Una Soluzione Graphk.java

Chi ha continuato lo sviluppo della classe Graphk avrà certamente realizzato che la ridefinizione del metodo removeN richiede una maggiore raccolta di informazione durante la costruzione e vita dei valori Graphk. Questo si ripercuote sullo Stato Concreto di Graphk che deve avere una struttura più ampia di quella assunta inizialmente e limitata alle sole strutture:

```
private Vector<Pair<Node<A>,Integer>> outd;
private int bound;
```

In effetti la ridefinizione in Graphk, di removeN richiede di conoscere la lista degli archi incidenti il nodo rimosso. Questa lista non é ottenibile dalle operazioni di Graph comunque composte tra loro. La soluzione è che Graphk sia in grado di costruirla utilizzando il proprio Stato Concreto. Questo è ottenuto ampliando lo Stato Concreto con un ulteriore filed che mantiene traccia di tutti gli archi correntemente nel grafo, ovvero, in aggiunta ai field `outd e bound` introduciamo:

```
private Vector<Edge<A,B>> egs;
```

Una Soluzione Graphk.java: pack5-6-18

Il pack5-6-18 fornisce la struttura generale per uno sviluppo avanzato del pack1-6-18 distribuito venerdì scorso: In particolare contiene:

- OCaml: Solutions con la soluzione completa di Graph e della sua API, raccolti in un unico file GRAPH.ml

- Java: Solutions con un pack1 avanzato contenente:
  > GraphAPI, NodeAPI, EdgeAPI e Tests, tutti in codice sorgente, relativi all'introduzione del Tipo Astratto Graph;
  > Codice oggetto degli ADT Graph, Node, Edge in GraphPack, utilizzabili per la programmazione con valori Graphs come in Tests o in Graphk che deve estendere Graph
  > Sottoclasse Graphk, con Stato Concreto esteso, AF e I, costruttore, metodi addN e AddE già ridefiniti. La sottoclasse deve essere completata con la ridefinizione dei rimanenti metodi di Graph.
  > Testsk in codice sorgente da compilare ed usare al completamento di Graphk per il test e la programmazione con valori Graphk.

La soluzione finale sarà distribuita venerdì prossimo.