

# Bacheca Annunci

## Relazione progetto Paradigmi di Progettazione

### CLASSI

Le classi utilizzate dal progetto sono contenute nel package “src/model” e sono:

- **Annuncio**
- **AnnuncioAcquisto**
- **AnnuncioVendita**
- **Bacheca**
- **GestoreSalvataggi**
- **Utente**

Per la gestione dell’ input/output nella versione CLI è stata utilizzata la classe **Input.java** fornita durante il corso e si trova nella cartella principale **src** del progetto in “jbook/util”.

### TEST

Le classi di test sono contenute all’interno di “src/model/test”:

- **TestAnnuncioAcquisto**
- **TestAnnuncioVendita**
- **TestBacheca**
- **TestUtente**

### VERSIONE CLI

La versione CLI del progetto è gestita mediante la classe **BachecaCLI** contenuta all’interno di “src/cli” e richiamata nel main presente nella cartella principale **src** dalla classe **MainCLI**.

### GUI

L’interfaccia grafica del progetto è richiamata dal main contenuto nella cartella principale **src** dalla classe **MainGUI** e la gestione del tutto è contenuta in “src/gui” che si suddivide in:

“src/gui/controller”:

- **BachecaController**
- **LoginController**

“src/gui/vista”

- **BachecaGUI**
- **LoginFrame**

## CLASSI

### Annuncio, AnnuncioAcquisto ed AnnuncioVendita

Questa classe astratta serve come modello per la creazione dei due tipi di annunci, ovvero **vendita** e **acquisto**. All'interno ho definito le funzioni ed il costruttore che poi saranno implementati nelle classi principali **AnnuncioAcquisto** ed **AnnuncioVendita**. La differenza tra le due classi risiede nella "data di scadenza". Un annuncio d'acquisto non scade a differenza di quella di vendita.

I metodi presenti nella classe astratta sono i seguenti:

- getId()**: restituisce l'id dell'annuncio.
- getUtente()**: restituisce l'utente dell'annuncio.
- getNomeArticolo()**: restituisce il nome dell'annuncio.
- getPrezzo()**: restituisce il prezzo dell'annuncio.
- getParoleChiave()**: restituisce le parole chiavi.
- setNextId(int id)**: imposta l'id successivo.
- getNextId()**: restituisce l'id dell'annuncio.
- toString()**: restituisce la stringa visualizzabile con i dati dell'annuncio.
- isScaduto()**: metodo astratto che verifica se un annuncio è scaduto o no.

Il costruttore **Annuncio()** crea l'oggetto base dell'annuncio in cui ci sono i parametri: utente, nomeArticol, prezzo, paroleChiave.

Solo **AnnuncioVendita** differisce per un parametro in più, ovvero la *dataScadenza*.

Quest'ultimo sarà visualizzabile attraverso l'Override di **toString()**, oltre a scrivere il tipo di annuncio ("Vendita") e si può verificare la scadenza mediante **isScaduto()**.

In **AnnuncioAcquisto** invece mediante l'Override di **toString()** viene definito l'annuncio "Acquisto" ed impostato **isScaduto()** come false proprio perchè non viene messa una data, ma semplicemente l'annuncio non scade mai.

### Utente

Questa classe serve per la creazione dell'oggetto **Utente** che, all'interno del sistema, serve per identificare colui che aggiunge un annuncio d'acquisto o vendita. Ogni utente è identificato dalla email e da un nome utente.

I metodi presenti nella classe sono i seguenti:

- getMail()**: restituisce la mail dell'utente.
- getNome()**: restituisce il nome dell'utente.
- toString()**: restituisce la stringa visualizzabile con i dati dell'utente.

## Bacheca

All'interno di questa classe vengono gestiti i vari tipi di annunci presenti in bacheca. Oltre al costruttore **bacheca()**, nel quale viene creata la lista degli annunci, troviamo anche i seguenti metodi:

**-addAnnuncio(Annuncio annuncio):**

Aggiunge un annuncio alla lista presente in bacheca.

**-rimuoviAnnuncio(int id, Utente utente):**

Rimuove l'annuncio con un determinato id solamente se è dell'utente che vuole rimuoverlo.

**-cercaPerParoleChiave(Set<string> parole):**

Cerca gli annunci che abbiano le parole chiavi prese per parametro e restituisce i corrispondenti.

**-inserisciAnnuncioAcquisto(AnnuncioAcquisto a):**

Inserisce l'annuncio d'acquisto in bacheca e restituisce gli annunci che corrispondono alle parole chiave indicate.

**-pulisciBacheca():**

Rimuove tutti gli annunci scaduti dalla bacheca

## GestioneSalvataggi

All'interno di questa classe sono presente tutti i metodi utilizzati per la gestione dei salvataggi sia degli annunci che degli utenti.

Gli annunci saranno salvati all'interno di `"/salvataggi/annunci.txt"`, mentre gli utenti vengono salvati a parte all'interno di `"/salvataggi/utenti.txt"`.

I metodi presenti all'interno della classe sono i seguenti:

**-salvaBacheca(Bacheca bacheca):**

Mediante questo metodo salvo tutti gli annunci all'interno del file di testo.

**-caricaBacheca():**

Mediante questo metodo carico tutti i dati che sono stati salvati nel file di testo. Se non è presente il percorso ed il file allora li crea.

**-salvaUtenti(List<Utente> utenti):**

Metodo utilizzato per il salvataggio delle email e del nome utente.

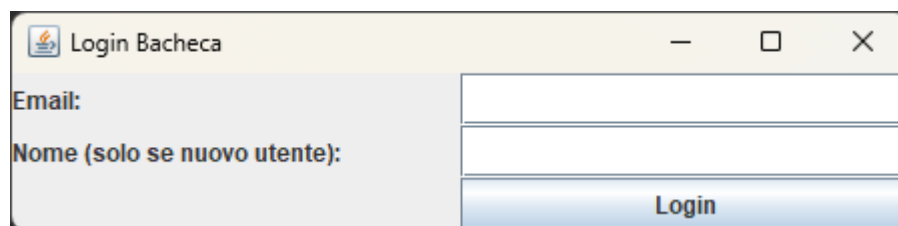
### -caricaUtenti():

Mediante questo metodo si caricano gli utenti presenti all'interno del sistema. Utile per effettuare la verifica dell'utente se già presente oppure bisogna registrarsi.

## GUI

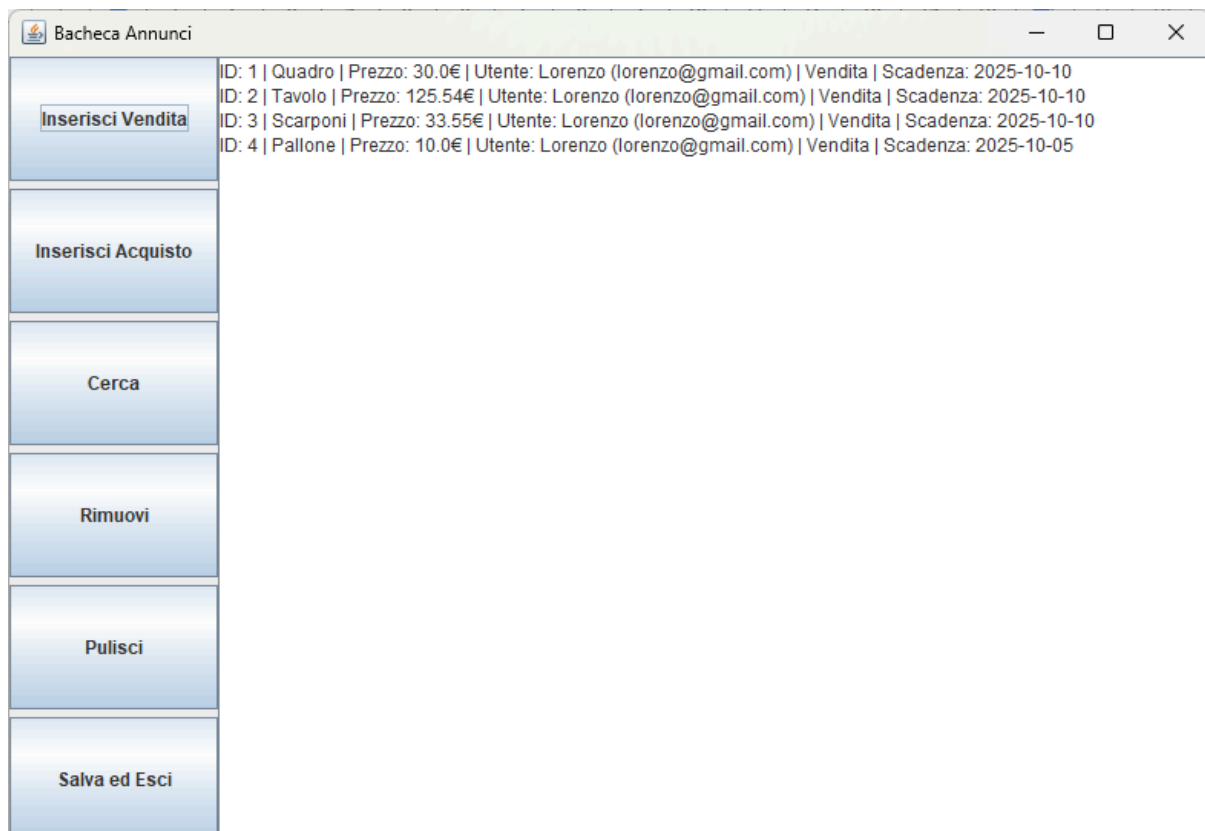
L'interfaccia grafica è divisa in due parti.

Durante il primo avvio comparirà un piccolo form di login che richiederà la mail dell'utente per fare l'accesso. Nel caso non fosse presente la mail, bisognerà inserire anche un nome utente e una volta premuto il tasto di login, verranno salvate in automatico le credenziali su un file di testo (/salvataggi/utenti.txt):



A screenshot of a Windows-style window titled "Login Bacheca". It contains two input fields: "Email:" and "Nome (solo se nuovo utente):". Below the "Nome" field is a "Login" button.

Una volta fatto l'accesso sarà possibile effettuare tutte le operazioni presenti nella sezione CLI e si vedrà in maniera interattiva la bacheca salvata nel file di testo (/salvataggi/annunci.txt):



A screenshot of a Windows-style window titled "Bacheca Annunci". On the left is a vertical sidebar with buttons: "Inserisci Vendita", "Inserisci Acquisto", "Cerca", "Rimuovi", "Pulisci", and "Salva ed Esci". The main area displays a list of items:

ID: 1		Quadro		Prezzo: 30.0€		Utente: Lorenzo (lorenzo@gmail.com)		Vendita		Scadenza: 2025-10-10
ID: 2		Tavolo		Prezzo: 125.54€		Utente: Lorenzo (lorenzo@gmail.com)		Vendita		Scadenza: 2025-10-10
ID: 3		Scarponi		Prezzo: 33.55€		Utente: Lorenzo (lorenzo@gmail.com)		Vendita		Scadenza: 2025-10-10
ID: 4		Pallone		Prezzo: 10.0€		Utente: Lorenzo (lorenzo@gmail.com)		Vendita		Scadenza: 2025-10-05

## Scelte effettuate e metodo di sviluppo

Il progetto è stato realizzato singolarmente utilizzando eclipse con la versione **java SE 22** ed è stato utilizzato GitHub per poter programmare con più facilità e maggiore flessibilità.

(Link progetto su GitHub: <https://github.com/LoreSanto/BachecaAnnunci.git> )

Nella prima parte di sviluppo mi sono concentrato nel creare gli oggetti principali, ovvero:

- **Utente**
- **Annuncio**, con relativo AnnuncioAcquisto e AnnuncioVendita

Dopodiché sono passato alla realizzazione dell'oggetto **Bacheca** in cui sarebbero state racchiuse tutte le funzionalità della bacheca, come l'aggiunta di un determinato tipo di annuncio, la rimozione, la pulizia ecc ...

Una volta realizzata la base e creato i test per verificare che tutto funzionasse correttamente, sono passato al creare la versione CLI del progetto ed a realizzare solo in seguito la classe che avrebbe gestito i salvataggi.

Per quanto riguarda quest'ultima è stata realizzata solo in seguito per via di alcune problematiche che ho riscontrato nella gestione dei salvataggi degli annunci, a differenza degli utenti che ha richiesto meno difficoltà.

La soluzione che ho adottato, nel salvataggio degli annunci, è stata quella di creare una variabile statica, presente ad inizio file di testo, atta alla memorizzazione del prossimo ID da utilizzare. In questa maniera ogni singolo annuncio ha un proprio ID statico diverso l'uno dall'altro e ad ogni apertura della bacheca, il sistema utilizzerà proprio questa variabile come base per la creazione degli ID di ogni singolo annuncio.

Per il caricamento invece, viene utilizzato un costruttore dell'annuncio in cui viene inserito l'ID direttamente tra i parametri, essendo già presente nel file di testo dedicato al salvataggio e non c'è quindi bisogno di assegnarne uno.

Tutto questo mi ha portato quindi alla realizzazione di due file distinti per i salvataggi, uno utilizzato per il salvataggio degli utenti ed uno utilizzato per il salvataggio della variabile statica **NEXT\_ID** con i relativi annunci presenti.

Inoltre, per semplificare e togliere eventuali problemi, la classe **GestoreSalvataggi** fa sempre una verifica di presenza della cartella **salvataggi**. In caso non ci sia viene direttamente creata insieme al relativo file di testo da utilizzare per il salvataggio degli utenti e della bacheca.