsi. In particolare permette code con più occorrenze di uno stesso oggetto di tipo A.*
*Si specializzi la classe MuQueueADTX in una nuova classe per code Polimorfe, Mutable, a sola presenza singola, ovvero che non ammetta occorrenze multiple di oggetti.*
*Si chiede di:*
*(a) Definire un ADT, MuQSingleADTX che specializzi MuQueueADTX aggiungendo tale proprietà*
*(b) Fornire gli additionals per MuQSingleADTX*
*(c) Fornire AF e I per MuQSingleADTX;*

# Esercizi: Coda Polimorfa Immutable in Ocaml

## Esercizio (3)

*Una queue e' un valore con le seguenti operazioni specifiche:*

*add per aggiungere un elemento alla coda*
*get per accedere il primo tra gli inseriti, non ancora rimossi*
*remove per rimuovere il primo tra gli inseriti, non ancora rimossi*
*size per conoscere quanti elementi sono ancora in coda*
*empty per conoscere se la coda ha elementi*

*Si chiede di:*
*(a) Introdurre in OCaml, un API per un tipo astratto per queue polimorfe Immutable*
*(b) Definire un ADT, ImQueueADT, per tale API;*
*(c) Definire gli additionals (clone e elements)*
*(d) Fornire AF e I per ImQueueADT;*
*(f) Provare la correttezza di I.*

# Esercizi: Coda Polimorfa e Riuso in Ocaml

## Esercizio (4)

*Analogamente a quanto visto in Esercizio 2, anche il tipo ImQueueADTX di OCaml per immutable non pone vincoli sulla dimensione delle code nè sugli oggetti che possono accedere. In particolare permette code con più occorrenze di uno stesso oggetto A.*
*In Ocaml non possiamo estendere un tipo ed ereditare operazioni dal tipo astratto esteso, tuttavia possiamo fare riuso di codice (anche se in forma limitata) ricorrendo ad un meccanismo di implementazione: Implementare il nuovo tipo utilizzando il tipo che vogliamo estendere.*
*Si applichi questo alla definizione di un nuovo tipo ImQSingleADTX con stessa API di ImQueueADTX, ma che fornisca code come ImQueueADTX ma aventi singole occorrenze.*
*In particolare si chiede di:*
*(a) Definire un ADT, MuQSingleADTX che specializzi MuQueueADTX aggiungendo tale proprietà*
*(b) Fornire gli additionals per MuQSingleADTX*
*(b) Fornire AF e I per MuQSingleADTX;*