

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 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.)

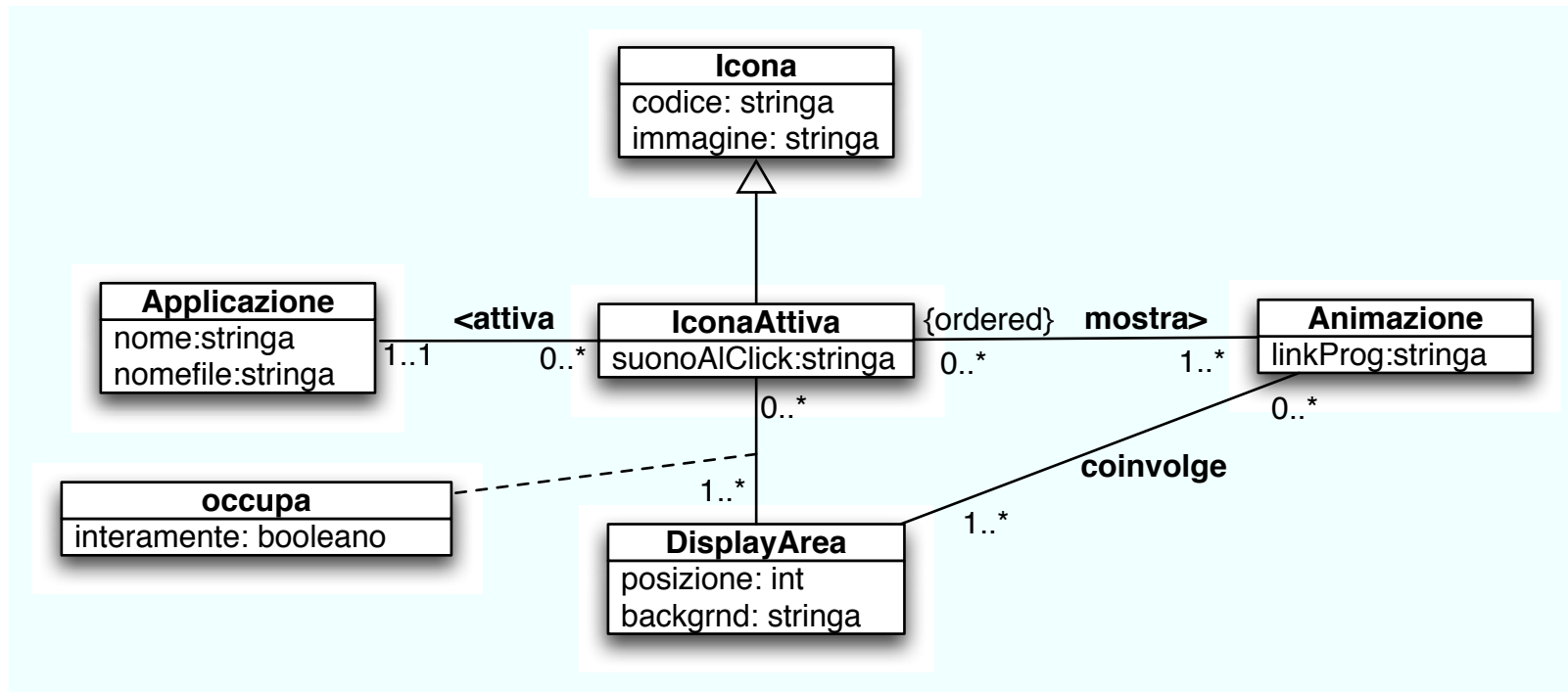
Domanda 1. Basandosi sui requisiti riportati sopra, effettuare la fase di analisi producendo lo schema concettuale in UML per l'applicazione e motivando, qualora ce ne fosse bisogno, le scelte effettuate.

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. 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.
<i>attiva</i>	<i>IconaAttiva</i>	SI ³
	<i>Applicazione</i>	NO
<i>coinvolge</i>	<i>Animazione</i>	SI ³
	<i>DisplayArea</i>	NO
<i>mostra</i>	<i>IconaAttiva</i>	SI ^{2,3}
	<i>Animazione</i>	SI ²
<i>occupa</i>	<i>IconaAttiva</i>	SI ³
	<i>DisplayArea</i>	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
<i>Icona</i>	<i>codice</i>
	<i>immagine</i>
<i>IconaAttiva</i>	<i>suonoAlClick</i>
<i>Applicazione</i>	<i>nome</i>
	<i>nomeFile</i>
<i>Animazione</i>	<i>linkProg</i>
<i>DisplayArea</i>	<i>posizione</i>
	<i>backgrnd</i>

Classe UML	Proprietà	
	nota alla nascita	non nota alla nascita
<i>IconaAttiva</i>	—	<i>applicazione</i>

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
|   ManagerMostra.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;

public class Icona {
    protected final String codice;
    protected final 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 final String suonoAlClick;
    private Applicazione applicazione;
    private LinkedList<TipoLinkMostra> mostra; //ordered
    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("Molteplicita' min/max violate");
    return applicazione;
}

public void eliminaApplicazione() {
    applicazione = null;
}

public void inserisciLinkMostra(TipoLinkMostra a) {
    if (a != null && a.getIconaAttiva() == this)
        ManagerMostra.inserisci(a);
}

public void eliminaLinkMostra(TipoLinkMostra a) {

```

```

    if (a != null && a.getIconaAttiva() == this)
        ManagerMostra.elimina(a);
}

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

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

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

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

```

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

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

}
```

La classe Java Applicazione

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

```
public class Applicazione {
    private final String nome;
    private final 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;

public class DisplayArea {
    private final int posizione;
    private final 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 AppCellulare.DisplayArea.DisplayArea;

import java.util.*;

public class Animazione {
    private final String linkProg;
    private HashSet<TipoLinkMostra> mostra;
    private HashSet<DisplayArea> linkCoinvolge;
    private final int MOLT_MIN_COINVOLGE = 1;

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

    public String getLinkProg() {
        return linkProg;
    }
}
```

```

public void inserisciLinkMostra(TipoLinkMostra a) {
    if (a != null && a.getAnimazione() == this)
        ManagerMostra.inserisci(a);
}

public void eliminaLinkMostra(TipoLinkMostra a) {
    if (a != null && a.getAnimazione() == this)
        ManagerMostra.elimina(a);
}

public void inserisciPerManagerMostra(ManagerMostra a) {
    if (a != null)
        mostra.add(a.getLink());
}

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

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

public void inserisciLinkCoinvolge(DisplayArea da) {
    if (da != null)

```

```

        linkCoinvolge.add(da);
    }

    public void eliminaLinkCoinvolge(DisplayArea da) {
        if (da != null)
            linkCoinvolge.remove(da);
    }

    public Set<DisplayArea> getLinkCoinvolge() throws EccezioneMolteplicita {
        if (linkCoinvolge.size() < MOLT_MIN_COINVOLGE)
            throw new EccezioneMolteplicita("Cardinalita' minima violata");
        return (HashSet<DisplayArea>) linkCoinvolge.clone();
    }
}

```


La classe Java TipoLinkMostra

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

import AppCellulare.IconaAttiva.*;
import AppCellulare.Animazione.*;

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 ManagerMostra

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

public final class ManagerMostra {

    private TipoLinkMostra link;

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

    public TipoLinkMostra getLink() {
        return link;
    }

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

    public static void elimina(TipoLinkMostra y) {
```

```
if (y != null) {  
    ManagerMostra k = new ManagerMostra(y);  
    y.getIconaAttiva().eliminaPerManagerMostra(k);  
    y.getAnimazione().eliminaPerManagerMostra(k);  
}  
}  
}
```

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;
    private final boolean interamente;

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

    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 boolean getInteramente() {
    return interamente;
}

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

```

La classe Java OperazioniUtente

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

import AppCellulare.IconaAttiva.*;
import AppCellulare.Animazione.*;
import java.util.*;

public final class OperazioniUtente {

    public static List<Animazione> invertiAnimazioni(IconaAttiva ia)
        throws EccezioneMolteplicita {
        HashSet<TipoLinkMostra> linkAnimazioni = new HashSet<TipoLinkMostra>();
        List<TipoLinkMostra> linkMostra = ia.getLinkMostra();
        Iterator<TipoLinkMostra> it = linkMostra.iterator();
        while (it.hasNext()) {
            TipoLinkMostra l = it.next();
            linkAnimazioni.add(l);
        }
        LinkedList<Animazione> result = new LinkedList<Animazione>();
        int i = linkAnimazioni.size();
        while (i > 0) {
            Iterator<TipoLinkMostra> itA = linkAnimazioni.iterator();
            while (itA.hasNext()) {
                TipoLinkMostra la = itA.next();
```

```

        if (linkMostra.indexOf(la) == i) {
            linkAnimazioni.remove(la);
            result.add(la.getAnimazione());
            i--;
            // NOTA: il ciclo continua anche se l'elemento corrente e'
            // stato scelto. Il prossimo elemento da selezionare
            // potrebbe, infatti, trovarsi in quelli non ancora scansionati.
        }
    }// while(itA.hasNext())
} // while(i > 0)
return result;
}

public static Set<IconaAttiva> chiMostra(Animazione a) {
    HashSet<IconaAttiva> result = new HashSet<IconaAttiva>();
    Set<TipoLinkMostra> linksMostra = a.getLinkMostra();
    Iterator<TipoLinkMostra> it = linksMostra.iterator();
    while (it.hasNext()) {
        TipoLinkMostra linkMostra = it.next();
        result.add(linkMostra.getIconaAttiva());
    }
    return result;
}
}

```


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;  
}  
}
```