PROGRAMMAZIONE II (A,B) - a.a. 2018-19

Secondo Appello – 13 Febbraio 2019
Traccia Soluzione

Domande di base

1. Si consideri il seguente frammento di programma scritto in una sintassi simile al C.

```
int x;

int main() {
   x = 14;
   f();
   g();
}

void f() {
   int x = 13;
   h();
}

void g() {
   int x = 12;
   h();
}

void h() {
   printf("%d\n",x);
}
```

   Si descriva il risultato osservabile dell'esecuzione del programma in caso di (*i*) scoping statico e (*ii*) scoping dinamico. Si motivi la risposta.

2. Nell'ipotesi in cui l'istruzione `MyClass mc = new MyClass()` sia la prima istruzione del metodo `main`, si descriva brevemente le azioni effettuate dalla `JVM` durante l'esecuzione dell'istruzione precedente.

3. Descrivere il ruolo e le caratteristiche del principio di sostituzione nella realizzazione di applicazioni Java.

Esercizio 1

Si consideri il tipo di dato *IndexedOrderedList* che è un contenitore di dati generici. Un `IndexedOrderedList` è una collezione finita di liste di uno specifico tipo generico. Le liste della collezione sono *indicizzate* da un intero che permette di individuare in modo univoco le liste presenti nella collezione. Inoltre si assume che le liste siano ordinate in modo decrescente. Supponiamo che l'*interfaccia* del tipo di dato `IndexedOrderedList` sia definita nel modo seguente:

```
interface IndexedOrderedList<E> {
  /* Overview: Tipo modificabile di liste generiche, ordinate in modo decrescente,
      indicizzate da un valore intero. */
  /* Typical element 0::L0, 1::L1, .... k::Lk,
     Li lista di elementi di tipo E  ordinata in modo decrescente . */

   /* Restituisce la lista di indice index nella collezione */
   List<E>  search(Integer index);

   /* Restituisice l'indice della lista elts se presente nella collezione. */
```

```
Integer  indexOf(List<E> elts);

/* Inserisce nella lista  individuata dall'indice index  l'elemento elem */
void put(E elem, Integer index)

/* altri metodi */
}
```

1. Assumendo di adottare una strategia di programmazione difensiva, si completi il progetto del tipo di dato astratto `IndexedOrderedList<E>`, definendo le clausole REQUIRES MODIFIES, e EFFECTS di ogni metodo, indicando le eccezioni eventualmente lanciate e se sono checked o unchecked.

   Traccia Soluzione Si presenta una soluzione per il problema generalizzato.

```
interface IndexedOrderedList<E> {
  /* Overview: Tipo modificabile di liste generiche, ordinate in modo decrescente,
      indicizzate da un valore intero. */
  /* Typical element: insieme delle coppie della funzione  f:INDEX -> LIST<E>,
      f(index)=L, e INDEX insieme finito di Integer
      L = lista di elementi di tipo E  ordinata in modo decrescente . */

   /* Restituisce la lista di indice index nella collezione */
   List<E>  search(Integer index);
   @REQUIRE index !=null && index appartiene a INDEX
   @THROWS NullPointerException se index = null
   @THROWS IllegalArgumentException se index non appartiene a ItendsNDEX
   @RETURN f(index)

   /* Restituisce l'indice della lista elts se presente nella collezione. */
   Integer  indexOf(List<E> elts);
   @REQUIRE elts !=null
   @THROWS NullPointerException se elts = null
   @RETURN -1 se non  esiste index f(index)=elts, index altrimenti

   /* Inserisce nella lista  individuata dall'indice index  l'elemento elem */
   void put(E elem, Integer index)
   @REQUIRE index !=null && index appartiene a INDEX
   @REQUIRE elem !=null
   @THROWS NullPointerException se index = null
   @THROWS NullPointerException se elem = null
   @THROWS IllegalArgumentException se index non appartiene a INDEX
   @MODIFY this
   @EFFECT pre(f(index)) = L && post(f(index)) = L',
   L' ordinata in modo decrescente && L'.indexOf(elem) = k


   /* altri metodi */
   }
```

2. Si consideri la seguente struttura di implementazione per la classe `MyIndexedOrderedList<E>`

```
private HashMap<Integer, ArrayList<E>> hmap;
private int size;
```

   Si definisca l'invariante di rappresentazione per l'implementazione di `MyIndexedOrderedList<E>`.

   Traccia soluzione

```
public class MyIndexedOrderedList <E extends Comparable<E>>{

/* Vincolo di estensione di Comparable serve per poter ordinare gli elementi generici */}

REP = hmap !=null && size >= 0 && hmap.size()=size &&
forall index in hmap.keySet() => hmap.get(index) !=null &&
forall lst in hmap.values() => lst != null && ( forall i in lst.size()=> lst.get(i) !=null)
 &&  (lst e' ordinato in modo decrescente)

lst e' ordinata in modo descrescente =
forall i,j in lst.size(), i < j => lst.get(i).compareTo(lst.get(j)) > 0
```

3. Si fornisca l'implementazione del costruttore e dei metodi `search` e `put` e si dimostri che le implementazioni proposte preservano l'invariante di rappresentazione.

   Traccia soluzione

```
public class MyIndexedOrderedList <E extends Comparable<E>>{

public MyIndexedOrderedList() {

private HashMap<Integer, ArrayList<E extends Comparable<E>>> hmap =
   new HashMap<Integer, ArrayList<E extends Comparable<E>>>()
private int size = 0;
}

/* Rispetta REP hmap !=null & size >=0 */

public List<E estends Comparable<E>> search(Integer index) {
if index = null throws NullPointerException();
if !(hmap.keySet().contains(index)) then throws IllegalArgumentException();
return hmap.get(index);
}

/* Rispetta REP perche' e' un metodi osservatore */

void put(E elem, Integer index) {
if (elem = null || index = null) throws NullPointerException();
if !(hmap.keySet().contains(index)) then throws IllegalArgumentException();
addSorted(hmap.get(index),elem);
/metodo privato che inserisce nella lista rispettando ordinamento */
}

/* Rispetta le proprieta' strutturali di REP */
```

Esercizio 2

Si estenda il linguaggio didattico funzionale con i costrutti opportuni per poter operare con `Stack` contenenti valori interi.

1. Si mostri come deve essere modificato l'interprete del linguaggio didattico funzionale.

Traccia soluzione

```
Modifica della sintassi astratta

type exp = .... | Stack of valuesList | Push of exp * exp | Pop exp

and valuesList= Empty | Val of  exp  * valuesList

type evT = ..... | StackVal of evT list


Modifica della funzione di valutazione

let rec eval (e:exp) (r: evT env): evT =
                     match e with
                           :
                    | Stack(l) -> StackVal(evalList l r)
                    | Push(s,arg)  ->
                        match eval s r with
                             StackVal(l) ->
                                  let val = (eval arg r) in if type(val, "int") then
                                      StackVal (val::l )  else  failwith("type error")
                              |- failwith(" stack error")

                     | Pop(s)  ->  match eval s r with
                                          StackVal(l) ->    match l with
                                                 [] --> failwith("empty stack")
                                               | v:ls  --> StackVal (ls)


and let rec evalList (l: valuesList) (r : evT env) : evT list = match l with
     Empty ->[]
   | Val(e, ls) -> let val = (eval e r) in
   if type(val, "int") then val::(evalList ls  r) else failwith("type  error")
```

Esercizio 3

Si consideri il seguente programma OCaml

```
let const = 10;;

let myfun l x y = match l with
  [] -> x
  | a::ls -> if a > 0 then x+y else y-x;;

 let checkAndApply l (f:int ->int) =
  let rec aux l f = match l with
    [] -> const
   | elem::ls -> if elem > 0 then f const else aux ls f
   in aux l f;;

let ls1 = [100;1000];;

let ls2 = [1;2;3;4];;

let const = 5;;

checkAndApply ls2 ((myfun ls1) const);;
```

1. Si mostri la struttura della pila dei record di attivazione al momento dell'invocazione della funzione identificata dal parametro formale `f`. *Vedere file RT.pdf*
2. Si determini il valore calcolato dal programma. *Il valore calcolato è* 15