Università di Roma "La Sapienza"
Facoltà di Ingegneria

Corso di
"PROGETTAZIONE DEL SOFTWARE I"
(Corso di Laurea in Ingegneria Informatica)
Proff. Marco Cadoli e Giuseppe De Giacomo
Canali A-L & M-Z
A.A. 2005-06

Compito d'esame del 20 aprile 2006

SOLUZIONE

Requisiti

L'applicazione da progettare riguarda la gestione di prenotazioni di una catena di agenzie di viaggi. Un'agenzia è caratterizzata da un codice (una stringa). Un cliente è caratterizzato da nome, cognome e data di nascita. Una destinazione è caratterizzata dal nome. Per ogni agenzia interessa conoscere le prenotazioni in atto: una prenotazione viene stipulata fra un'agenzia e un cliente per una certa destinazione e prevede una data di partenza. L'unica restrizione sulle prenotazioni è che uno stesso cliente non può stipulare due prenotazioni diverse con la stessa agenzia per la stessa destinazione. Per ogni destinazione è di interesse conoscere le prenotazioni in atto. Le destinazioni a rischio sono particolari destinazioni per cui esiste almeno un'agenzia abilitata dal Ministero degli Esteri, di cui interessa un codice di rischio (una stringa). Delle agenzie abilitate interessa la data di scadenza dell'abilitazione. Un'agenzia non può essere abilitata per più di tre destinazioni a rischio.

Requisiti (cont.)

Un cliente inizialmente viene catalogato come nuovo. Se paga il dovuto (e continua a pagare) viene riconosciuto come affidabile, se è in ritardo nei pagamenti viene catalogato come moroso. Se è stato catalogato come cliente moroso anche solo una volta, anche qualora pagasse il dovuto (quindi non sarebbe più moroso) sarebbe sempre ritenuto sospetto.

La Camera di Commercio è interessata ad effettuare un'analisi di qualità dei servizi erogati, che si concretizza nei seguenti controlli:

- data un'agenzia abilitata, conoscere quanti clienti morosi hanno prenotazioni con essa.
- data un'agenzia abilitata, se ha effettuato prenotazioni per destinazioni a rischio senza avere l'autorizzazione per esse, oppure se ha effettuato prenotazioni per destinazioni a rischio la cui data di partenza è successiva alla data di scadenza dell'autorizzazione.

Fase di analisi

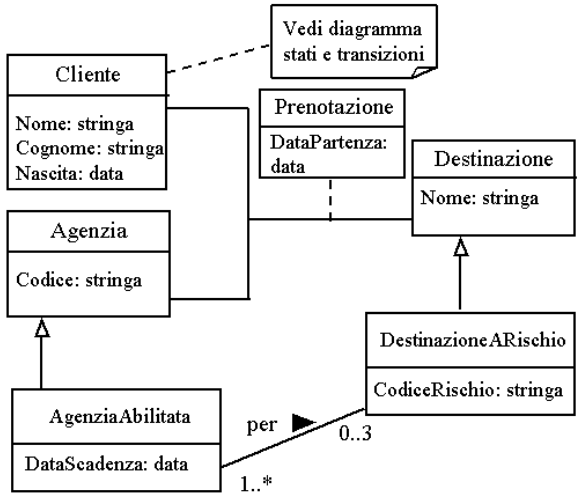


Diagramma degli stati e delle transizioni classe Cliente

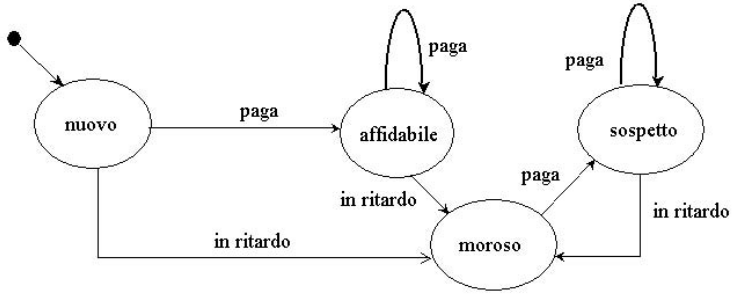
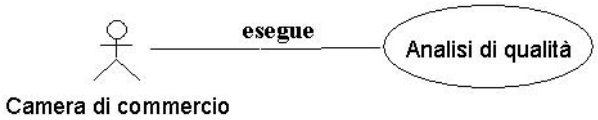


Diagramma degli use case



Specifica dello use case

InizioSpecificaUseCase AnalisiDiQualità

QuantiClientiMorosi *(a: AgenziaAbilitata): intero*
pre: *true*
post: definiamo Cli come l'insieme

$$\{c \mid c \in Cliente \wedge \exists d \in Destinazione \wedge \langle a, c, d\rangle \in Prenotazione \wedge c.EstMoroso = true\}.$$

result = card(Cli)

PrenotazioniInRegola *(a: AgenziaAbilitata): booleano*
pre: *true*
post: *result* = *true* se e solo se è vero che

$$\forall c, d \quad c \in Cliente \wedge d \in DestinazioneARischio \wedge \langle a, c, d\rangle \in Prenotazione \rightarrow (DataPartenza(a, c, d) \leq DataScadenza(a) \wedge \langle a, d\rangle \in Per)$$

FineSpecifica

Fase di progetto

Algoritmi per le operazioni dello use-case

Adottiamo i seguenti algoritmi:

- Per l'operazione **QuantiClientiMorosi**:

```
Insieme(Cliente) Cli = new Insieme(Cliente);  // restituisce un ins. vuoto
per ogni link l di tipo Prenotazione in cui a è coinvolto
        se (l.Cliente.EstMoroso = true)
            Cli = Cli Unione {l.Cliente}
return Cli.size();
```

- Per l'operazione **PrenotazioniInRegola**:

```
Insieme(TipoLinkPer) TLP = {l | l è un link di tipo Per in cui a è coinvolto}:
per ogni link l di tipo Prenotazione in cui a è coinvolto
        se (l.Destinazione.class = DestinazioneARischio &&
              ( (DataPartenza(a,l.Cliente,l.Destinazione > DataScadenza(a))) ||
                <a.l.Destinazione> non appartiene a TLP
              )
           )
        return false;
return true;
```


Responsabilità sulle associazioni

La seguente tabella delle responsabilità si evince da:

1. i requisiti,
2. la specifica degli algoritmi per le operazioni di classe e use-case,
3. i vincoli di molteplicità nel diagramma delle classi.

Associazione	Classe	ha resp.
prenotazione	*Cliente*	NO
	Agenzia	SÌ[1,2]
	Destinazione	SÌ[1]
per	*AgenziaAbilitata*	SÌ[2,3]
	DestinazioneARischio	SÌ[3]

Strutture di dati

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

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

Per fare ciò, utilizzeremo le classi del collection framework di Java 1.5: `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
intero	`int`
booleano	`boolean`
stringa	`String`
data	`Data`
Insieme	`HashSet`

Utilizzeremo la classe Java `Data` presentata durante le lezioni del corso.

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**
Cliente	*Nome*
	Cognome
	Nascita
Destinazione	*Nome*

Classe UML	**Proprietà**	
	nota alla nascita	**non nota alla nascita**
AgenziaAbilitata	–	*DataScadenza*

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.

Rappresentazione degli stati in Java

Per la classe UML *Cliente*, ci dobbiamo occupare della rappresentazione in Java del diagramma degli stati e delle transizioni.

Scegliamo di rappresentare gli stati mediante una variabile `int`, secondo la seguente tabella.

	Rappresentazione in Java	
	tipo var.	`int`
Stato	nome var.	`stato`
nuovo	valore	1
affidabile	valore	2
sospetto	valore	3
moroso	valore	3

API delle classi Java progettate

Omesse per brevità. (Si faccia riferimento al codice Java).

Fase di realizzazione

Considerazioni iniziali

La traccia ci richiede di realizzare:

1. la classe UML *AgenziaAbilitata*;
2. la classe UML *DestinazioneARischio*
3. l'associazione UML *per* con responsabilità doppia avente un attibuto e con vincoli di molteplicità 0..3 (molteplicità massima finita) e 1..* (molteplicità minima diversa da zero);
4. uno use case.

Nel seguito tuttavia verranno realizzate tutte le classi e gli use case individuati in fase di analisi.

Struttura dei file e dei package

```
+---AppVacanze
|   |   EccezioneCardMax.java
|   |   EccezioneCardMin.java
|   |   AnalisiQualita.java
|   |   EccezionePrecondizioni.java
|   |   AssociazionePer.java
|   |   TipoLinkPer.java
|   |   AssociazionePrenotazione.java
|   |   TipoLinkPrenotazione.java
|   |   Cliente.java
|   |
|   +---AgenziaAbilitata
|   |       AgenziaAbilitata.java
|   |
|   +---Agenzia
|   |       Agenzia.java
|   |
|   +---Destinazione
|   |       Destinazione.java
|   |
|   \---DestinazioneARischio
|           DestinazioneARischio.java
|
\---data
        Data.java
```

La classe Java Cliente

```
// File AppVacanze/Cliente.java
package AppVacanze;
import AppVacanze.Agenzia.*;
import data.*;
import java.util.*;

public class Cliente {
    private final String nome, cognome;
    private final Data nascita;
    private static final int NUOVO = 1,
        AFFIDABILE = 2, SOSPETTO = 3, MOROSO = 4;
    private int stato_corrente;
    public Cliente(String no, String c, Data na) {
        nome = no;
        cognome = c;
        nascita = na;
        stato_corrente = NUOVO;
    }
    public String getNome() { return nome; }
    public String getCognome() { return cognome; }
    public Data getNascita() { return nascita; }
    public boolean estMoroso() {
        return stato_corrente == MOROSO;
    }
```



```
    public void paga() {
        if (stato_corrente == NUOVO)
            stato_corrente = AFFIDABILE;
        else if (stato_corrente == MOROSO)
            stato_corrente = SOSPETTO;
    }
    public void inRitardo() {
        stato_corrente = MOROSO;
    }
    public String toString() {
        return nome + cognome;
    }
}
```

La classe Java Agenzia

```
// File AppVacanze/Agenzia/Agenzia.java
package AppVacanze.Agenzia;
import AppVacanze.*;
import java.util.*;

public class Agenzia {
    protected String codice;
    protected HashSet<TipoLinkPrenotazione> prenotazione;
    public Agenzia(String c) {
        codice = c;
        prenotazione = new HashSet<TipoLinkPrenotazione>();
    }
    public String getCodice() { return codice; }
    public void setCodice(String fa) { codice = fa; }
    public void inserisciLinkPrenotazione(AssociazionePrenotazione a) {
        if (a != null) prenotazione.add(a.getLink());
    }
    public void eliminaLinkPrenotazione(AssociazionePrenotazione a) {
        if (a != null) prenotazione.remove(a.getLink());
    }
    public Set<TipoLinkPrenotazione> getLinkPrenotazione() {
        return (HashSet<TipoLinkPrenotazione>)prenotazione.clone();
    }
}
```


La classe Java AgenziaAbilitata

```
// File AppVacanze/AgenziaAbilitata/AgenziaAbilitata.java
package AppVacanze.AgenziaAbilitata;
import AppVacanze.Agenzia.*;
import AppVacanze.*;
import data.*;
import java.util.*;

public class AgenziaAbilitata extends Agenzia {
    public static final int MAX_LINK_PER = 3;
    protected Data dataScadenza;
    protected HashSet<TipoLinkPer> insieme_link_per;
    public AgenziaAbilitata(Data d, String c) {
        super(c);
        dataScadenza = d;
        insieme_link_per = new HashSet<TipoLinkPer>();
    }
    public Data getDataScadenza() { return dataScadenza; }
    public void setDataScadenza(Data d) { dataScadenza = d; }
    public int quantiPer() {
        return insieme_link_per.size();
    }
    public void inserisciLinkPer(AssociazionePer a) {
        if (a != null) insieme_link_per.add(a.getLink());
    }
```

```
    public void eliminaLinkPer(AssociazionePer a) {
        if (a != null) insieme_link_per.remove(a.getLink());
    }
    public Set<TipoLinkPer> getLinkPer() throws EccezioneCardMax {
        if (quantiPer() > MAX_LINK_PER)
            throw new EccezioneCardMax("Cardinalita' massima violata");
        else return (HashSet<TipoLinkPer>)insieme_link_per.clone();
    }
}
```

La classe Java Destinazione

```
// File AppVacanze/Destinazione/Destinazione.java
package AppVacanze.Destinazione;
import AppVacanze.*;
import java.util.*;

public class Destinazione  {
    private String nome;
    protected HashSet<TipoLinkPrenotazione> prenotazione;
    public Destinazione(String n) {
        nome = n;
        prenotazione = new HashSet<TipoLinkPrenotazione>();
    }
    public String getNome() { return nome; }
    public void setNome(String n) { nome = n; }
    public void inserisciLinkPrenotazione(AssociazionePrenotazione a) {
        if (a != null) prenotazione.add(a.getLink());
    }
    public void eliminaLinkPrenotazione(AssociazionePrenotazione a) {
        if (a != null) prenotazione.remove(a.getLink());
    }
    public Set<TipoLinkPrenotazione> getLinkPrenotazione() {
        return (HashSet<TipoLinkPrenotazione>)prenotazione.clone();
    }
}
```


La classe Java DestinazioneARischio

```
// File AppVacanze/DestinazioneARischio/DestinazioneARischio.java
package AppVacanze.DestinazioneARischio;
import AppVacanze.Destinazione.*;
import AppVacanze.*;
import java.util.*;

public class DestinazioneARischio extends Destinazione {
    public static final int MIN_LINK_PER = 1;
    protected String codiceRischio;
    protected HashSet<TipoLinkPer> insieme_link;
    public DestinazioneARischio(String n) {
        super(n);
        insieme_link = new HashSet<TipoLinkPer>();
    }
    public void setCodiceRischio(String c) {
        codiceRischio = c;
    }
    public String getCodiceRischio() {
        return codiceRischio;
    }
    public int quantiPer() { return insieme_link.size(); }
    public void inserisciLinkPer(AssociazionePer a) {
        if (a != null) insieme_link.add(a.getLink());
    }
```



```
    public void eliminaLinkPer(AssociazionePer a) {
        if (a != null) insieme_link.remove(a.getLink());
    }
    public Set<TipoLinkPer> getLinkPer() throws EccezioneCardMin {
        if (quantiPer() < MIN_LINK_PER)
            throw new EccezioneCardMin("Cardinalita' minima violata");
        else return (HashSet<TipoLinkPer>)insieme_link.clone();
    }
}
```

La classe Java AssociazionePrenotazione

```
// File AppVacanze/AssociazionePrenotazione.java
package AppVacanze;

public final class AssociazionePrenotazione {
    private AssociazionePrenotazione(TipoLinkPrenotazione x) { link = x; }
    private TipoLinkPrenotazione link;
    public TipoLinkPrenotazione getLink() { return link; }
    public static void inserisci(TipoLinkPrenotazione y) {
        if (y != null) {
            AssociazionePrenotazione k = new AssociazionePrenotazione(y);
            k.link.getAgenzia().inserisciLinkPrenotazione(k);
        }
    }
    public static void elimina(TipoLinkPrenotazione y) {
        if (y != null) {
            AssociazionePrenotazione k = new AssociazionePrenotazione(y);
            k.link.getAgenzia().eliminaLinkPrenotazione(k);
        }
    }
}
```


La classe Java TipoLinkPrenotazione

```
// File AppVacanze/TipoLinkPrenotazione.java
package AppVacanze;
import data.*;
import AppVacanze.Destinazione.*;
import AppVacanze.Agenzia.*;

public class TipoLinkPrenotazione {
    private final Agenzia laAgenzia;
    private final Destinazione laDestinazione;
    private final Cliente ilCliente;
    private final Data dataPartenza;
    public TipoLinkPrenotazione(Agenzia a, Cliente c, Destinazione d,
                        Data dp)
        throws EccezionePrecondizioni {
        if (a == null || c == null || d == null)  // CONTROLLO PRECONDIZIONI
            throw new EccezionePrecondizioni
                ("Gli oggetti devono essere inizializzati");
        laAgenzia = a; ilCliente = c; laDestinazione = d;
        dataPartenza = dp;
    }
    public boolean equals(Object o) {
        if (o != null && getClass().equals(o.getClass())) {
            TipoLinkPrenotazione b = (TipoLinkPrenotazione)o;
            return b.ilCliente == ilCliente &&
```



```
                b.laAgenzia == laAgenzia &&
                b.laDestinazione == laDestinazione;
        }
        else return false;
    }
    public int hashCode() {
        return laAgenzia.hashCode() + ilCliente.hashCode() +
            laDestinazione.hashCode();
    }
    public Agenzia getAgenzia() { return laAgenzia; }
    public Destinazione getDestinazione() { return laDestinazione; }
    public Cliente getCliente() { return ilCliente; }
    public Data getDataPartenza() { return dataPartenza; }
    public String toString() {
        return "<" + laAgenzia + ", " + ilCliente + ", " + laDestinazione
            + ">";
    }
}
```

La classe Java AssociazionePer

```
// File AppVacanze/AssociazionePer.java
package AppVacanze;
import AppVacanze.AgenziaAbilitata.*;
import AppVacanze.DestinazioneARischio.*;

public final class AssociazionePer {
    private AssociazionePer(TipoLinkPer x) { link = x; }
    private TipoLinkPer link;
    public TipoLinkPer getLink() { return link; }
    public static void inserisci(TipoLinkPer y) {
        if (y != null) {
            AssociazionePer k = new AssociazionePer(y);
            k.link.getAgenziaAbilitata().inserisciLinkPer(k);
            k.link.getDestinazioneARischio().inserisciLinkPer(k);
        }
    }
    public static void elimina(TipoLinkPer y) {
        if (y != null) {
            AssociazionePer k = new AssociazionePer(y);
            k.link.getAgenziaAbilitata().eliminaLinkPer(k);
            k.link.getDestinazioneARischio().eliminaLinkPer(k);
        }
    }
}
```

La classe Java TipoLinkPer

```
// File AppVacanze/TipoLinkPer.java
package AppVacanze;
import AppVacanze.AgenziaAbilitata.*;
import AppVacanze.DestinazioneARischio.*;
import data.*;

public class TipoLinkPer {
    private final AgenziaAbilitata laAgenziaAbilitata;
    private final DestinazioneARischio laDestinazioneARischio;
    public TipoLinkPer(AgenziaAbilitata a, DestinazioneARischio d)
        throws EccezionePrecondizioni {
        if (a == null || d == null)
            throw new EccezionePrecondizioni
                ("Gli oggetti devono essere inizializzati");
        laAgenziaAbilitata = a; laDestinazioneARischio = d;
    }
    public boolean equals(Object o) {
        if (o != null && getClass().equals(o.getClass())) {
            TipoLinkPer b = (TipoLinkPer)o;
            return b.laAgenziaAbilitata == laAgenziaAbilitata &&
                b.laDestinazioneARischio == laDestinazioneARischio;
        }
        else return false;
    }
```



```
    public int hashCode() {
        return laDestinazioneARischio.hashCode() +
            laAgenziaAbilitata.hashCode();
    }
    public AgenziaAbilitata getAgenziaAbilitata() {
        return laAgenziaAbilitata; }
    public DestinazioneARischio getDestinazioneARischio() {
        return laDestinazioneARischio; }
    public String toString() {
        return "(<" + laAgenziaAbilitata + ", " +
            laDestinazioneARischio + ">)";
    }
}
```

Realizzazione in Java dello use case

```
// File AppVacanze/AnalisiQualita.java
package AppVacanze;
import java.util.*;
import AppVacanze.Agenzia.*;
import AppVacanze.AgenziaAbilitata.*;
import AppVacanze.Destinazione.*;
import AppVacanze.DestinazioneARischio.*;

public final class AnalisiQualita {
    private AnalisiQualita() { }
    public static int quantiClientiARischio(AgenziaAbilitata a) {
        HashSet<Cliente> cli = new HashSet<Cliente>();
        Set<TipoLinkPrenotazione> ins = a.getLinkPrenotazione();
        Iterator<TipoLinkPrenotazione> it = ins.iterator();
        while(it.hasNext()) {
            TipoLinkPrenotazione t = it.next();
            if (t.getCliente().estMoroso())
                cli.add(t.getCliente());
        }
        return cli.size();
    }
    public static boolean prenotazioniInRegola(AgenziaAbilitata a)
    throws EccezioneCardMax, EccezionePrecondizioni {
        Set<TipoLinkPrenotazione> insPre = a.getLinkPrenotazione();
```



```
        Set<TipoLinkPer> insPer = a.getLinkPer();
        Iterator<TipoLinkPrenotazione> it = insPre.iterator();
        while(it.hasNext()) {
            TipoLinkPrenotazione l = it.next();
            TipoLinkPer tper = null;
    tper =
new TipoLinkPer(a,(DestinazioneARischio)l.getDestinazione());
            if (
                l.getDestinazione().getClass() == DestinazioneARischio.class &&
                ( l.getDataPartenza().prima(a.getDataScadenza()) ||
                  (! insPer.contains(tper)))
                )
                return false;
        }
        return true;
    }
}
```

Realizzazione in Java delle classi per eccezioni

```
// File AppVacanze/EccezioneCardMin.java
package AppVacanze;

public class EccezioneCardMin extends Exception {
    private String messaggio;
    public EccezioneCardMin(String m) {
        messaggio = m;
    }
    public String toString() {
        return messaggio;
    }
}
```

```
// File AppVacanze/EccezioneCardMax.java
package AppVacanze;
public class EccezioneCardMax extends Exception {
    private String messaggio;
    public EccezioneCardMax(String m) {
        messaggio = m;
    }
    public String toString() {
        return messaggio;
    }
}
```



```
// File AppVacanze/EccezionePrecondizioni.java
package AppVacanze;

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;
  }
}
```