Cognome e Nome _____

Matricola _____

Programmazione e Algoritmi 2 A.A. 2024/2025

Appello del 13 febbraio 2025 in Java

Esercizio 1 [6 punti, da svolgere su questo foglio]

Si considerino le seguenti classi.

```
class A {
      protected int n;

      public A(int n) {
            this.n = n + 2;
      }

      public A(int n, int m) {
            this(n - m);
      }

      public int metodo(A a) {
            n = n - a.n;
            return n;
      }
}
```

1

```
class B extends A {
      public B(int n) {
            this(n, n + 1);
      }

      public B(int n, int m) {
            super(m, n);
      }

      public int metodo(A a) {
            n = n + a.n;
            return n;
      }

}
```

2

Si dica cosa viene stampato a video dal seguente codice:

```
      A a = new A(1, 2);
      B b = new B(3);
      A ab = new B(4, 5);
      System.out.println(a.metodo(ab));
      System.out.println(ab.metodo(b));
      System.out.println(ab.metodo(ab));
```

Si giustifichi la risposta mostrando in particolare le firme associate a tempo di compilazione e a tempo di esecuzione ad ogni chiamata di metodo:

	Tempo di compilazione	Tempo di esecuzione
`a.metodo(ab)`		
`ab.metodo(b)`		
`ab.metodo(ab)`		

Esercizio 2 [6 punti]

Mostrare **passo-passo l'esecuzione dell'heap-sort** per ordinare l'array **[17, 30, 8, 1, 9, 7, 2]** in modo non decrescente (compresa la fase iniziale di **build-max-heap**).

Esercizio 3 [11 punti]

Si vogliono gestire, in Java, i movimenti associati al credito di SIM di telefonia mobile.
Si assuma, senza scrivere nulla, l'esistenza di una classe astratta **Movimento** con

- una variabile di istanza **idSim** (tipo String, private), che contiene il numero telefonico associato alla SIM a cui il movimento si riferisce;
- una variabile di istanza **tempo** (tipo long, private), che rappresenta l'istante del movimento espresso in secondi trascorsi dalle ore 00:00 del 1° gennaio 2000.

e i seguenti metodi di istanza:

- un costruttore che crea un movimento dati campi **idSim** e **tempo**;
- metodo di accesso **`public String getIdSim()`**.
- metodo di accesso **`public long getTempo()`**.
- metodo pubblico astratto **`public abstract double getImporto()`** che restituisce l'importo in Euro del movimento. In particolare, se il movimento è un movimento di spesa viene restituito un valore minore o uguale a zero (ogni sms costa 0,25 €, mentre ogni chiamata costa 0,10€ di scatto alla risposta più mezzo centesimo per ogni secondo di durata), mentre se è un movimento di ricarica credito viene restituito un valore positivo.

La classe astratta **Movimento** è estesa dalle sottoclassi (non astratte) **Chiamata**, **Sms** e **Ricarica.**
Si tenga conto del fatto che la classe **Chiamata** ha:

- una variabile di istanza **durata** (tipo int, private) che contiene la durata in secondi della chiamata;
- una variabile di istanza **destinatario** (tipo String, private), che contiene il numero chiamato.

La classe **Sms** deve avere:

- una variabile di istanza **destinatario** (tipo String, private), che contiene il destinatario dell'SMS.

La classe **Ricarica** deve avere:

- una variabile di istanza **importo** (tipo double, private), che contiene l'importo della ricarica e deve valere almeno 0,01.

Si assuma, senza scrivere nulla, che tutte le sottoclassi abbiano i costruttori (che prendono in input gli opportuni parametri e che richiamino opportunamente il costruttore della classe **Movimento**), i metodi di accesso per tutti i campi.

[4 punti] Si scriva l'implementazione del metodo **`double getImporto()`** per tutte le sottoclassi **Chiamata**, **Sms** e **Ricarica** di Movimento.

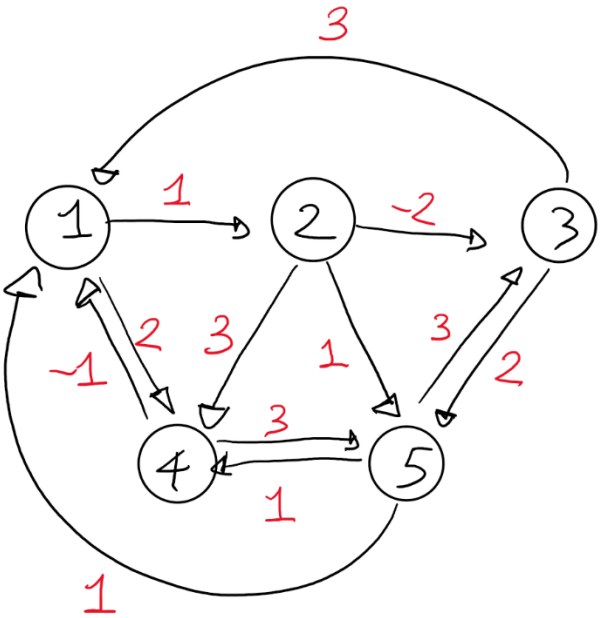
[7 punti] Si scriva una classe **Gestore** con

- una variabile di istanza **movimenti** (tipo ArrayList<Movimento>, private);

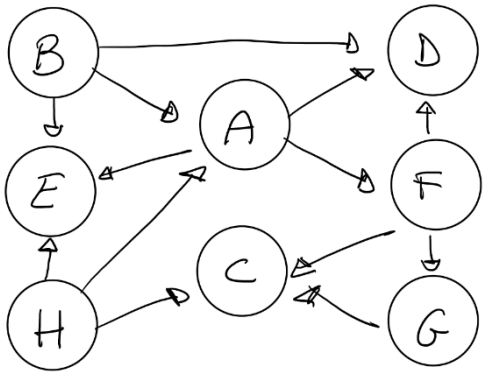
Implementare i seguenti metodi di istanza:

- un costruttore senza parametri che crea un ArrayList **movimenti** vuoto.
- un metodo **`public double totaleRicariche (String idSim)`** che restituisce il totale delle ricariche effettuate per la sim **`idSim`** (ottenuto sommando tutti gli importi dei movimenti di ricarica nell'ArrayList **movimenti** relativi alla sim **`idSim`**).
- un metodo **`public ArrayList<String> numeriChiamati (String idSim)`** che restituisce un ArrayList contenente, senza ripetizioni, tutti gli identificativi (i numeri) delle SIM destinatari di almeno una chiamata o un SMS da parte della sim **idSim**.

Esercizio 4 [11 punti]



a. **[6 punti]** Mostrare una possibile esecuzione **passo-passo** dell'algoritmo di **Bellman-Ford** per calcolare i cammini minimi a partire dalla sorgente **2.** Mostrare, per ogni iterazione completa che si fa sugli archi, l'evoluzione del **vettore delle distanze** e del **vettore dei padri**, assumendo di visitare, in ogni iterazione, gli archi in ordine crescente rispetto alle etichette dei nodi da cui escono e, a parità di nodo di partenza, in ordine crescente rispetto all'etichetta del nodo di arrivo. Dire, giustificando la risposta, se il grafo possiede cicli di peso negativo.



b. **[5 punti]** Calcolare un ordinamento topologico del DAG in figura, usando il metodo della visita in profondità ed evidenziando per ogni nodo i timestamp di inizio e fine visita. Si assuma di iniziare e riprendere la visita sempre dal nodo (non ancora visitato) avente l'etichetta alfabeticamente più piccola.

Regole per lo svolgimento della prova scritta:

- Per svolgere il compito si hanno a disposizione **150** minuti.
- Scrivere **subito** nome, cognome, matricola su OGNI FOGLIO (**compreso questo**).
- Durante la prova scritta **non** è possibile abbandonare l'aula.
- Non è ammesso **per nessun motivo** comunicare in qualsiasi modo con altre persone
- È possibile consultare appunti, libri e dispense.
- Qualsiasi strumento elettronico di calcolo o comunicazione (telefoni cellulari, calcolatrici, palmari, computer, etc…) deve essere **completamente disattivato** e **depositato in vista sulla cattedra**
- Mettere in vista sul banco un valido documento di identità.