

Cognome e Nome _____

Matricola _____

Appello del 18 gennaio 2023

Esercizio 1 [6 punti]

Si considerino le seguenti classi.

```
class A {
    protected String s;

    public A(String s) {
        this.s = s;
    }
    public A metodo(A a) {
        return new A(a.s + "-" + s);
    }
    public B metodo(B b) {
        return new B(b.s + "-" + s);
    }
    public String toString() {
        return s;
    }
}
```

```
class B extends A {
    public B(String s) {
        super("<" + s + ">");
    }
    public A metodo(A a) {
        return new A(s + "+" + a.s);
    }
    public B metodo(B b) {
        return new B(s + "+" + b.s);
    }
}
```

Si dica cosa viene stampato a video dal seguente codice:

```
A a = new A("pippo");
B b = new B("topolino");
A ab = new B("pluto");
System.out.println(a.metodo(b));
System.out.println(a.metodo(b.metodo(b)));
System.out.println(ab.metodo(b.metodo(b)));
```

Si giustifichi la risposta mostrando in particolare:

- le firme associate a tempo di compilazione e di esecuzione ad ogni chiamata di metodo
- l'evoluzione della memoria nelle parti stack ed heap.

Esercizio 2 [5 punti]

Mostrare **passo-passo l'esecuzione dell'heap-sort** per ordinare l'array **[76, 32, 98, 17, 77, 99, 74, 43, 5, 10]** in modo non decrescente (compresa la fase iniziale di build-max-heap).

Esercizio 3 [11 punti]

Si vogliono gestire, in Java, gli spettacoli di un cinema/teatro con una sala avente capienza di 524 posti.

Si assuma, senza scrivere nulla, l'esistenza di una classe **Persona** con

- una variabile di istanza **nome** (tipo String, private);
 - una variabile di istanza **cognome** (tipo String, private);
- e i seguenti metodi di istanza:
- un costruttore che crea un oggetto dati **nome e cognome**;
 - metodi pubblici accessori **getNome ()** e **getCognome ()**;
 - metodo **equals** che restituisce *true* se e solo se due oggetti della classe Persona hanno uguale il nome e uguale il cognome.

Si scriva una classe astratta **Spettacolo** con

- una variabile di istanza **spettatori** (tipo Array di Persona, private);
- una variabile di istanza **titolo** (tipo String, private);
- una variabile di istanza **giorno** (tipo int, private), che rappresenta il giorno della data dello spettacolo contando i giorni trascorsi dal 1° gennaio 2000.

e i seguenti metodi di istanza:

- un costruttore che crea uno spettacolo dato **titolo** e **giorno**, con l'array di spettatori di lunghezza **524** contenente *null* in tutte le posizioni;
- metodo di accesso **public int getGiorno ()**.
- un metodo pubblico **int aggiungiSpettatore(Persona p, int pos)** che aggiunge uno spettatore nel primo posto libero a partire dalla posizione **pos** (considerando l'array come circolare) . Il metodo restituisce la posizione assegnata alla persona, se è stato possibil assegnarla, e -1 altrimenti.
- metodo pubblico **int getTotaliSpettatori ()** che restituisce il numero totale di spettatori coinvolti.
- metodo pubblico astratto **int getTotaliPersone ()** che restituisce il numero totale di persone coinvolte (tenendo conto sia degli spettatori che, eventualmente, del cast).

La classe astratta **Spettacolo** è estesa dalle sottoclassi (non astratte) **Proiezione** e **Rappresentazione**.

Si tenga conto del fatto che la classe **Proiezione** deve avere:

- una variabile di istanza **durata** (tipo int, private) che contiene la durata in minuti della proiezione;

La classe **Rappresentazione** deve avere:

- una variabile di istanza **cast** (tipo ArrayList<Persona>, private) che contiene la lista degli attori coinvolti nella rappresentazione.

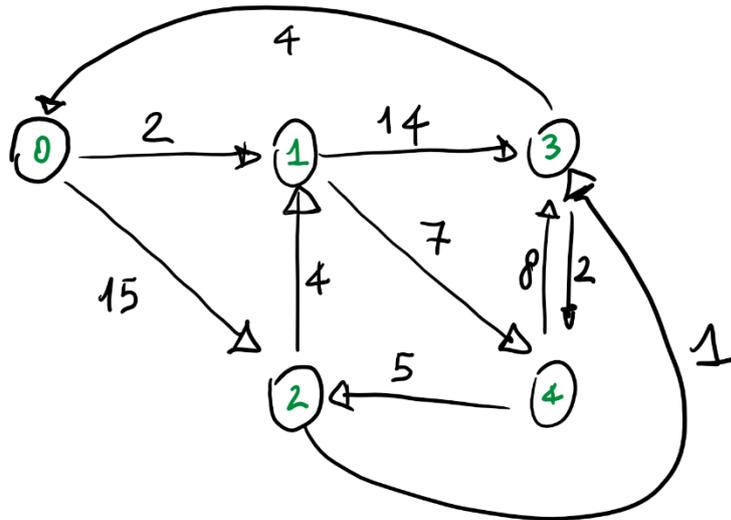
Per tutte e due le sottoclassi vanno scritti i costruttori (che prendono in input la durata e l'ArrayList del cast, rispettivamente) che richiamino opportunamente il costruttore della classe **Spettacolo** e va implementato il metodo **int getTotaliPersone ()**.

Si progetti infine una classe **CinemaTeatro** con

- una variabile di istanza **spettacoli** (tipo ArrayList<Spettacolo>, private);
- Implementare i seguenti metodi di istanza:
- un costruttore senza parametri che crea un ArrayList **spettacoli** vuoto.
 - un metodo **public ArrayList<Spettacolo> inIntervallo (int giornoDa, int giornoA)** che restituisce un arrayList contenente tutti i soli gli spettacoli dell'ArrayList compresi tra **giornoDa** e **giornoA**, estremi inclusi. Se **giornoA** è minore di **giornoDa** viene restituito un ArrayList vuoto.
 - un metodo **public ArrayList<Spettacolo> piuSeguiti ()** che restituisce gli spettacoli (tra quelli nell'arraylist **spettacoli**) che sono stati seguiti da più spettatori.

Esercizio 4 [12 punti]

Si consideri il grafo diretto in figura.



- [6 punti] Mostrare una possibile esecuzione **passo-passo** dell'algoritmo di **Dijkstra** per i cammini minimi da singola sorgente, a partire dal nodo sorgente **0**. Ad ogni passo, bisogna mostrare il contenuto della coda con priorità utilizzata dall'algoritmo.
- [6 punti] Mostrare una possibile esecuzione **passo-passo** dell'algoritmo di **Bellman-Ford** per i cammini minimi da singola sorgente, a partire dal nodo sorgente **3**.

Regole per lo svolgimento della prova scritta:

- Per svolgere il compito si hanno a disposizione **120** 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.