Guida alla soluzione (testo – sfida pg.376 / mini-guida pg. 21) senza **gestione data**

Un negozio di elettrodomestici desidera poter produrre fattura per gli acquisti di un dato cliente.

Ogni fattura è caratterizzata dal nome cliente, dal codice progressivo, da quantità, descrizione e prezzo unitario dei prodotti acquistati, dall'imponibile, dall'iva (pari al **23,5%**) e dal totale lordo.

Progettare (dall'illustrazione con UML all'implementazione di un segmento significativo) una soluzione che preveda la ***registrazione delle fatture*** per un dato cliente/società, permettendo di **stampare la fattura individuata da un dato codice.**

Modifiche - 3 B

Una farmacia desidera poter produrre fattura per i farmaci acquistati da un dato cliente.

Ogni fattura è caratterizzata dal nome cliente, dal codice progressivo, dal numero dei medicinali acquistati, descrizione e prezzo unitario, dall'imponibile, dall'iva (**ridotta pari al 13%**) e dal totale lordo.

Livello1: ***semplificando al massimo nei limiti di tempo … poi analisi più approfondita***

Si pensi ad una classe **Fattura** da illustrare con UML *(diagramma statico evidenziando attributi e metodi)*

*Tra i metodi significativi: un **costruttore parametrico** che non inizializzi né **l'imponibile** né il **totale lordo** che si prevedono calcolati (eventualmente da metodi dedicati). Semplificando al massimo anche il nome del cliente (pensato unico) non verrà inizializzato.*

Esplicitare in linguaggio naturale i comportamenti (*metodi*) comuni degli oggetti di tipo Fattura

*Metodi per **stampare la singola voce** presente nella fattura (per data tipologia di articolo) ed eventualmente i metodi dedicati di cui sopra: uno che calcola l'**imponibile** (cioè la spesa per tipologia di articolo)*

Descrizione	Quantità	Prezzo unitario	Prezzo totale
Articolo n. 1	1	200,00 €	200,00 €
Articolo n. 2	2	200,00 €	400,00 €

*l'altro che calcola il **totale lordo** (spesa ivata per tipologia di articolo)*

Implementare in linguaggio Java: la classe Fattura (almeno gli *attributi e metodi* necessari alla soluzione)

Livello 2: (*senza lettura da tastiera*)

Pensare ad un ***insieme*** di oggetti di tipo Fattura al fine di registrare fatture di dato cliente e *stampare*[1] quella con dato codice

Semplificando al massimo: tale insieme è *inizializzato da programma*

potrebbe essere pensato come variabile istanza (attributo comune ad oggetti di tipo UsaFattura)

*Eventualmente il **codice numerico** di ogni fattura è un **contatore auto-incrementante** (da non confondersi con l'indice dell'array)*

il nome dell'unico cliente è *impostato da programma*

il *codice noto* (della fattura da stampare) è *impostato da programma*

Illustrare con UML la classe **UsaFattura** individuando gli attributi e metodi significativi (in particolare la *ricerca*)

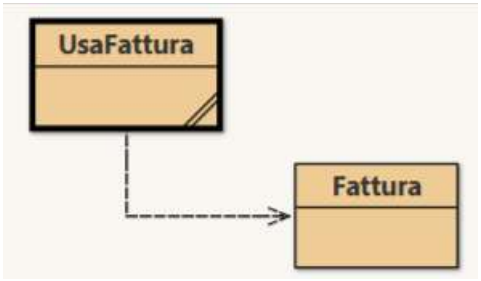
*Proporre metodi dedicati: uno che calcola l'**imponibile** (cioè la spesa complessiva) come somma i cui addendi sono il prodotto, per ogni articolo, tra numero e prezzo unitario; l'altro che calcola il **totale lordo** (saldo relativo all'acquisto di più tipologie di articoli)*

Implementare in linguaggio Java: la creazione dell'insieme di oggetti di tipo Fattura

*il/i metodo/i utili per la **ricerca** della fattura da stampare*

[1] Si intenda *visualizzare a monitor*, vedremo in seguito come *salvare su file di testo* (mini-guida pg.33)

Una farmacia desidera poter produrre fattura per i farmaci acquistati da un dato cliente.



Fattura

Attributi e metodi significativi evidenziati con commenti Javadoc nel listing propri di ogni oggetto della classe
oltre al costruttore parametrico public **Fattura** (int c, String d, int q, double pu){ ..} *che permette di settare:*

- ***il codice numerico*** *(pensato come* ***contatore auto-incrementante*** *passato alla chiamata, nell'applicazione che userà un* ***insieme*** *di oggetti di tipo Fattura)*
- *il nome del cliente*
- *la quantità della voce in acquisto*
- *il prezzo unitario*

… mentre l'importo sarà calcolato allo scopo di poterlo stampare in una sezione della fattura


```
class Fattura {
   /**
    *  codice numerico  che identifica la fattura
    */
   private int codice;
   /**
    *  cognome del cliente
    */
   private String cliente;
   private String descrizione;   // variabili istanza o attributi della singola voce in fattura
   private int quantita;
   private double prezzoUnitario;
   private double imponibile;
   private double valIVA;
   private double IVA =13.00/100; // ridotta per medicinali
   private double totaleLordo;
   /**
    * costruttore parametrico per inizializzare le variabili istanza per data tipologia di acquisto
    * @param c  codice della fattura
    * @param d  nominativo del cliente
    * @param q  quantità
    * @param pu prezzo unitario
    */
   public Fattura (int c, String d,int q,double pu){
      codice = c;
      descrizione =d;
      quantita = q;
      prezzoUnitario = pu;
   }
```

```
  public void setDescr(String s){  descrizione = s;  }
   public void setQuantita(int i){ quantita = i;  }
   public void setPrezzoUnitario(double d){ prezzoUnitario = d; }
  public void setIVA(double val){
     IVA =val;
  }
  public double getIVA(){
     return IVA;
  }
  /**
   * metodo d'accesso
   * @return codice - identifica la fattura (solitamente più voci)
   */
  public int getCodice(){
     return codice;
  }
  /**
   * metodo per calcolare imponibile (importo) e totale lordo (importo ivato)
   * della singola voce in fattura
   */
  public void calcola(){
    imponibile = quantita * prezzoUnitario;
    valIVA = imponibile * IVA;
    totaleLordo = imponibile + valIVA;
  }
  /**
   * metodo d'accesso
   * @return imponibile - spesa non ivata della singola voce in fattura
   */
  public double getImponibile(){
     return imponibile;
  }
  /**
   * metodo d'accesso
   * @return totaleLordo - spesa ivata della singola voce in fattura
   */
  public double getTotaleLordo(){
     return totaleLordo;
  }
  /**
   * metodo che stampa la sezione della fattura per data tipologia di acquisto  con formattazione
   * @return s - stampa della singola voce in fattura
  */
  public String toString()  {
     String s = "" + "\n**********************************";
     s = s + "\nFornitura di N." + quantita  +" "+ descrizione;
     s = s+ "\nAl prezzo unitario di Euro "  + prezzoUnitario;
     s = s +"\n**********************************";
     s = s+ "\n ";

     return s;
  }
} // fine class Fattura
```

```
**********************************
Fornitura di N.3 aspirina
Al prezzo unitario di Euro 9.25
**********************************
```

Producendo documentazione (IDE BlueJ)

Class Fattura

java.lang.Object
    Fattura

```
class Fattura
extends java.lang.Object
```

Constructor Summary

Constructors

Constructor	Description
`Fattura(int c, java.lang.String d, int q, double pu)`	costruttore parametrico della classe Fattura per inizializzare le variabili istanza per data tipologia di acquisto

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method	Description
void	`calcola()`	metodo per calcolare imponibile (importo) e totale lordo (importo ivato) della singola voce in fattura
int	`getCodice()`	metodo d'accesso
double	`getImponibile()`	metodo d'accesso
double	`getIVA()`	
double	`getTotaleLordo()`	metodo d'accesso
void	`setDescr (java.lang.String s)`	
void	`setIVA(double val)`	
void	`setPrezzoUnitario (double d)`	
void	`setQuantita(int i)`	
java.lang.String	`toString()`	metodo che stampa la sezione della fattura per data tipologia di acquisto con formattazione

UsaFattura

```
/**
 * Applicazione
 *
 * @author 3INF
 * @version 1
 */
public class UsaFattura{
   /**
    * metodo che stampa la sezione finale della fattura
    * con formattazione
    */
   public void stampaFattura(double somma, double sommaTot, double IVA)  {
      System.out.println (" ");
      System.out.println ("---------------------------------");
      System.out.println ("Totale imponibile..... "  + somma);
      System.out.print ("IVA % .............");
      System.out.printf("%8.2f  %n", IVA*somma);                    // visualizza[2] solo due cifre decimali
      System.out.println ("Totale IVA compresa .. "   + sommaTot);
      System.out.println ("*********************************");
      System.out.println (" ");
   }
   /*
    * metodo principale per testare la classe Fattura
    */
   public static void main(String arg[])  {
      UsaFattura ogg = new UsaFattura();
      String s ="Rossi"; // cliente Sig.Rossi ad esempio 2 fatture
                         // e almeno una fattura con più voci
       double somma = 0, sommaTot =0;
      // parametri previsti: int c, String d,int q,double pu
      Fattura [] f ={ new Fattura (1,"aspirina", 3, 9.25),
                      new Fattura (1,"cerotti", 4, 4.06),
                      new Fattura (2,"vitamine", 10, 3.60)
                    };
       System.out.println("IVA ridotta applicata ai medicinali "+ f[0].getIVA());

       // per tutte le voci calcoli parziali
       for (int i = 0; i<f.length; i++)
              f[i].calcola();

       // per fattura con cod =1
       int cod = 1;

       System.out.println("Fattura n."+ cod +" di "+ s);
       for (int i = 0; i<f.length; i++)                    // scansione di un insieme alla ricerca di più occorrenze
          if (f[i].getCodice() == cod) {
                      System.out.print(f[i].toString());
                      somma = somma + f[cod].getImponibile();
                      sommaTot = sommaTot + f[cod].getTotaleLordo();
           }
      ogg.stampaFattura (somma, sommaTot, f[cod].getIVA());
   }
} // fine dell'applicazione
```

```
---------------------------------
Totale imponibile..... 32.48
IVA % .............    4,22
Totale IVA compresa .. 36.7024
*********************************
```

```
IVA ridotta applicata ai medicinali 0.13
Fattura n.1 di Rossi
```

[2] Si potrebbe non solo visualizzare ma *arrotondare i risultati dei calcoli a* due cifre decimali

Producendo documentazione (IDE BlueJ)

Constructor Summary

Constructors

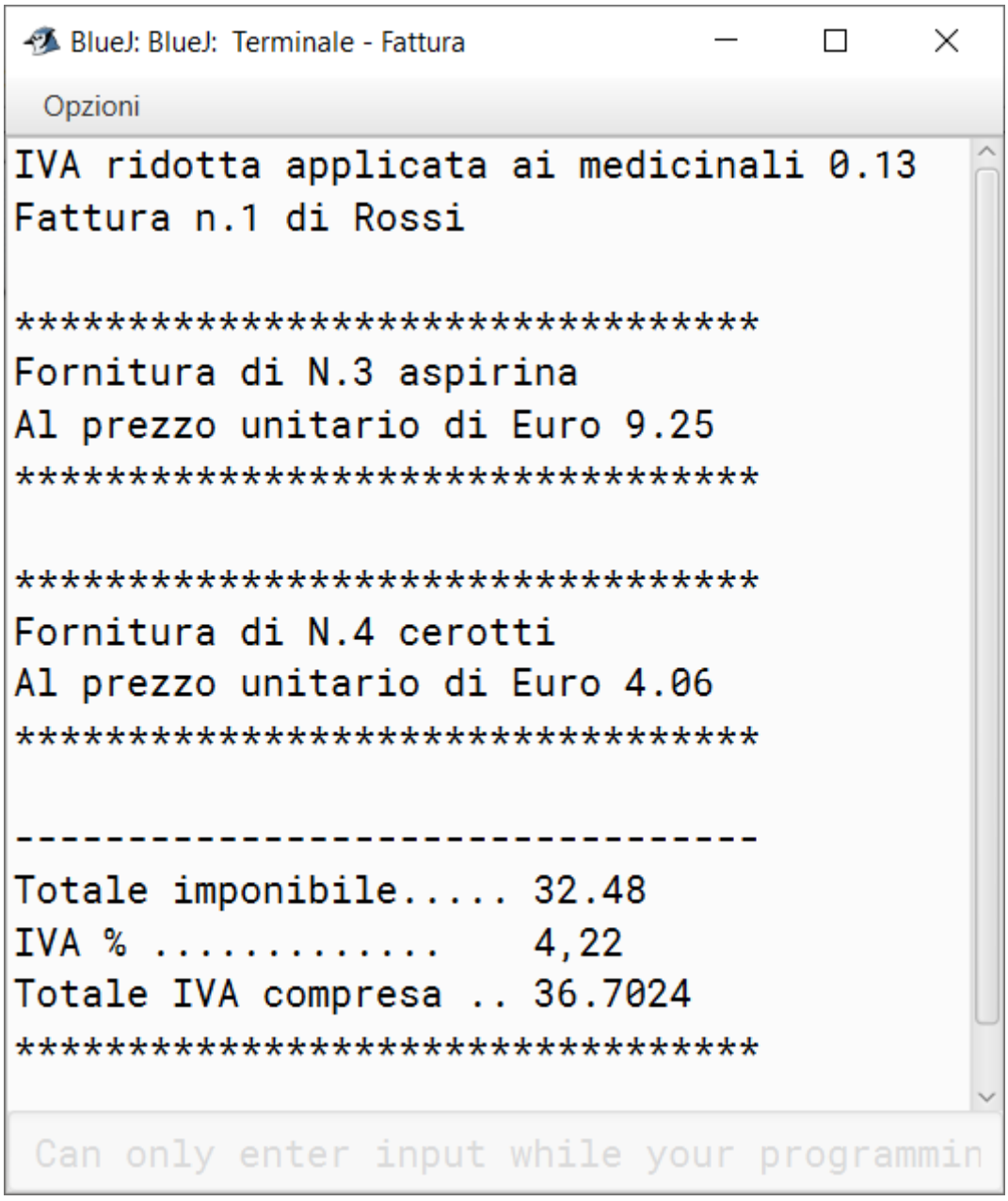
Constructor	Description
`UsaFattura()`	

Method Summary

All Methods | **Static Methods** | **Instance Methods** | **Concrete Methods**

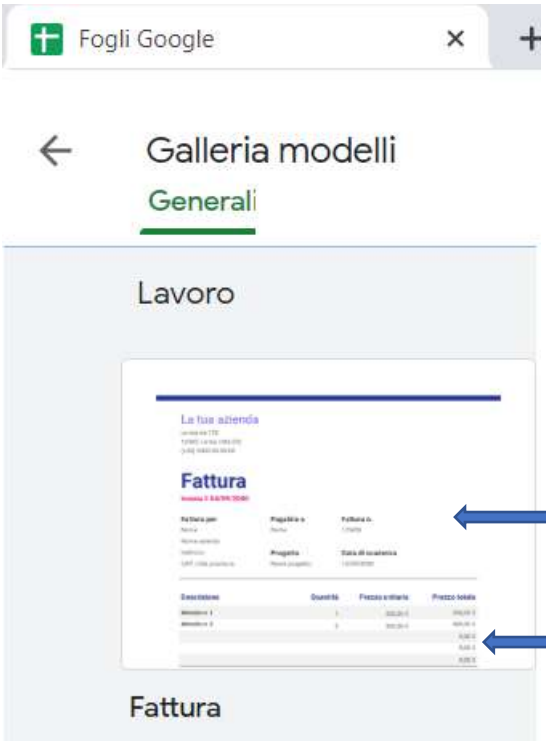
Modifier and Type	Method	Description
`static void`	`main(java.lang.String[] arg)`	
`void`	`stampaFattura(double somma, double sommaTot, double IVA)`	metodo che stampa la sezione finale della fattura con formattazione

Output (IDE BlueJ):



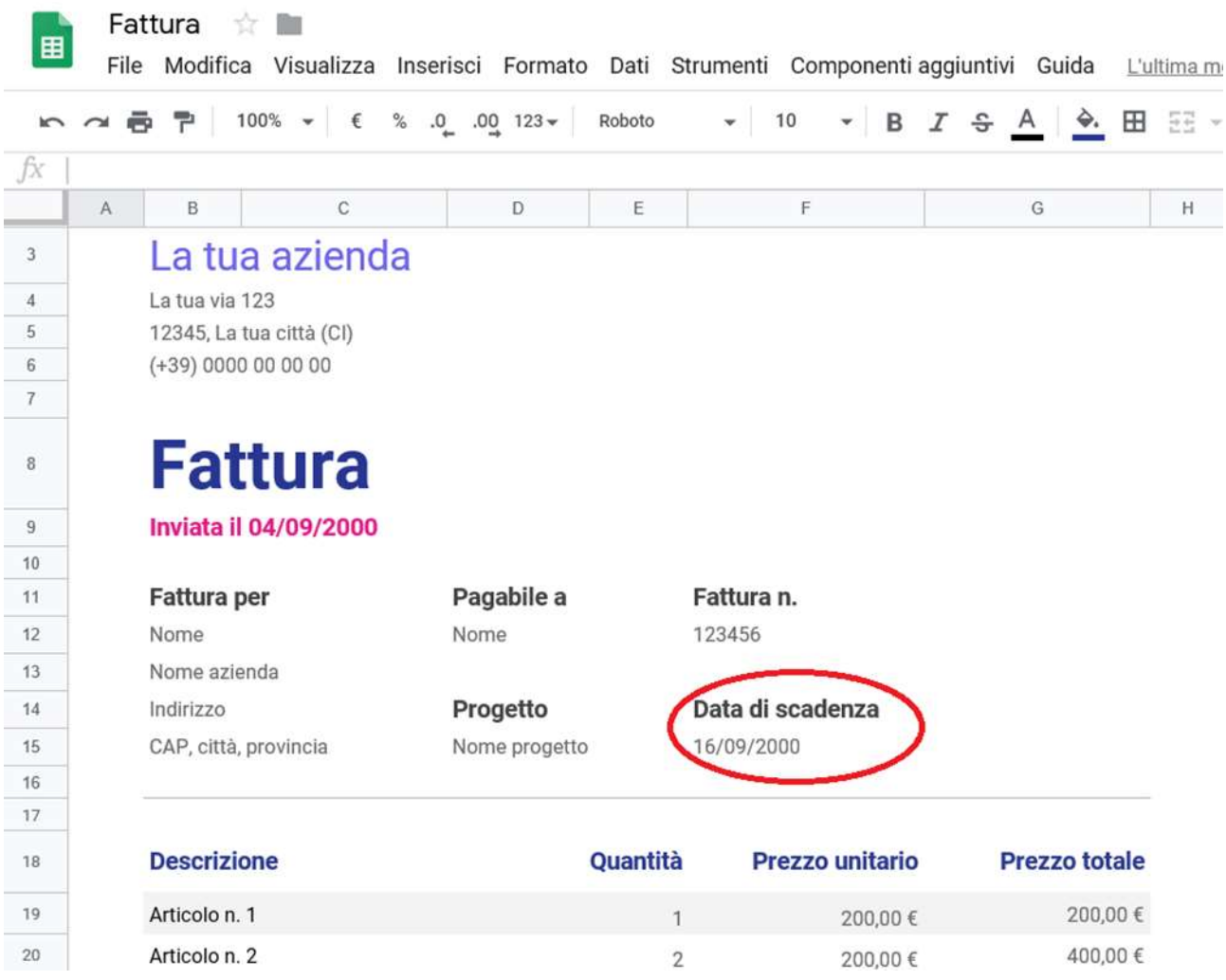
***Con approccio più orientato a focalizzare la composizione di una fattura*:**

prendendo come esempio il modello disponibile nella galleria **Google Fogli – modello di Fattura**



una Fattura **ha (è composta da)**;

- **un'intestazione**
- **più voci di dettaglio**
- **una sezione *risultati* (valori calcolati) complessivi e di sintesi**



Suggerimento *… in una programmazione orientata agli oggetti:*

un **oggetto di tipo Fattura ha (è composta da)**

- **un'intestazione (oggetto di una classe Intestazione)**
- **più voci di dettaglio (array di oggetti di tipo VoceDiFattura)**
- **una sezione *risultati* (valori calcolati) complessivi e di sintesi (oggetto di una classe Fine)**

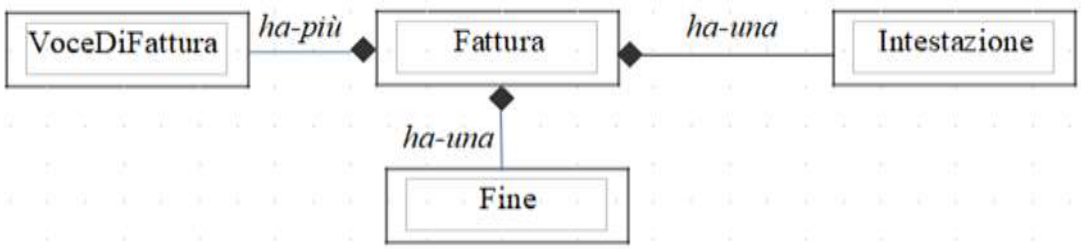
Note:

Subtotale **400,00 €**

Adeguamenti ***o IVA*** **....,00 €**

Importo ivato **500,00 €**

Illustrando con UML le relazioni tra classi:



NB: una relazione di tipo *contenimento*: è illustrata con un piccolo rombo pieno ◆ per indicare ***composizione***

Ulteriore suggerimento ……. gestire date in Java 8


```
import java.time.*;
import java.time.format.*;
```

```
General Output
--------------------Configuration: UsaData
Oggi: 2020-02-16
Scadenza a 120 gg: 2020-06-15


Data fattura: 16/02/2020
Scadenza a 120 gg: 15/06/2020

Process completed.
```

```
LocalDate date4 = LocalDate.now();
LocalDate date5 = date4.plusDays(120);

/// formattazione
DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");
String oggi = date4.format(formatter);
System.out.println("\n\nData fattura: " + oggi);                    // 16 febbraio 2020
String scadenza = date5.format(formatter);
System.out.println("Scadenza a 120 gg: " + scadenza);               //// circa 4 mesi
```