LABORATORIO DI PROGRAMMAZIONE 2009-2010
EDIZIONE 2 - TURNI A E B
EDIZIONE 3 (SERALE)
PROVA D'ESAME

Andrea Trentini - DICo - UniMi

Febbraio 2010

Avvertenza. Non e' ammesso l'uso delle classi del package *prog.io* allegato al libro di testo del corso.

1 Testo esame

Lo scopo e' rappresentare un telefono cellulare semplificato (modello concettuale) che gestisce solo messaggi sms e una rubrica di numeri telefonici.
Le classi da realizzare sono le seguenti:

- *Cellulare*, rappresenta un'unita' mobile
- *SMS*, rappresenta un messaggio di testo
- *MMS*, rappresenta un messaggio "multimediale"
- *NumeroTelefonico*, rappresenta un numero telefonico con il nome associato
- *Main*, contiene il solo metodo *main()* con alcune invocazioni di test (importante e' riuscire a testare TUTTI i metodi almeno una volta)

Tali classi dovranno esporre almeno i metodi specificati nelle sezioni seguenti. Eventuali metodi di servizio possono essere aggiunti a piacimento.
Ogni classe deve avere il *toString()* che rappresenti lo stato delle istanze e i **costruttori** adeguati per gli attributi che vengono dichiarati.
Si suggerisce, anche dove non segnalato, di utilizzare, se esistenti e se applicabili, le classi parametriche (es. *Vector<E>* invece di *Vector*).

1.1 Cellulare

Contiene una lista (usare *Vector<E>*) di numeri telefonici (istanze di *NumeroTelefonico*).
Mantiene anche la lista (usare *Vector<E>*) degli sms inviati con successo.
Deve esporre i seguenti metodi **pubblici**:

- *void aggiungiNumero(NumeroTelefonico num)*, aggiunge un numero alla rubrica
- *boolean rimuoviNumero(int indice)*, toglie un numero dalla rubrica
- *NumeroTelefonico trovaNumero(String nome)*, trova un numero a partire dal nome
- *String trovaNome(String numero)*, trova il nome a partire dal numero (in forma di stringa)
- *boolean inviaSMS(SMS sms)*, simula l'invio di un sms, il tasso di successo deve essere del 95% (si suggerisce di usare un numero casuale per decidere se il msg parte o meno), ritorna false se il msg non e' stato inviato
- *SMS[] elencoSMSinviati()*, restituisce un array con tutti gli sms inviati presenti nel telefono

1.2 SMS

Contiene degli attributi (di tipo opportuno) per memorizzare un *NumeroTelefonico* e un messaggio di testo (limite max. 160 caratteri).
Deve esporre i seguenti metodi **pubblici**:

- *void setNumero(NumeroTelefonico num)*
- *NumeroTelefonico getNumero()*
- *boolean setTesto(String testo)*, restituisce false se il testo e' troppo lungo
- *String getTesto()*

1.3 MMS

Estende *SMS* e aggiunge i metodi per la gestione degli allegati multimediali, cioe' deve esporre i seguenti metodi **pubblici**:

- *void setUrl(String url)*, permette di allegare l'URL di un'immagine, non e' necessario fare complicati controlli di coerenza, ma il metodo deve verificare che la stringa inizi con "http://"
- *String getUrl()*

1.4 NumeroTelefonico

Contiene degli attributi (di tipo opportuno) per memorizzare un nome e un numero di telefono (attenzione agli zeri dei prefissi).
Deve esporre i seguenti metodi **pubblici**:

- *void setNumero(String num)*
- *String getNumero()*
- *void setNome(String nome)*
- *String getNome()*

1.5 Main

Deve contenere il metodo *main()* in cui vanno istanziate le classi realizzate e opportunamente testate, ad es. creando qualche numero telefonico, qualche sms, inviando degli sms, etc.

2 Consegna

Ricordo che le classi devono essere tutte *public* e che vanno consegnati tutti i file *.java* prodotti.
Per la consegna, eseguite l'upload dei singoli file sorgente dalla pagina web: http://upload.dico.unimi.it.