Esercitazione sui design patterns

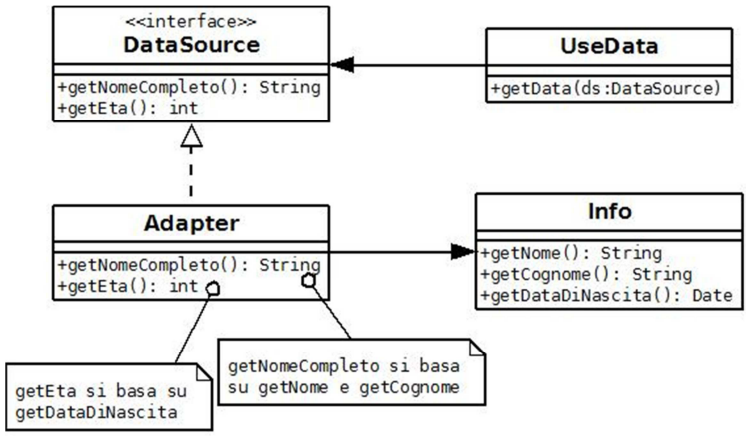
Esercizio 1

Scrivere il codice che implementa il pattern Adapter in una applicazione che deve usare le due classi seguenti. È fornita anche l'interfaccia usata dalla prima classe.

```
public interface DataSource {
        public String getNomeCompleto();
        public int getEta();
}

public class UseData {
  private String nomeCompleto;
  private int eta;
  public void getData(DataSource ds) {
    nomeCompleto = ds. getNomeCompleto ();
    eta = ds.getEta();
  }
}

public class Info {
  private String nome;
  private String cognome;
  private Date dataDiNascita;
  public String getNome() { return nome;}
  public void setNome(String nome) { this.nome = nome;}
  public String getCognome() { return cognome;}
  public void setCognome(String cognome) { this. cognome = cognome;}
  public Date getDataDiNascita() { return dataDiNascita;}
  public void DataDiNascita (Date dataDiNascita) { this. dataDiNascita = dataDiNascita;}
}
```



In particolare, si noti che:

- UseData chiama getNomeCompleto mentre Info mette a disposizione getNome e getCognome;
- UseData chiama getEta mentre Info mette a disposizione getDataDiNascita;
- Info non implementa l'interfaccia DataSource.

Definire quindi una classe Adapter che implementi DataSource e possa essere utilizzata da UseData, ma allo stesso tempo recuperi le informazioni da un oggetto di classe Info.

Esercizio 2

Data l'applicazione descritta in seguito, identificare:

1. Quali possono essere i problemi o le situazioni comuni ad altre applicazioni;
2. Quali pattern posso esprimere delle soluzioni per i problemi identificati in precedenza.

Applicazione

L'applicazione da sviluppare è costituita da una parte client e una parte server.

La parte client interagisce con l'utente e gli permette di formulare delle richieste da inviare al server. Il client deve poter essere eseguito su diverse piattaforme, indipendentemente dalla modalità di creazione di un canale di comunicazione con il server.

La parte server gestisce diversi tipi di informazioni che vengono messe a disposizione dei client. Ogni accesso al server deve essere autenticato. Il server può gestire richieste semplici, che vengono eseguite da un singolo componente, o richieste complesse, che richiedono l'interazione di più componenti.