Cognome e Nome _____

Matricola _____

Programmazione e Algoritmi 2 A.A. 2021/2022

Appello del 12 settembre 2022

Esercizio 1 [5 punti]

Si considerino le seguenti classi.

```
class A{
    int method(A a){
     return 1;
    }
    int method(B a){
     return 2;
    }
}
```

```
class B extends A{
    int method(A a){
     return 3;
    }
}
```

Si dica cosa viene stampato a video dal seguente codice:

```
      A a = new A();
      B b = new B();
      A c = new B();
      System.out.println(b.method(b));
      System.out.println(b.method(c));
      System.out.println(c.method(a));
      System.out.println(c.method(c));
```

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

Esercizio 2 [7 punti]

a. Mostrare **l'heap di massimo** che si ottiene a partire dall'heap vuoto inserendo nell'ordine indicato le chiavi **6, 32, 8, 45, 2, 80, 9, 7, 43** (mostrare l'heap che si ottiene dopo l'inserimento di ogni chiave);

b. Mostrare **passo-passo l'esecuzione dell'heap-sort** per ordinare l'array **[6, 32, 8, 45, 2, 80, 9, 7, 43]** in modo non decrescente (compresa la fase iniziale di build-max-heap).

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.

Esercizio 3 [12 punti]

Si vogliono gestire, in Java, gli ordini di un negozio di fast food.

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

- una variabile di istanza **descrizione** (tipo String, private);
- una variabile di istanza **prezzoUnitario** (tipo double, private);

e i seguenti metodi di istanza:

- un costruttore che crea un oggetto dati **descrizione e prezzoUnitario**;
- metodi pubblici accessori **`getDescrizione() e getPrezzoUnitario()`**;

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

- una variabile di istanza **prodotto** (tipo Prodotto, private)
- una variabile di istanza **quantita** (tipo int, private)
- un costruttore che crea un oggetto dati **prodotto e quantita**;
- metodi pubblici accessori **`getProdotto() e getQuantita()`**.

Si scriva una classe astratta **Ordine** con

- una variabile di istanza **comanda** (tipo ArrayList<ProdottoOrdinato>, final, private);

e i seguenti metodi di istanza:

- un costruttore senza parametri che crea una comanda vuota associandola all'ordine;
- un metodo pubblico **`void aggiungi(Prodotto p, int quantita)`** che aggiunge alla comanda il prodotto indicato con la relativa quantità.
- metodo pubblico astratto **`double getTotale()`**che restituisce l'importo totale dell'ordine.

La classe astratta **Ordine** è estesa dalle sottoclassi (non astratte) **OrdineTavolo, OrdineAsporto e OrdineDelivery.**

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

- una variabile di istanza **tavolo** (tipo String, private) che contiene l'identificativo del tavolo;
- una variabile di istanza **coperti** (tipo int, private) che contiene il numero di coperti.

La classe **OrdineAsporto** deve avere:

- una variabile di istanza **nominativo** (tipo String, private) che contiene il nominativo della persona che ha effettuato l'ordine d'asporto.

La classe **OrdineDelivery** deve avere:

- una variabile di istanza **destinazione** (tipo String, private) che contiene il nominativo e l'indirizzo di destinazione dell'ordine.

Per tutte e tre le sottoclassi vanno scritti opportunamente i costruttori che richiamano opportunamente il costruttore della classe **Ordine** e va implementato il metodo **`double getTotale(),`** tenendo conto del fatto che ogni coperto ha un costo di € 1,80 (nel caso di OrdineTavolo), la consegna ha un costo di € 4,00 (nel caso di OrdineDelivery) mentre non c'è nessun costo aggiuntivo per gli ordini da asporto.

Si progetti infine una classe **FastFood** con

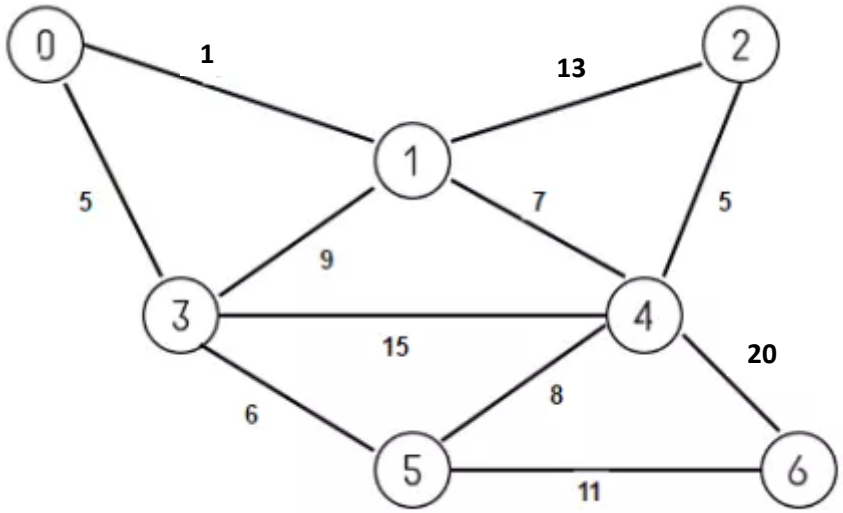
- una variabile di istanza **prodotti** (tipo ArrayList<Prodotto>, final, private);
- una variabile di istanza **ordini** (tipo ArrayList<Ordine>, final, private);

Implementare i seguenti metodi di istanza:

- un costruttore senza parametri che crea arraylist vuoti per i prodotti e gli ordini.
- un metodo **public double totaleAsporto ()** che restituisce il totale degli importi degli ordini da asporto.
- un metodo **public Prodotto piuVenduto ()** che restituisce il prodotto (tra quelli nell'arraylist prodotti) che è stato venduto di più negli ordini (come somma di tutte le quantità a lui associate nei vari ordini). *(Si assuma che i prodotti associati agli ordini si riferiscano allo stesso oggetto cui si riferiscono i prodotti presenti nell'arraylist prodotti.)*

Esercizio 4 [10 punti]

Si consideri il grafo non diretto in figura.



a. **[5 punti]** Mostrare una possibile esecuzione **passo-passo** dell'algoritmo di **Prim** per il minimo albero ricoprente, partendo dal nodo sorgente **3**. Ad ogni passo, bisogna mostrare il contenuto della coda con priorità utilizzata dall'algoritmo.

b. **[5 punti]** Mostrare una possibile esecuzione **passo-passo** dell'algoritmo di **Dijkstra** per i cammini minimi da singola sorgente, a partire dal nodo sorgente **3**. Ad ogni passo, bisogna mostrare il contenuto della coda con priorità utilizzata dall'algoritmo.