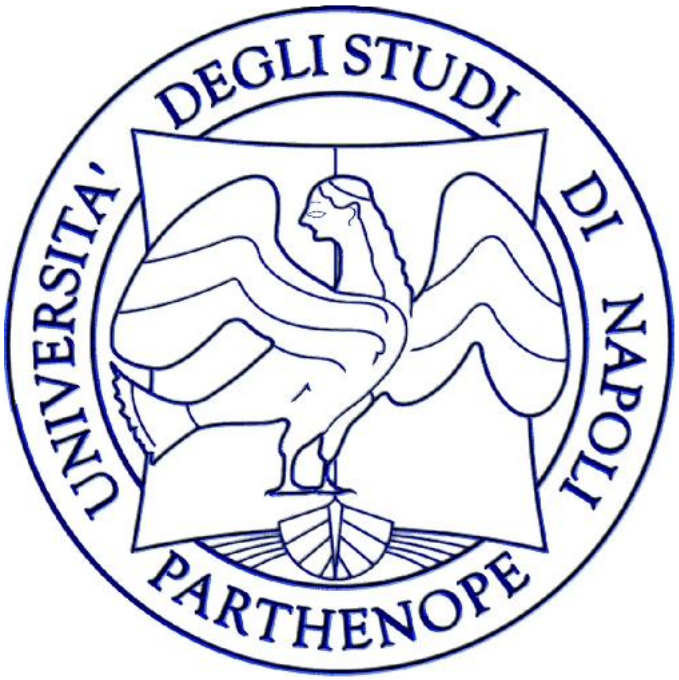
Progetto Programmazione 3



Gestionale - SistemaBancario

Docente
Prof. Angelo Ciaramella

Candidato
Pasquale De Trino
Matr. 012400/1637

Anno Accademico 2018-2019

Indice

1 Descrizione **1**

2 Implementazione **2**

2.1 Web Aplication 2

2.2 Descrizione UML 3

2.2.1 Factory Pattern 3

2.2.2 Chain of Responsibility 3

2.2.3 Template Method 4

2.2.4 Memento 4

2.2.5 Strategy 5

2.2.6 Decorator 6

Capitolo 1

Descrizione

Si richiede di simulare un SistemaBancario per gestire conti correnti e investimenti.
Il sistema prevede una doppia modalità di accesso, amministratore e utente. In modalità amministratore è possibile, dopo una fase di autenticazione, ricercare un utente tramite id o nome e cognome usando una struttura dati opportuna; Registrare un utente inserendo dati anagrafici, tipologia di conto corrente saldo della carta prepagata e del conto corrente. Automaticamente il sistema proprorrà agli utenti una forma di investimento considerando le caratteristiche dell'utente.

In modalità utente è possibile visualizzare le informazioni sulle operazioni fatte riguardo il proprio conto, fare acquisti tramite una carta, richiedere l'annullamento dell'ultimo addebito su conto corrente, richiedere uno sconto sugli acquisti tramite un codice sconto, partecipare ad un investimento scegliendo tra quelli disponibili o quello proposto dal sistema.

Capitolo 2

Implementazione

Il progetto è sviluppato in modalità Web Application in modo che il software possa risiedere su un server ed essere raggiungibile ed utilizzabile da diversi utenti tramite un comune browser web.
È inoltre possibile in questo modo aggiornare l'applicazione solo sul server e offrire sempre un prodotto aggiornato agli utilizzatori.
La web Application può anche essere estesa, è infatti possibile aggiungere funzionalità anche abbastanza complesse poichè, girando su un server, la potenza dell'hardware è tale da soddisfare i requisiti tecnici.

2.1 Web Aplication

I principali vantaggi di una web application riguardano le possibilità di utilizzare l'applicazione senza problemi di compatibilità di versione di Java oppure di dipendenze da software esterni quali ad esempio DBMS o particolari librerie grafiche.

2.2 Web Server

Il Web Server si occupa di predisporre e restituire le pagine web al client. L'Applicaton Server si occupa di gestire la logica applicativa e interfacciarsi con altri moduli.
Talvolta i ruoli sono fusi in un unico oggetto, definito Web Container, Tomcat ne è un esempio.

2.2.1 Servlets

Le servlets sono classi Java che consentono l'interazione richiesta/risposta secondo il modello client/server utilizzando i metodi doGet e doPost.
Il pattern MVC (Model View Controller), consente di separare la logica di presentazione da quella di business.
Utilizzando le servlets è possibile congiungere le due logiche scambiando informazioni utilizzando i metodi estesi dalla classe HttpServlet.

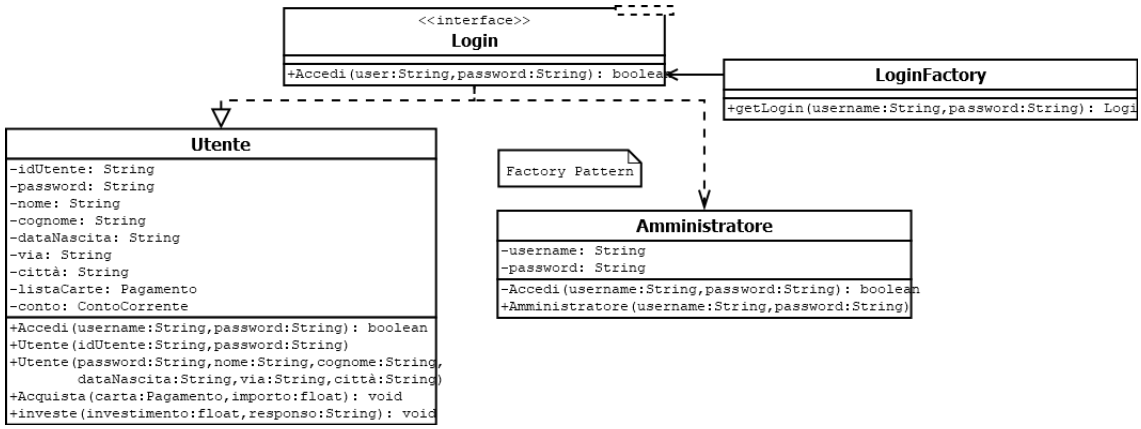
Capitolo 3

Descrizione UML

Seguendo il diagramma UML delle classi spieghiamo come è stato sviluppato il software.

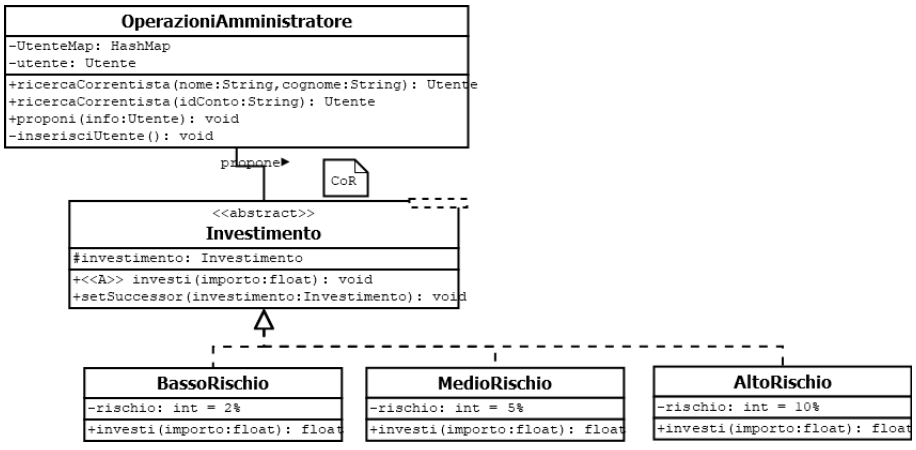
3.1 Factory Pattern

L'accesso può avvenire in modalità utente o in modalità amministratore, a tal proposito è stato utilizzato un **factory pattern**, il metodo Accedi della classe Login è implementato dalle classi Utente e Amministratore per controllare le credenziali di accesso in due differenti tabelle.



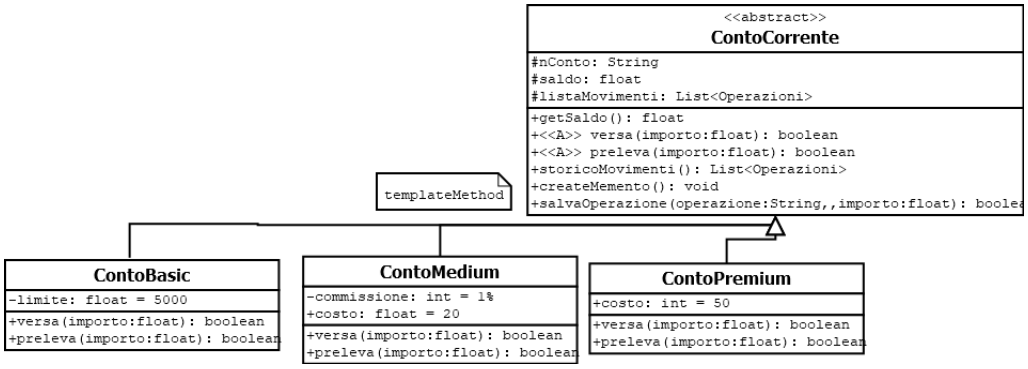
3.2 Chain of Responsibility

Proseguendo la lettura del diagramma in direzione dell'Amministratore, il pattern **CoR** definisce una classe astratta Investimento e le classi concrete BassoRischio, MedioRischio e AltoRischio che applicano una diversa logica al metodo investi.
La conclusione del pattern è nella servlet EffettuaInvestimento che data una stringa da alla prima classe della catena la richiesta che seguirà poi fino a trovare chi può soddisfare la richiesta.



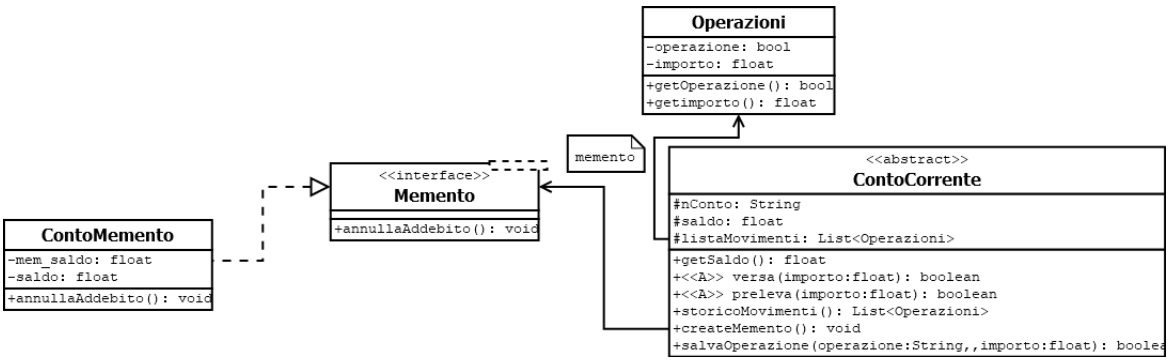
3.3 Template Method

Passando alla parte dello schema riguardante l' utente, il primo pattern è **TemplateMethod** il quale è stato pensato per astrarre il comportamento dei metodi versa e preleva che sono implementati dalle classi ContoBasic, ContoMedium, ContoPremium fornendo l'implementazione dei restanti metodi che risulteranno comuni per i vari tipi di conto corrente.



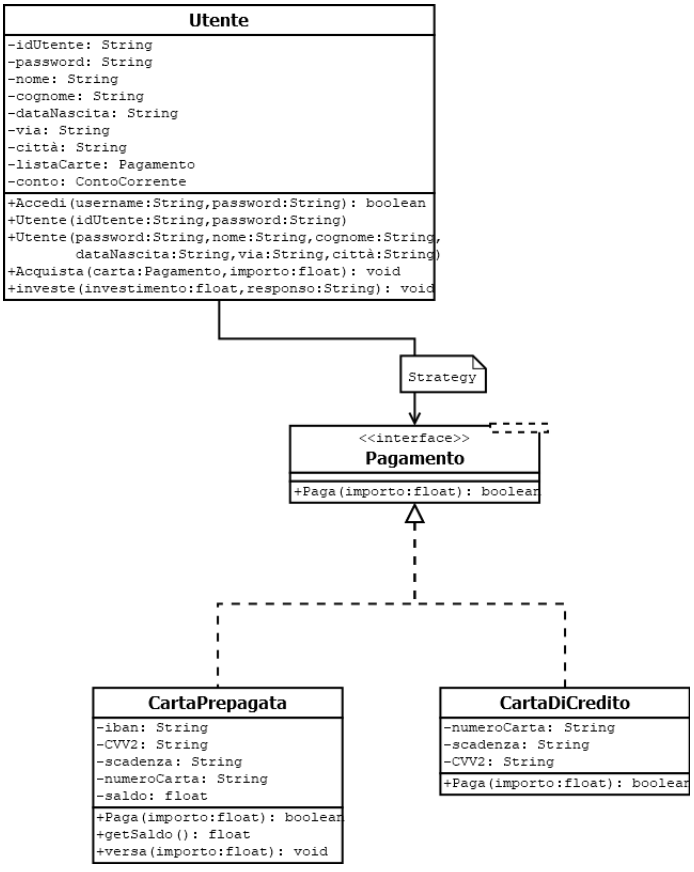
3.4 Memento

Il pattern **Memento** permette all'utente di poter annullare l'ultimo acquisto effettuato tramite carta di credito.



3.5 Strategy

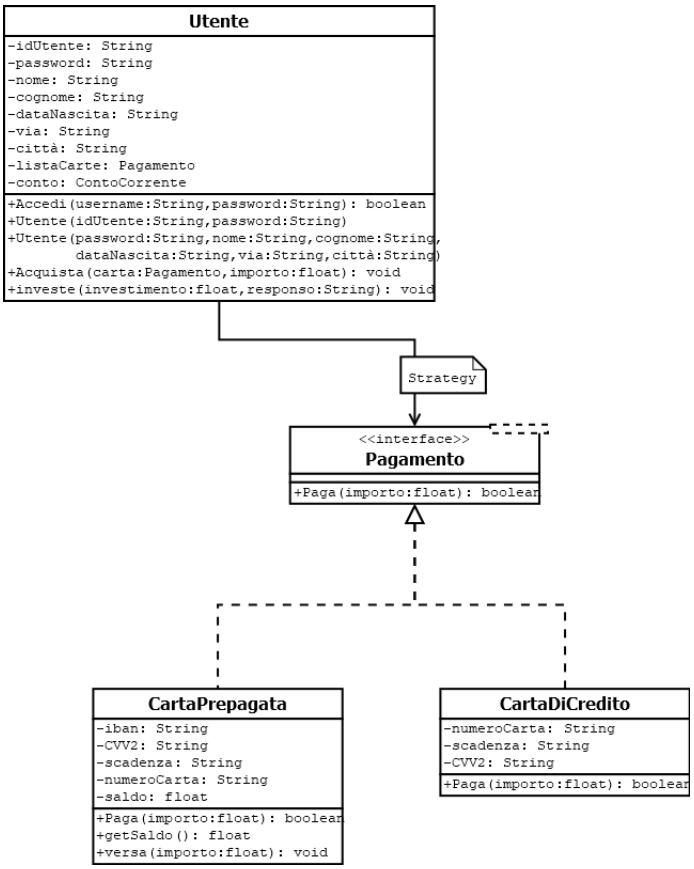
L' utente può effettuare un acquisto con carta di credito o con carta prepagata, il pattern **strategy** consente di implementare due strategie di pagamento differenti, nel primo caso viene applicata la logica di pagamento relativa al tipo di conto corrente posseduto, nel secondo controllando il saldo della carta prepagata.



3.6 Decorator

La possibilità di aggiungere uno sconto a run-time al proprio conto corrente è realizzata mediante il pattern **decorator**. In base al codice sconto iserito, l'utente riceve un diverso tipo di codice sconto, viene dunque decora-

to l'oggetto relativo al proprio conto corrente.



Il pattern **singleton** è usato in diversi contesti per mantenere un unica istanza dell' oggetto contoCorrente, in modo che essa possa essere eventualmente decorata.