Esercitazione n° 5

Obiettivi

- Progettare gerarchie di classi
- Acquisire manualità nella scrittura di algoritmi non banali
- Linguaggio Java:
  - Gestione Eccezioni
  - Import e uso di classi esterne
- Esempio guida: scambio di messaggi tra cellulari

Specifica del Problema (1)

Scambio di SMS tra telefoni cellulari

Occorre modellare i telefoni cellulari in termini di servizi offerti al cliente.

Ai fini dell'esercitazione, ogni ***Cellulare*** dovrà essere modellato come una astrazione di dato e sarà costituito da alcune caratteristiche essenziali:

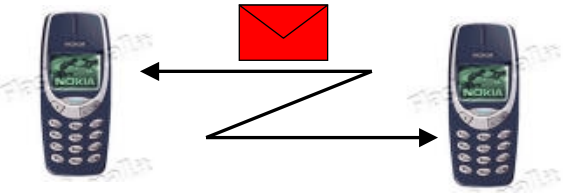
1 Ogni telefono può

chiamare un altro Cellulare (***void chiama(Cellulare dest)***)
suonare (***void suona()***)
squillare (***void squilla()***).

2 Ogni telefono possiede un Servizio per lo scambio di SMS

Specifica del Problema (2)

Gestione SMS…



Ogni cellulare può:

- Inviare ad un altro cellulare un messaggio di testo
  void inviaSms(String testo, Cellulare dest)
- Controllare se ci sono messaggi in memoria
  void controllaSms()
- Leggere un messaggio in memoria in posizione x
  void leggiSms(int posizioneSMS)
- Eliminare un messaggio in memoria in posizione x
  void eliminaSms(int posizioneSMS)
- Memorizzare un messaggio in memoria
  void memorizzaSms(Sms msg)

Specifica del Problema (3)

Inoltre…

Alcuni modelli (non tutti) oltre ai servizi di base elencati possiedono anche alcuni servizi WAP. Tali cellulari dovranno essere modellati con il nome di ***CellulareWAP***

Tali cellulari possono:

- Collegarsi al fornitore dei servizi (provider)
  void collProvider()
- Navigare tra le pagine
  void naviga()

NB: i valori di ritorno e gli argomenti scelti per i metodi sono adatti solamente ai fini dell'esercitazione e non hanno presunzione di modellare la realtà….

Specifica del Problema (4)

Inoltre…

Alcuni modelli (non tutti) oltre ai servizi di base elencati possiedono anche una piccola agenda elettronica. Tali cellulari dovranno essere modellati con il nome di ***CellulareOrganizer***

Tali cellulari possono:

- Gestire un calendario
  void calendario()
- Gestire degli appuntamenti
  void appuntamenti()

Specifica del Problema (5)

Importante!

Nonostante le funzionalità di tutti i cellulari sono comuni e conformi alla specifica data, ogni casa costruttrice può poi implementare a piacere tali funzionalità sui propri telefoni.

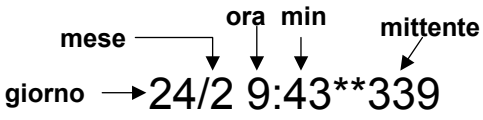
Per esempio tutti i telefoni cellulari, di differenti modelli e case costruttrici potranno scambiarsi dei messaggi utilizzando il metodo ***inviaSms***, ma ogni modello potrà implementare tale metodo a piacere, in maniera conforme, per es, alla tecnologia adottata.

Gli SMS

Specifica

- Ogni Sms è costituito da un **mittente** che lo ha spedito (`int` che indica un numero di cellulare), un **giorno**, **mese**, **ora** e minuto (**min**) di spedizione (tutti `int`) e un **messaggio** che è una stringa di testo.

- Possiede un unico costruttore che riceve come argomenti il mittente ed il messaggio di testo. Dopo aver inizializzato opportunamente con i parametri ricevuti le due variabili interne corrispondenti, provvede anche alla definizione delle altre attraverso i metodi della classe eserc_utility.Utilita.
- Il metodo toString() mostra l'Header del Sms:



Il NOKIA 3330 (1)

Nokia3330 è un cellulare con le funzionalità aggiuntive di Organizer e relative al Wap.

- Internamente questo modello memorizza il numero del cellulare **numCell** (`int`) e possiede una memoria contenente al massimo 10 SMS **memoriaSms** (per semplificare, si semplifica la realtà omettendo la gestione della scheda SIM). Tali variabili sono *private*.

  Inoltre, per comodità, possiede una `stringa` con il **prompt** da visualizzare ad ogni messaggio all'utente del tipo:

  "NOKIA [numCell] >"

- Il costruttore,senza argomenti, assegna un numero progressivo univoco a ***numCell***, inizializza tutto ciò che è necessario, quindi stampa in uscita: [prompt] + "ATTIVO!"

Il NOKIA 3330 (2)

Gestione SMS

- ***void inviaSms (String testo, Cellulare dest)***

  1 Crea un nuovo Sms con mittente = numCell del telefono e il **testo** passato come argomento

  2 Memorizza il messaggio sul cellulare ***dest***

  3 Fa squillare il cellulare ***dest***

- ***void memorizzaSms (Sms msg)***

  1 Controlla se memoriaSms è piena

  1.1 Se è piena scrive: [prompt] + "La memoria è piena!"

  1.2 Altrimenti scrive il messaggio nella prima cella di memoria libera

Suggerimento (non vincolante): su usi un intero come indice per tenere traccia del numero di messaggi memorizzato e riempire l'array della memoria in maniera ordinata (senza buchi), mantenendo l'array ordinato (senza buchi) anche a fronte di eliminazioni di messaggi.

Il NOKIA 3330 (3)

Gestione SMS

- ***void controllaSms ()***

  Visualizza in uscita l'elenco delle intestazioni di tutti i messaggi presenti in memoria ed il numero del messaggio come, per esempio:

Numero (posizione)

```
NOKIA:2> Elenco Messaggi
1->24/2 19:21**339
2->24/2 19:21**111
3->24/2 19:21**35
NOKIA:2> FINE Elenco Messaggi
```

- ***void eliminaSms (int posizioneSMS)***

  Elimina uno dei messaggi presenti in memoria dato il suo numero. Se il numero non esiste scrive in uscita [prompt] + "Messaggio inesistente!", altrimenti [prompt] + Messaggio Eliminato!"

Il NOKIA 3330 (4)

Gestione SMS

- ***void leggiSms (int posizioneSMS)***

  Dato il numero **posizioneSMS** che indica il numero del messaggio ricavabile,come visto, dal metodo *controllaSms*, visualizza il messaggio completo corrispondente nel modo seguente:

```
NOKIA:2> Messaggio N. 1
24/2 19:21**339
In bocca al lupo x l'esame!
Ciao
** FINE messaggio **
```


Il NOKIA 3330 (5)

Altri metodi

- ***void suona()***

  Stampa in uscita [prompt] + “DRIN! DRIN! DRIN!”

- ***void squilla()***

  Stampa in uscita [prompt] + “BIP!”

- ***void chiama(Cellulare dest)***

  1 Scrive in uscita [prompt] + “Chiamata in corso…”
  2 Fa suonare il cellulare ***dest***

Il NOKIA 3330 (6)

Vengono forniti gli altri metodi il cui comportamento non è rilevante ai fini dell'esercitazione.

```
// Cellulare Organizer
public void calendario(){System.out.println( prompt + "
                                    il calendario");}
public void appuntamenti() {System.out.println(prompt +
                                  " gli appuntamenti");}
// Cellulare Wap
public void collProvider(){ System.out.println( prompt
                                + ":  Connessione..");}
public void naviga(){ System.out.println( prompt + ":
                                  Esplora pagine");}
```


MainProgram (1)

MainProgram è il componente software che gestisce una simulazione relativa ad uno scambio di messaggi tra cellulari

- Da linea di comando accetta uno o più argomenti. Il componente software dovrà controllare che almeno uno sia stato passato, in caso negativo scrive in uscita "Almeno un argomento!" e termina l'esecuzione (*System.exit(0)*).
- Gli argomenti sono i nomi degli oggetti di tipo **Cellulare** che si vogliono creare che saranno messi tutti in un array (**arr[]**). Per es., il comando:

```
c:\TEMP> java MainProgram Nokia3330 Nokia3330 Nokia3330
```

Crea un array di tipo Cellulare con tre oggetti Nokia3330

Suggerimento: si usi il metodo *creaOggetti()* di **eserc_utility.Utilita**

MainProgram (2)

- Se si lancia il programma con un argomento di nome sbagliato (non corrispondente ad alcuna classe) si scatena un'eccezione. La si catturi stampando in uscita un messaggio relativo all'errore con il consiglio "La classe [nome classe] non esiste. Controlla il nome e il CLASSPATH!"
- Creato l'array necessariamente **arr[0]** esisterà perché occorre passare almeno un argomento. Si controlli se questo Cellulare ha messaggi in memoria.
- Si mandino da tutti gli altri Cellulari in **arr[]** (se esistono), due Sms diversi ad **arr[0]** con messaggi a piacere.
- Si ricontrolli quindi se **arr[0]** ha dei messaggi, quindi ordinatamente:
  1 Si provi a leggere il primo ed eliminare il quarto Sms in memoria.
  2 Si ricontrolli la memoria e si provi a leggere il secondo Sms.

Il Package eserc_utility

La directory **eserc_utility** contiene il bytecode relativo alla classe **Utilita** che contiene alcuni metodi utili per completare l'esercitazione. Non viene fornito il sorgente, ma solo la documentazione (si apra il file **Index.html** presente nella directory sotto `docs`).

Questa classe dovrà essere importata ove necessario e i suoi metodi usati in maniera conforme alla documentazione fornita.

Si intende simulare in questo modo l'uso di 'pacchetti' già pronti, scritti da terzi.

Esercitazione n°5 (1)

Come procedere?

1. Progettare e implementare la gerarchia di classi per modellare il mondo dei telefoni cellulari come descritto.
2. Scaricare il file `Utility.zip` e decomprimerlo in `c:\Temp`. Verrà creata una cartella Eserc_Utilita. Esplorarne il contenuto consultando la slide N°16.
3. Implementare la classe **Sms.java** seguendo la specifica della slide N°7, quindi compilarla.
4. Implementare una classe concreta della gerarchia progettata : la classe **Nokia3330.java** seguendo la specifica delle slide N°8-13, quindi compilarla.

Esercitazione n°5 (2)

Come procedere?

5. Implementare la classe **MainProgram** come descritto nelle slide N°14-15 e compilarla.
6. Eseguire provando a passare un diverso numero di argomenti al metodo main.

```
c:\TEMP> java MainProgram Nokia3330 ...
```

Esercitazione n°5: Facoltativo

NOKIA 3210

Implementare anche una classe per modellare i cellulari NOKIA 3210. Tali cellulari sono analoghi al 3330 ma senza i servizi WAP e senza Organizer e cambiare il MainProgram a piacere provando a scambiare Sms e chiamate tra i due telefoni….

Si possono mettere in gerarchia anche le classi Nokia3330 e Nokia3210? E' utile? E' corretto?

Cosa richiederebbe una buona progettazione?

APPENDICE JCreator

Per passare degli argomenti al Main si può evitare di usare la linea di comando usando JCreator

Si riporta un passaggio del manuale di JCreator in cui viene spiegata una procedura che chiede all'utente gli argomenti per il main da interfaccia grafica, dopo aver premuto "Execute File"

1 Choose Options from the Configure menu.

2 Select JDK Tools, select Run Application from the pulldown list, Edit.

3 The Tool Configuration dialogue box will be displayed.

4 Enable the checkbox "Prompt for main function arguments".