SAPIENZA Università di Roma

Facoltà di Ingegneria dell'Informazione, Informatica e Statistica

Esercitazioni di

PROGETTAZIONE DEL SOFTWARE

(Corso di Laurea in Ingegneria Informatica ed Automatica

Corso di Laurea in Ingegneria dei Sistemi Informatici)

a.a. 2010-11

Estratto del compito d'esame del 13 settembre 2007

SOLUZIONE

Requisiti

L'applicazione da progettare riguarda una parte di un sistema di gestione di informazioni su aziende. Ogni azienda è caratterizzata da un nome (una stringa), da una descrizione testuale (una stringa) e dalle sedi in cui essa è presente con nome della località (una stringa) e indirizzo (una stringa). Le aziende sono suddivise in aziende pubbliche e aziende private. Delle prime interessa l'ente che le gestisce, con codice (una stringa) e nome (una stringa). Delle seconde, invece, interessa il capitale sociale (un reale). Di ogni azienda (sia essa pubblica o privata) interessa inoltre l'eventuale azienda controllante.

Requisiti (cont.)

Il fruitore dell'applicazione è interessato ad effettuare diversi controlli, in particolare:

- data una azienda a, verificare se essa è controllata direttamente o indirettamente (cioè attraverso altre aziende) da se stessa.

- data una azienda a, restituire l'insieme delle aziende private che essa controlla direttamente.

Requisiti (cont.)

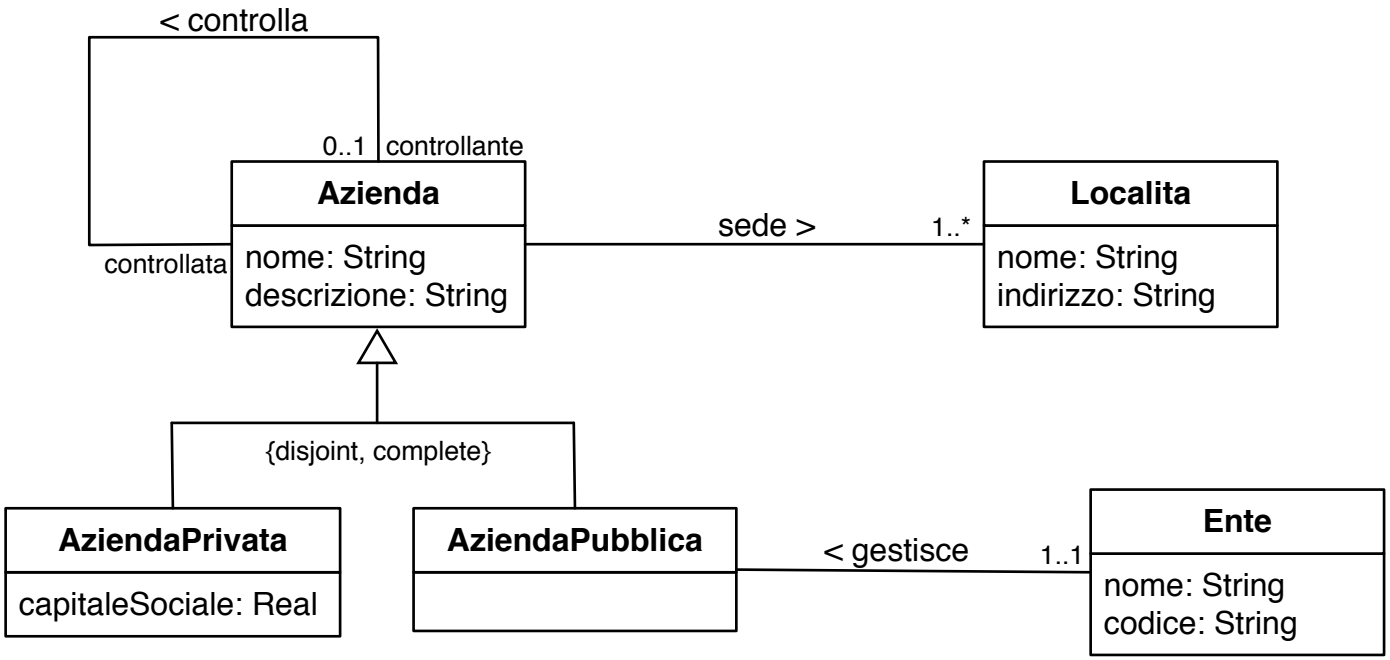
Domanda 1. Basandosi sui requisiti riportati sopra, effettuare la fase di analisi producendo lo schema concettuale in UML per l'applicazione e motivando, qualora ce ne fosse bisogno, le scelte effettuate.

Domanda 2. Effettuare la fase di progetto, illustrando i prodotti rilevanti di tale fase e motivando, qualora ce ne fosse bisogno, le scelte effettuate.

Domanda 3. Effettuare la fase di realizzazione, producendo un programma Java e motivando, qualora ce ne fosse bisogno, le scelte effettuate.

Fase di analisi

Diagramma delle classi



Fase di progetto

Responsabilità sulle associazioni

La seguente tabella delle responsabilità si evince da:

- 1. i requisiti,
- 2. le operazioni,
- 3. i vincoli di molteplicità nel diagramma delle classi.

Associazione	Classe	ha resp.
controlla	*controllata*	Sì[2,3]
	controllante	Sì[2]
sede	*Azienda*	Sì[3]
	Localita	NO
gestisce	*AziendaPubblica*	Sì[3]
	Ente	NO

Strutture di dati

Abbiamo la necessità di rappresentare collezioni omogenee di oggetti, a causa:

- dei vincoli di molteplicità 0..* delle associazioni,
- delle variabili necessarie per vari algoritmi.

Per fare ciò, utilizzeremo le classi del Java Collection Framework: `Set`, `HashSet`.

Corrispondenza fra tipi UML e Java

Riassumiamo le nostre scelte nella seguente tabella di corrispondenza dei tipi UML.

Tipo UML	Rappresentazione in Java
Real	`float`
String	`String`
boolean	`boolean`
Insieme	`HashSet`

Tabelle di gestione delle proprietà di classi UML

Riassumiamo le nostre scelte differenti da quelle di default mediante la *tabella delle proprietà immutabili* e la *tabella delle assunzioni sulla nascita*.

Classe UML	**Proprietà immutabile**
Azienda	*nome*
	descrizione
Localita	*nome*
	indirizzo
Ente	*nome*
	codice

Classe UML	**Proprietà**	
	nota alla nascita	**non nota alla nascita**
AziendaPubblica	–	*ente*

Altre considerazioni

Sequenza di nascita degli oggetti: Non dobbiamo assumere una particolare sequenza di nascita degli oggetti.

Valori alla nascita: Non sembra ragionevole assumere che per qualche proprietà esistano valori di default validi per tutti gli oggetti.

Fase di realizzazione

Struttura dei file e dei package

```
+---AppAzienda
    |   TipoLinkControlla.java
    |   ManagerControlla.java
    |   EccezioneMolteplicita.java
    |   EccezionePrecondizioni.java
    |
    +---Azienda
    |       Azienda.java
    |
    +---AziendaPubblica
    |       AziendaPubblica.java
    |
    +---AziendaPrivata
    |       AziendaPrivata.java
    |
    +---Localita
    |       Localita.java
    |
    \---Ente
            Ente.java
```

La classe Java Azienda

```
// File AppAzienda/Azienda/Azienda.java
package AppAzienda.Azienda;

import AppAzienda.*;
import AppAzienda.Localita.*;
import java.util.*;

public abstract class Azienda {

  protected final String nome;
  protected final String descrizione;

  protected HashSet<Localita> sede;
  private final int MOLT_MIN_SEDE = 1;
  protected HashSet<TipoLinkControlla> controllata; // le aziende controllate da
                                                    // this
  protected TipoLinkControlla controllante; // l'azienda controllante (l'azienda
                                            // che controlla this)

  protected Azienda(String nome, String descrizione) {
    this.nome = nome;
    this.descrizione = descrizione;
    sede = new HashSet<Localita>();
    controllata = new HashSet<TipoLinkControlla>();
```

```
  controllante = null;
}

public String getNome() {
  return nome;
}

public String getDescrizione() {
  return descrizione;
}

public void inserisciLinkSede(Localita l) {
  if (l != null)
    sede.add(l);
}

public void eliminaLinkSede(Localita l) {
  if (l != null)
    sede.remove(l);
}

public Set<Localita> getLinkSede() throws EccezioneMolteplicita {
  if (sede.size() < MOLT_MIN_SEDE)
    throw new EccezioneMolteplicita("Molteplicita' minima violata");
  return (HashSet<Localita>) sede.clone();
}
```

```
public void inserisciLinkControllante(TipoLinkControlla c) {
  if (c != null && c.getControllata() == this) {
    ManagerControlla.inserisci(c);
  }
}

public void eliminaLinkControllante(TipoLinkControlla c) {
  if (c != null && c.getControllata() == this) {
    ManagerControlla.elimina(c);
  }
}

public TipoLinkControlla getLinkControllante() {
  return controllante;
}

public void inserisciLinkControllata(TipoLinkControlla c) {
  if (c != null && c.getControllante() == this) {
    ManagerControlla.inserisci(c);
  }
}

public void eliminaLinkControllata(TipoLinkControlla c) {
  if (c != null && c.getControllante() == this) {
    ManagerControlla.elimina(c);
```

```
  }
}

public Set<TipoLinkControlla> getLinkControllata() {
  return (HashSet<TipoLinkControlla>) controllata.clone();
}

public void inserisciControllantePerManagerControlla(ManagerControlla a) {
  if (a != null) {
    controllante = a.getLink();
  }
}

public void eliminaControllantePerManagerControlla(ManagerControlla a) {
  if (a != null) {
    controllante = null;
  }
}

public void inserisciControllataPerManagerControlla(ManagerControlla a) {
  if (a != null) {
    controllata.add(a.getLink());
  }
}

public void eliminaControllataPerManagerControlla(ManagerControlla a) {
```

```
    if (a != null) {
      controllata.remove(a.getLink());
    }
  }
}
```

La classe Java AziendaPrivata

```
// File AppAzienda/AziendaPrivata/AziendaPrivata.java
package AppAzienda.AziendaPrivata;

import AppAzienda.Azienda.*;

public class AziendaPrivata extends Azienda {
  private float capitaleSociale;

  public AziendaPrivata(String nome, String descrizione, float capitaleSociale) {
    super(nome, descrizione);
    this.capitaleSociale = capitaleSociale;
  }

  public float getCapitaleSociale() {
    return capitaleSociale;
  }

  public void setCapitaleSociale(float capitaleSociale) {
    this.capitaleSociale = capitaleSociale;
  }
}
```

La classe Java AziendaPubblica

```
// File AppAzienda/AziendaPubblica/AziendaPubblica.java
package AppAzienda.AziendaPubblica;

import AppAzienda.*;
import AppAzienda.Azienda.*;
import AppAzienda.Ente.*;

public class AziendaPubblica extends Azienda {
  private Ente gestore;

  public AziendaPubblica(String nome, String descrizione) {
    super(nome, descrizione);
  }

  public void setGestore(Ente e) {
    if (e != null)
      gestore = e;
  }

  public Ente getGestore() throws EccezioneMolteplicita {
    if (gestore == null)
      throw new EccezioneMolteplicita("Molteplicita min/max violata");
    return gestore;
  }
}
```

La classe Java Localita

```
// File AppAzienda/Localita/Localita.java
package AppAzienda.Localita;

public class Localita {
  private final String nome;
  private final String indirizzo;

  public Localita(String nome, String indirizzo) {
    this.nome = nome;
    this.indirizzo = indirizzo;
  }

  public String getNome() {
    return nome;
  }

  public String getIndirizzo() {
    return indirizzo;
  }
}
```

La classe Java Ente

```
// File AppAzienda/Ente/Ente.java
package AppAzienda.Ente;

public class Ente {
  private final String nome;
  private final String codice;

  public Ente(String nome, String codice) {
    this.nome = nome;
    this.codice = codice;
  }

  public String getNome() {
    return nome;
  }

  public String getCodice() {
    return codice;
  }
}
```

La classe Java TipoLinkControlla

```
// File AppAzienda/TipoLinkControlla.java
package AppAzienda;

import AppAzienda.Azienda.*;
import java.util.*;

public class TipoLinkControlla {
  private final Azienda controllante;
  private final Azienda controllata;

  public TipoLinkControlla(Azienda controllante, Azienda controllata)
      throws EccezionePrecondizioni {
    if (controllante == null || controllata == null) // CONTROLLO PRECONDIZIONI
      throw new EccezionePrecondizioni(
          "Gli oggetti devono essere inizializzati");
    this.controllante = controllante;
    this.controllata = controllata;
  }

  public boolean equals(Object o) {
    if (o != null && getClass().equals(o.getClass())) {
      TipoLinkControlla l = (TipoLinkControlla) o;
      return l.controllante == controllante && l.controllata == controllata;
    } else
```

```
      return false;
    }

    public int hashCode() {
      return controllante.hashCode() + controllata.hashCode();
    }

    public Azienda getControllante() {
      return controllante;
    }

    public Azienda getControllata() {
      return controllata;
    }

    public String toString() {
      return "<" + controllante + ", " + controllata + ">";
    }
}
```

La classe Java ManagerControlla

```
// File AppAzienda/ManagerControlla.java
package AppAzienda;

public final class ManagerControlla {
  private TipoLinkControlla link;

  private ManagerControlla(TipoLinkControlla link) {
    this.link = link;
  }

  public TipoLinkControlla getLink() {
    return link;
  }

  public static void inserisci(TipoLinkControlla y) {
    if (y != null && y.getControllata().getLinkControllante() == null) {
      ManagerControlla k = new ManagerControlla(y);
      y.getControllante().inserisciControllataPerManagerControlla(k);
      y.getControllata().inserisciControllantePerManagerControlla(k);
    }
  }

  public static void elimina(TipoLinkControlla y) {
    if (y != null && y.getControllata().getLinkControllante().equals(y)) {
```

```
      ManagerControlla k = new ManagerControlla(y);
      y.getControllante().eliminaControllataPerManagerControlla(k);
      y.getControllata().eliminaControllantePerManagerControlla(k);
    }
  }
}
```

Realizzazione in Java delle classi per eccezioni

```
// File AppAzienda/EccezioneMolteplicita.java
package AppAzienda;

public class EccezioneMolteplicita extends Exception {
  private String messaggio;

  public EccezioneMolteplicita(String m) {
    messaggio = m;
  }

  public String toString() {
    return messaggio;
  }
}

// File AppAzienda/EccezionePrecondizioni.java
package AppAzienda;

public class EccezionePrecondizioni extends RuntimeException {
  private String messaggio;

  public EccezionePrecondizioni(String m) {
    messaggio = m;
  }
```

```
  public EccezionePrecondizioni() {
    messaggio = "Si e' verificata una violazione delle precondizioni";
  }

  public String toString() {
    return messaggio;
  }
}
```