

Fondamenti di informatica B (9 CFU)

Esame del 16 Luglio 2012

NOTE:

- Scrivere su ogni foglio utilizzato cognome, nome, numero di matricola (tutto in forma leggibile).
- Consegnare tutti i fogli ricevuti (anche il testo).
- Durante la prova non è consentita la consultazione di alcun tipo di materiale, né è consentito uscire.
- La prova è ritenuta sufficiente se almeno una parte significativa di ogni esercizio proposto è svolta correttamente. Il voto finale sarà espresso in ventisettesimi.
- Il tempo a disposizione per il compito è 2h 30'. Coloro i quali consegnano perdono il voto eventualmente conseguito in precedenza.

Esercizio 1 (12/40)

Si vuole realizzare la parte principale di un programma per gestire un servizio di trasporto e distribuzione di colli. Vi è una classe astratta **Item** che rappresenta una generica richiesta, estesa dalle classi concrete **BaseItem** e **TWItem** (cioè, *Time Window Item*). La classe **Service** gestisce le operazioni di spedizione.

La classe astratta **Item** ha i seguenti campi e metodi:

- **String origin**: campo che contiene l'origine della spedizione.
- **String destination**: campo che la destinazione del collo.
- Costruttore con due argomenti di tipo **String**. Qualora vi sia almeno un argomento nullo o vuoto, deve essere generata un'eccezione.
- **String getOrigin()**, **String getDestination()**: metodi concreti che restituiscono origine e destinazione del collo. Si supponga che tali metodi siano già implementati.
- **void deliver()**: metodo astratto che permette di servire la richiesta. Sarà implementato in modo specifico nelle classi derivate.

Si supponga che la classe **BaseItem** sia già implementata.

La classe **TWItem** ha i seguenti campi e metodi:

- **int after** e **int before**: campi che rappresentano, la finestra temporale in cui il collo deve essere consegnato. Per esempio, se **after** vale 8 e **before** vale 12, il pacco deve essere consegnato tra le 8:00 e le 12:00. I vincoli sulle variabili sono i seguenti: **after**, **before** $\in [7, 22]$ e **after** < **before**.

- Costruttore con quattro argomenti: due di tipo `String` e due di tipo `int`. Qualora siano inseriti valori non ammissibili per `after` e `before`, deve essere generata un'eccezione.
- `void deliver()`: si implementi il metodo mostrando a video un messaggio di testo contenente origine, destinazione e la finestra temporale di consegna.

La classe `Service` ha i seguenti campi e metodi:

- `List<Item> requests`: campo che mantiene la lista delle richieste.
- `Map<String,List<Item>> requestsByDest`: campo che mantiene memoria delle richieste suddivise per destinazione. La chiave della mappa è la destinazione, mentre il valore è dato dalla lista delle richieste relative.
- Costruttore di default.
- `void addItem(Item id)`: aggiunge una richiesta a `requests` e `requestsByDest`.
- `void deliverAllDest(String dest)`: si occupa di servire tutte le richieste per la destinazione `dest`, invocando su ogni item il metodo `deliver()` ed eliminando le relative richieste dalle strutture dati (nella mappa si elimini l'intera coppia). Per determinare l'elenco dei colli con destinazione `dest` si agisca sulla mappa.

Si implementino le classi così come descritte e si implementi anche un programma che mostri come utilizzare le classi e i metodi implementati. Definire opportunamente campi e metodi come *private*, *protected* o *public*, se non specificato nel testo. È possibile aggiungere metodi, campi e variabili qualora lo si ritenga opportuno. Si riporta uno stralcio della documentazione di interesse:

* Interface `List<E>`

`boolean add(E e)`: Appends the specified element to the end of this list.

`boolean remove(Object o)`: Removes the first occurrence of the specified element from this list, if it is present.

* Interface `Map<K,V>`

`V get(Object key)`: Returns the value to which the specified key is mapped, or null if this map contains no mapping for the key.

`V put(K key, V value)`: Associates the specified value with the specified key in this map.

`V remove(Object key)`: Removes the mapping for a key from this map if it is present.

Esercizio 2 (10/40)

Si analizzi il codice seguente e si riporti l'output prodotto dall'esecuzione del programma (che, per semplicità, si suppone scritto in un unico *file*). Si spieghi anche il motivo per cui l'ultima riga del *main* genera un'eccezione. Le risposte devono essere opportunamente motivate (per esempio tramite uno schema che riproduce legami e informazioni di tipo relative a riferimenti e oggetti).

```
class Boson {
    public Boson() {System.out.println("Class Boson");}
    public void field() {System.out.println("Boson");}
}

class Higgs extends Boson{
    public Higgs(){
        super();
        System.out.println("Class Higgs");
    }
    public void field(){System.out.println("Higgs");}
}

class StdModel {
    public StdModel(Boson obj){
        System.out.println("Test on generic boson");
        obj.field();
    }
    public StdModel(Higgs obj){
        System.out.println("Test on Higgs boson");
        obj.field();
    }
}

public class Esercizio2 {
    public static void main(String [] args){
        Boson b1 = new Higgs();
        StdModel std1 = new StdModel(b1);
        StdModel std2 = new StdModel((Higgs) b1);
        Boson b2 = new Boson();
        StdModel std3 = new StdModel((Higgs) b2); /* eccezione runtime! */
    }
}
```

Esercizio 3 (8/40)

Si completi l'implementazione sotto riportata del pannello di una semplice applicazione la cui interfaccia grafica è costituita da un campo di testo (`txt`) e un oggetto `JCheckBox` `ck`. Quando il cursore è su `txt`, alla pressione del tasto ENTER della tastiera deve essere eseguita la seguente operazione: se `ck` è selezionata, il testo contenuto in `txt` deve essere cancellato (cioè, si scriva una stringa vuota in `txt`). Il numero di volte in cui si può premere ENTER è limitato a `n`, il cui valore è impostato nel costruttore del pannello (v. codice sorgente). Quando tale limite è raggiunto, non deve in ogni caso essere eseguita alcuna operazione sul campo di testo. Per verificare se la checkbox è selezionata, è possibile utilizzare il metodo boolean `isSelected()`.

```
public class MyPanel extends JPanel implements ActionListener {
    JTextField txt;
    JCheckBox ck;
    int n;

    public MyPanel(int n) {
        ...[1]...
    }

    public void actionPerformed(ActionEvent e) {
        ...[2]...
    }
}
```

Esercizio 4 (10/40)

a) Analizzare il seguente programma e indicare quali sono i valori stampati a tempo di esecuzione. Si motivi opportunamente la risposta.

```
#include <stdio.h>
#include <stdlib.h>

#define N 6

int k = 2;

int lhc(int a, char *vec)
{
    int i;

    i = k++;
    while (i < a){
        if (i % 2 == 0){
            vec[i] = 'H';
        }
        i++;
    }
    return k;
}

int main()
{
    char *x;
    int k,n;

    x = (char *) malloc(sizeof(char) * N);
    for (k = 0; k < N-1; k++) x[k] = 'z';
    x[k] = '\\0';

    n = lhc(k,x);

    printf("%s,%d,%d",x,n,k);

    return 0;
}
```

b) Data la funzione:

```
int cern(int a, int b)
{
    if (a < b) return a;
    else return b + cern(b,a);
}
```

si scriva il risultato della funzione quando invocata come `cern(4,3)` e si disegnino i corrispondenti record di attivazione. Si supponga che le espressioni siano valutate in ordine da sinistra verso destra. Indicare anche se la funzione è *tail ricorsiva* e motivare la risposta. Si osservi che la funzione è mal posta, poiché potrebbe incorrere in *loop* infiniti: per quale insieme di valori iniziali questo accade?