SAPIENZA Università di Roma

Facoltà di Ingegneria Informatica

Esercitazioni di

PROGETTAZIONE DEL SOFTWARE

(Corso di Laurea in Ingegneria Informatica)

A.A. 2009-10

Estratto del compito d'esame del 3 luglio 2007

SOLUZIONE

Requisiti

L'applicazione da progettare riguarda una parte dell'interfaccia ad icone di un telefono cellulare di nuova generazione. Ogni icona è caratterizzata da un codice (una stringa) e da una immagine (rappresentata anche essa da una stringa). Alcune icone sono *icone-attive* e sono caratterizzate da: *i.* un suono (rappresentato da una stringa) che viene prodotto al click su di esse, *ii.* dall'applicazione che viene attivata al click, *iii.* dalla sequenza (non vuota) di animazioni che vengono mostrate al click e *iv.* dalle *display-area* (una o più) che occupano, ciascuna con l'indicazione se essa è occupata interamente o meno. Le applicazioni sono caratterizzate dal loro nome e dal nome del file (una stringa) dove è memorizzato il codice eseguibile. Le animazioni sono caratterizzate da un link (una stringa) al codice di visualizzazione. Le display-area dalla posizione (un intero) e dall'immagine di background (una stringa). Un'animazione, a sua volta, può coinvolgere una o più display-area (anche non correlate con quelle occupate dall'icona-attiva che la mette in esecuzione).

Requisiti (cont.)

Il fruitore della applicazione è interessato ad effettuare diverse operazioni, in particolare:

- data una icona-attiva ia, restituire una lista contenente l'inverso della sequenza della animazioni che ia utilizza;

- data una animazione a restituire l'insieme delle icone-attive che mostrano a.

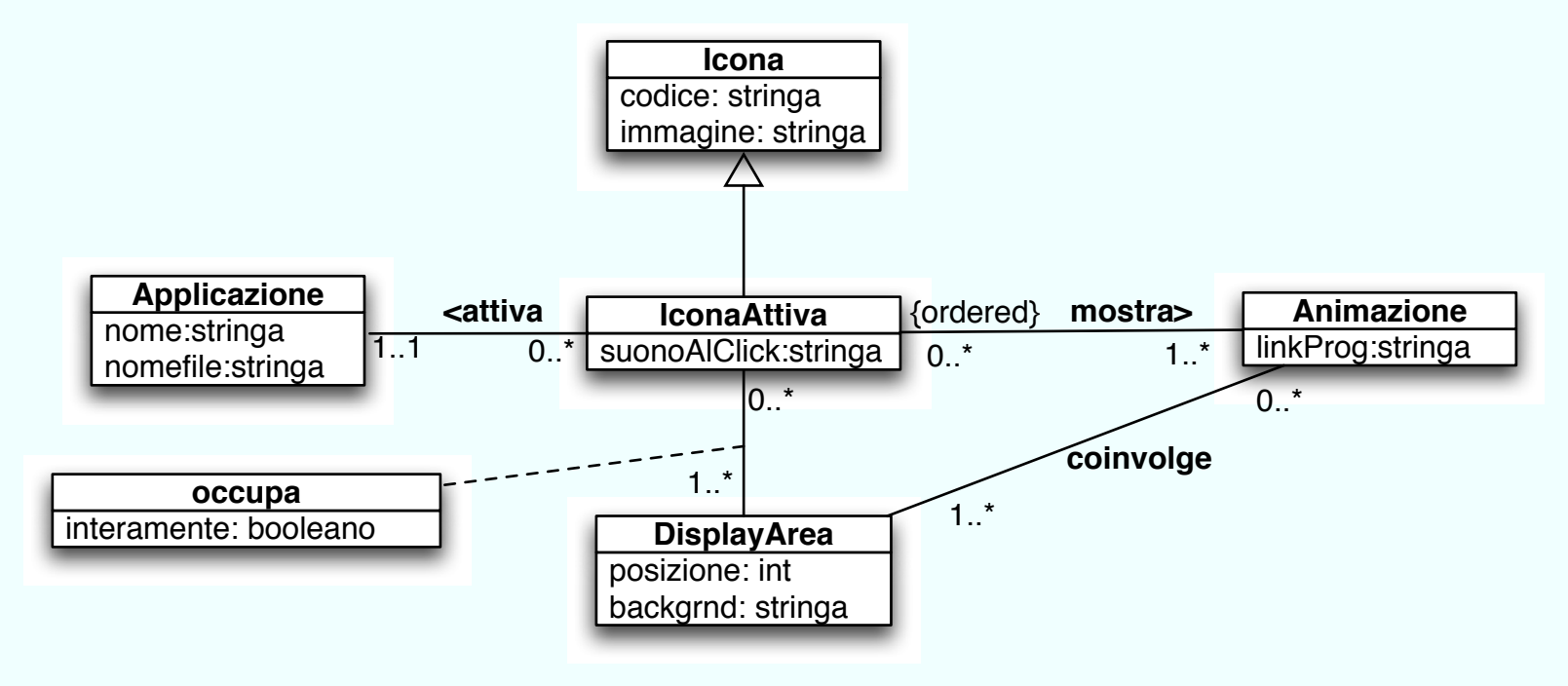
Requisiti (cont.)

È obbligatorio realizzare in Java solo i seguenti aspetti dello schema concettuale:

- la classi `IconaAttiva` e tutte le associazioni in cui è coinvolta;
- il primo use case.

Fase di analisi

Diagramma delle classi



Fase di progetto

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.
attiva	*IconaAttiva*	SÌ[3]
	Applicazione	NO
coinvolge	*Animazione*	SÌ[3]
	DisplayArea	NO
mostra	*IconaAttiva*	SÌ[2,3]
	Animazione	SÌ[2]
occupa	*IconaAttiva*	SÌ[3]
	DisplayArea	NO

Strutture di dati

Abbiamo la necessità di rappresentare collezioni e liste omogenee di oggetti, eventualmente ordinate, a causa:

- dei vincoli di molteplicità 0..* delle associazioni,
- della presenza di associazioni ordinate,

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

Corrispondenza fra tipi UML e Java

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

Tipo UML	Rappresentazione in Java
stringa	`String`
booleano	`boolean`
Insieme	`HashSet`
Lista	`LinkedList`

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**
Icona	*codice*
	immagine
IconaAttiva	*suonoAlClick*
Applicazione	*nome*
	nomeFile
Animazione	*linkProg*
DisplayArea	*posizione*
	backgrnd

	Proprietà	
Classe UML	**nota alla nascita**	**non nota alla nascita**
IconaAttiva	–	*applicazione*

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

```
+---AppCellulare
    |   TipoLinkMostra.java
    |   AssociazioneMostra.java
    |   TipoLinkCoinvolge.java
    |   TipoLinkOccupa.java
    |   EccezioneSubset.java
    |   EccezioneMolteplicita.java
    |   EccezionePrecondizioni.java
    |
    +---Icona
    |       Icona.java
    |
    +---IconaAttiva
    |       IconaAttiva.java
    |
    +---Applicazione
    |       Applicazione.java
    |
    +---DisplayArea
    |       DisplayArea.java
    |
    \---Animazione
            Animazione.java
```

La classe Java Icona

```
// File AppCellulare/Icona/Icona.java
package AppCellulare.Icona;
//import AppCellulare.*;
import java.util.*;

public class Icona {
  private String codice;
  private String immagine;

  public Icona(String codice, String immagine){
    this.codice = codice;
    this.immagine=immagine;
  }

  public String getCodice(){
    return codice;
  }

  public String getImmagine(){
    return immagine;
  }
}
```

La classe Java IconaAttiva

```
// File AppCellulare/IconaAttiva/IconaAttiva.java
package AppCellulare.IconaAttiva;
import AppCellulare.*;
import AppCellulare.Icona.*;
import AppCellulare.Applicazione.*;
import java.util.*;

public final class IconaAttiva extends Icona{
  private final int MOLT_MIN_MOSTRA=1,MOLT_MIN_OCCUPA=1;
  private String suonoAlClick;
  private Applicazione applicazione;
  private LinkedList<TipoLinkMostra> mostra;
  private HashSet<TipoLinkOccupa> occupa;


  public IconaAttiva(String codice, String immagine, String suonoAlClick){
    super(codice, immagine);
    this.suonoAlClick = suonoAlClick;
    applicazione = null;
    mostra = new LinkedList<TipoLinkMostra>();
    occupa = new HashSet<TipoLinkOccupa>();
  }

  public String getSuonoAlClick(){
```

```
  return suonoAlClick;
}

public void inserisciApplicazione(Applicazione a){
  if (a != null)
    this.applicazione = a;
}

public Applicazione getApplicazione() throws EccezioneMolteplicita{
  if (applicazione == null)
    throw new EccezioneMolteplicita("Molteplicità min/max violate");
  return applicazione;
}

public void eliminaApplicazione(){
  applicazione = null;
}

public void inserisciLinkMostra(AssociazioneMostra a){
  if (a != null && !mostra.contains(a.getLink())){
    mostra.add(a.getLink());
  }
}

public void eliminaLinkMostra(AssociazioneMostra a){
  if (a != null)
```

```
    mostra.remove(a.getLink());
  }

    public List<TipoLinkMostra> getLinkMostra() throws EccezioneMolteplicita{
      if (mostra.size() < MOLT_MIN_MOSTRA)
        throw new EccezioneMolteplicita("Molteplicità minima violata");
      return (LinkedList<TipoLinkMostra>) mostra.clone();
      }

  public void inserisciLinkOccupa(TipoLinkOccupa a){
    if (a != null)
      occupa.add(a);
  }

  public void EliminaLinkOccupa(TipoLinkOccupa a){
    if (a != null)
      occupa.remove(a);
  }

    public Set<TipoLinkOccupa> getLinkOccupa() throws EccezioneMolteplicita{
      if (occupa.size() < MOLT_MIN_OCCUPA)
        throw new EccezioneMolteplicita("Molteplicità minima violata");
      return (HashSet<TipoLinkOccupa>) occupa.clone();
      }

}
```

La classe Java Applicazione

```
// File AppCellulare/Applicazione/Applicazione.java
package AppCellulare.Applicazione;

import AppCellulare.*;
import java.util.*;

public class Applicazione {
  private String nome;
  private String nomeFile;

  public Applicazione(String nome, String nomeFile){
    this.nome = nome;
    this.nomeFile = nomeFile;
  }

  public String getNome(){
    return nome;
  }

  public String getNomeFile(){
    return nomeFile;
  }
}
```

La classe Java DisplayArea

```
// File AppCellulare/DisplayArea/DisplayArea.java
package AppCellulare.DisplayArea;
import AppCellulare.*;
import java.util.*;

public class DisplayArea {
  private int posizione;
  private String backgrnd;

  public DisplayArea(int posizione, String backgrnd){
    this.posizione = posizione;
    this.backgrnd = backgrnd;
  }

  public int getPosizione(){
    return posizione;
  }

  public String getBackgrnd(){
    return backgrnd;
  }
}
```

La classe Java Animazione

```
// File AppCellulare/Animazione/Animazione.java
package AppCellulare.Animazione;

import AppCellulare.*;
import java.util.*;

public class Animazione{
  private String linkProg;
  private HashSet<TipoLinkMostra> mostra;
  private HashSet<TipoLinkCoinvolge> coinvolge;
  private final int MOLT_MIN=1;

  public Animazione(String linkProg){
    this.linkProg = linkProg;
    mostra = new HashSet<TipoLinkMostra>();
    coinvolge = new HashSet<TipoLinkCoinvolge>();
  }

  public String getLinkProg(){
    return linkProg;
  }

  public void inserisciLinkMostra (AssociazioneMostra a){
    if (a != null)
```

```
    mostra.add(a.getLink());
}

public void eliminaLinkMostra(AssociazioneMostra a){
  if (a != null)
    mostra.remove(a.getLink());
}

public Set<TipoLinkMostra> getLinkMostra(){
  return (HashSet<TipoLinkMostra>) mostra.clone();
}

public void inserisciLinkCoinvolge(TipoLinkCoinvolge c){
  if (c != null)
    coinvolge.add(c);
}

public Set<TipoLinkCoinvolge> getLinkCoinvolge() throws EccezioneMolteplicita{
  if (coinvolge.size() < MOLT_MIN)
    throw new EccezioneMolteplicita("Cardinalità minima violata");
  return (HashSet<TipoLinkCoinvolge>) coinvolge.clone();
}

public void eliminaLinkCoinvolge(TipoLinkCoinvolge c){
  if (c != null)
    coinvolge.remove(c);
```

```
    }
}
```

La classe Java TipoLinkMostra

```
// File AppCellulare/TipoLinkMostra.java
package AppCellulare;
import AppCellulare.IconaAttiva.*;
import AppCellulare.Animazione.*;
import java.util.*;

public class TipoLinkMostra{
    private final IconaAttiva laIconaAttiva;
    private final Animazione laAnimazione;

    public TipoLinkMostra(IconaAttiva ia, Animazione a)
        throws EccezionePrecondizioni {
        if (ia == null || a == null)  // CONTROLLO PRECONDIZIONI
            throw new EccezionePrecondizioni
                ("Gli oggetti devono essere inizializzati");
        laIconaAttiva = ia;
        laAnimazione = a;
    }

    public boolean equals(Object o) {
        if (o != null && getClass().equals(o.getClass())) {
            TipoLinkMostra l = (TipoLinkMostra) o;
            return l.laIconaAttiva == laIconaAttiva &&
                l.laAnimazione == laAnimazione;
```

```
        }
        else return false;
    }

    public int hashCode() {
      return  laIconaAttiva.hashCode() + laAnimazione.hashCode();
    }

    public IconaAttiva getIconaAttiva(){
      return laIconaAttiva;
    }

    public Animazione getAnimazione(){
      return laAnimazione;
    }

    public String toString() {
        return "<" + laIconaAttiva + ", " + laAnimazione + ">";
    }
}
```

La classe Java AssociazioneMostra

```
// File AppCellulare/AssociazioneMostra.java
package AppCellulare;

public final class AssociazioneMostra{
  private TipoLinkMostra link;

  private AssociazioneMostra(TipoLinkMostra link){
    this.link = link;
  }

  public TipoLinkMostra getLink(){
    return link;
  }

  public static void inserisci(TipoLinkMostra y) {
    if (y != null) {
      AssociazioneMostra k = new AssociazioneMostra(y);
      y.getIconaAttiva().inserisciLinkMostra(k);
      y.getAnimazione().inserisciLinkMostra(k);
    }
  }

  public static void elimina(TipoLinkMostra y) {
    if (y != null) {
```

```
      AssociazioneMostra k = new AssociazioneMostra(y);
      y.getIconaAttiva().eliminaLinkMostra(k);
      y.getAnimazione().eliminaLinkMostra(k);
    }
  }
}
```

La classe Java TipoLinkCoinvolge

```
// File AppCellulare/TipoLinkCoinvolge.java
package AppCellulare;
import AppCellulare.Animazione.*;
import AppCellulare.DisplayArea.*;
import java.util.*;

public class TipoLinkCoinvolge{
    private final Animazione laAnimazione;
    private final DisplayArea laDisplayArea;

    public TipoLinkCoinvolge(Animazione a, DisplayArea da)
        throws EccezionePrecondizioni {
        if (a == null || da == null)  // CONTROLLO PRECONDIZIONI
            throw new EccezionePrecondizioni
                ("Gli oggetti devono essere inizializzati");
        laAnimazione = a;
        laDisplayArea = da;
    }

    public boolean equals(Object o) {
        if (o != null && getClass().equals(o.getClass())) {
            TipoLinkCoinvolge l = (TipoLinkCoinvolge) o;
            return l.laAnimazione == laAnimazione &&
                l.laDisplayArea == laDisplayArea;
```

```
        }
        else return false;
    }

    public int hashCode() {
      return  laAnimazione.hashCode() + laDisplayArea.hashCode();
    }

    public Animazione getAnimazione(){
      return laAnimazione;
    }

    public DisplayArea getDisplayArea(){
      return laDisplayArea;
    }

    public String toString() {
        return "<" + laAnimazione + ", " + laDisplayArea + ">";
    }
}
```

La classe Java TipoLinkOccupa

```
// File AppCellulare/TipoLinkOccupa.java
package AppCellulare;
import AppCellulare.IconaAttiva.*;
import AppCellulare.DisplayArea.*;
import java.util.*;

public class TipoLinkOccupa{
    private final IconaAttiva laIconaAttiva;
    private final DisplayArea laDisplayArea;

    public TipoLinkOccupa(IconaAttiva a, DisplayArea da)
        throws EccezionePrecondizioni {
        if (a == null || da == null)  // CONTROLLO PRECONDIZIONI
            throw new EccezionePrecondizioni
                ("Gli oggetti devono essere inizializzati");
        laIconaAttiva = a;
        laDisplayArea = da;
    }

    public boolean equals(Object o) {
        if (o != null && getClass().equals(o.getClass())) {
            TipoLinkOccupa l = (TipoLinkOccupa) o;
            return l.laIconaAttiva == laIconaAttiva &&
                l.laDisplayArea == laDisplayArea;
```

```
        }
        else return false;
    }

    public int hashCode() {
      return  laIconaAttiva.hashCode() + laDisplayArea.hashCode();
    }

    public IconaAttiva getIconaAttiva(){
      return laIconaAttiva;
    }

    public DisplayArea getDisplayArea(){
      return laDisplayArea;
    }

    public String toString() {
        return "<" + laIconaAttiva + ", " + laDisplayArea + ">";
    }
}
```

Realizzazione in Java delle classi per eccezioni

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

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

// File AppCellulare/EccezionePrecondizioni.java
package AppCellulare;

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;
  }
}
```