COGNOME: **NOME:** **MATRICOLA:**

Programmazione II, 13 settembre 2011 – Laurea in INFORMATICA

Esercizio 1[16 punti]
Considerare la seguente gerarchia di eccezioni:

```
public class EccezionePila extends Exception {
    public EccezionePila(String s) { super(s);}
}

public class EccezionePilaPiena extends EccezionePila {
    public EccezionePilaPiena(String s) { super(s);}
}

public class EccezionePilaVuota extends EccezionePila {
    public EccezionePilaVuota(String s) { super(s);}
}
```

(a) Modificare il seguente codice in modo che i metodi `aggiungi` e `estrai` lancino le opportune eccezioni invece di stampare messaggi di errore:

```
public class PilaInteriLimitata {
    private int[] pila;
    private int top = 0;

    public PilaInteriLimitata(int max) {
this.pila = new int[max];
    }

    public boolean vuota() {
return top <= 0;
    }

    public boolean piena() {
return top > pila.length-1;
    }

    public int estrai() {
if (vuota()) {
    System.out.println("La pila e' vuota");
    return -1;
}
return pila[(top--)-1];
    }

    public void aggiungi(int x) {
if (piena()) {
    System.out.println("La  pila e' piena");
    return;
}
pila[top++] =x;
    }
}
```

(b) Scrivere un programma `UsaPilaInteriLimitata` che crea una pila di 5 elementi e prova a inserirne ed estrarne 6. Il programma deve catturare le eccezioni e stampare messaggi di errore.

Esercizio 2[16 punti]
Data la seguente classe astratta

```
abstract class Messaggio {
    protected String mittente;
    protected String destinatario;
    protected String data;
    protected String ora;

    public void impostaMittente(String mittente) {
this.mittente = mittente;
    }

     public void impostaDestinatario(String destinatario) {
this.mittente = mittente;
    }

    public void impostaOra(String ora) {
this.mittente = mittente;
    }

     public void impostaData(String data) {
this.mittente = mittente;
    }
}
```

definire le seguenti tre classi per rappresentare SMS *normali* (cioè lunghi al massimo 160 caratteri) e *lunghi* (senza limite di caratteri).

(a) La classe `MessaggioDiTesto` è una sottoclasse astratta di `Messaggio`, i cui oggetti sono messaggi con in più un testo. La classe contiene le variabili di istanza `protected`:

- `testo` di tipo `char[]`,

- `numCaratteri` di tipo `int`, inizializzata a 0,

e i seguenti metodi:

- il metodo astratto `boolean aggiungi(char c)` che stabilisce come aggiungere caratteri al testo di un messaggio;

- il metodo `String toString()` che restituisce una rappresentazione in forma di stringa del messaggio.

(b) La classe `SMS` definisce oggetti che sono messaggi di testo di al più 160 caratteri. Essa estende la classe astratta `MessaggioDiTesto` ridefinendone il metodo astratto; tale metodo dovrà restituire `true` se e solo se è possibile aggiungere `c` al testo senza superare la lunghezza limite, e in questo caso aggiungere il carattere al testo. La classe contiene inoltre un costruttore che inizializza `testo` con un array di caratteri di lunghezza pari alla lunghezza massima del messaggio.

(c) La classe `SMSLungo` definisce invece oggetti che sono messaggi di testo di lunghezza non limitata. Essa estende la classe astratta `MessaggioDiTesto` ridefinendone il metodo astratto; tale metodo restituisce `true` dopo aver aggiunto `c` al testo. La classe contiene inoltre

- una variabile privata `testoLungo` di tipo `String`, inizializzata con la stringa vuota,

- un metodo d'istanza `SMS[] spezza()` che divide opportunamente un messaggio lungo in messaggi normali che restituisce dopo averli organizzati in un array.

Scrivere infine un programma `UsaSMS` che permette di inserire da tastiera un SMS normale e uno lungo e poi li stampa a video.