Studstats: Esonero 3 del corso di Metodologie di Programmazione
Corso di Laurea Triennale in Informatica - Sapienza
prof. Roberto Navigli – A.A. 2017/2018
(Esonero 2 per gli studenti in teledidattica)

v1.1 (aggiornamenti pre-v1.1 in verde e su v1.1 in arancio - modifiche necessarie per permettere la correzione automatica - mi scuso per l'inevitabile disagio)

La possibilità di commentare è chiusa. Eventualmente utilizzate facebook (ma direi che ormai dovrebbe essere tutto chiaro).

CONSEGNA: domenica 20 maggio (24:00 Pacific Time)

L'homework consiste nella realizzazione del sistema modulare **Studstats** per l'analisi statistica delle informazioni relative agli studenti immatricolati quest'anno in informatica, ottenibile dal gruppo facebook del corso, sezione Files, `IMMATRICOLATI_INFORMATICA_SAPIENZA_2018_randomizzato.csv`. **NOTA BENE: il file NON deve essere consegnato insieme al progetto e si deve assumere che il file sia alla radice del progetto, ovvero FUORI della cartella src, ma DENTRO il progetto.**

Si richiede di implementare le seguenti classi da specificare all'interno del package (`it.uniroma1.lcl.studstats`) o in sottopackage appropriati:

- `Studstats`, che implementa l'interfaccia `AggregatoreStatistico`, definita come segue:

```
package it.uniroma1.lcl.studstats;

import java.util.List;

import it.uniroma1.lcl.studstats.dati.Analizzatore;
import it.uniroma1.lcl.studstats.dati.Rapporto;
import it.uniroma1.lcl.studstats.dati.Studente;
import it.uniroma1.lcl.studstats.dati.TipoRapporto;

// se vi serve potete aggiungere metodi e riconsegnare l'interfaccia
// purche' nel package it.uniroma1.lcl.studstats
public interface AggregatoreStatistico
{
    /**
     * Aggiunge uno studente per l'analisi
     * @param s studente da aggiungere
     */
    void add(Studente s);
```

```
    /**
     * Aggiunge un analizzatore all'aggregatore
     * @param an analizzatore da aggiungere
     */
    void add(Analizzatore an);

    default void addAll(Analizzatore... analizzatori)
    {
        for (Analizzatore a : analizzatori) add(a);
    }


    /**
     * Genera i rapporti dei tipi specificati
     * (tutti gli analizzatori inseriti se non
     * viene specificato nessun tipo).
     * NOTA BENE: non importa se un tipo di rapporto specificato
     * non viene generato da nessuno degli analizzatori. Nel caso
     * peggiore verra' restituita una lista di rapporti vuota.
     * @param tipiRapporto i tipi di cui si vogliono i rapporti
     * @return la lista dei rapporti generati
     */
    List<Rapporto> generaRapporti(TipoRapporto... tipiRapporto);

    /**
     * Restituisce il numero di analizzatori memorizzati
     */
    int numeroAnalizzatori();
}
```

dove `Analizzatore` è definito come segue:

```
package it.uniroma1.lcl.studstats.dati;

import java.util.Collection;

@FunctionalInterface
public interface Analizzatore
{
    Rapporto generaRapporto(Collection<Studente> studs);

    /**
     * Restituisce il tipo di rapporto che genera l'analizzatore
     * NOTA BENE: questo metodo può essere implementato di default
     * utilizzando le annotazioni che saranno viste a lezione
     * venerdì (e, per la teledidattica e gli assenti, spiegate su
```

```
     * diapositive) OPPURE può essere lasciato astratto e
     * implementato in ciascuna sottoclasse (richiedendo la
     * specifica in ciascuna implementazione di Analizzatore). In
     * questo secondo caso non sara' possibile utilizzare le lambda
     * per implementare gli analizzatori base.
     */
    default TipoRapporto getTipo() { /* da implementare */ }
}
```

e `TipoRapporto` è un'interfaccia "segnaposto" definita come segue:

```
package it.uniroma1.lcl.studstats.dati;

public interface TipoRapporto
{
}
```

la classe `Studstats` non permette di costruire oggetti se non tramite il metodo `fromFile` che, preso in input il percorso di un file (o, in alternativa, un `java.nio.file.Path`), restituisce una nuova istanza di `Studstats` costruita con i dati caricati. Il formato del file è: una riga per studente con i campi separati da un campo costante predeterminato (impostato come costante al carattere ';'). La prima riga deve essere saltata, essendo l'intestazione del documento.

- la classe `Rapporto`, che memorizza il tipo di rapporto e le informazioni calcolate. Il metodo `toString` restituisce il rapporto sotto forma di stringa.
- la classe `Studente`, che memorizza le informazioni relative a un singolo studente
- le seguenti implementazioni di `Analizzatore`:
    - `AnalizzatoreAnnoDiploma`: restituisce un rapporto la cui `toString` restituisce le statistiche sull'anno di diploma nel seguente formato, ordinate per anno in ordine decrescente (si noti che è la versione a stringa di una mappa):

      ```
      {ANNI_DIPLOMA={2017=intero, 2016=intero, ecc., 2000=intero}}
      ```

    - `AnalizzatoreIstituti`: restituisce un rapporto la cui `toString` restituisce le statistiche sul numero di studenti per istituto in ordine decrescente per numero di studenti, ad esempio:

      ```
      {ISTITUTI={ALTRO=16, LABRIOLA=14, ecc. ecc.}}
      ```

    - `AnalizzatoreSesso`: genera un rapporto la cui `toString` contiene i numeri di studenti di sesso maschile e di sesso femminile, ad esempio:

      ```
      {SESSO={M=intero, F=intero}}
      ```

- `AnalizzatoreTitoloDiStudio`: genera un rapporto la cui `toString` contiene i titoli di studio degli studenti ordinati in ordine decrescente di valore:

  ```
  {TITOLO={SCIENTIFICO=150, ... , SPERIMENTALE MUSICALE=1}}
  ```

- `AnalizzatoreVoto`: genera un rapporto la cui `toString` contiene il voto medio, il voto massimo, il voto minimo e il voto mediano (ovvero l'elemento centrale della lista ordinata dei voti). **Ad esempio**:

  ```
  {VOTO={VOTO_MEDIO=76.65, VOTO_MAX=101, VOTO_MIN=0, VOTO_MEDIANO=75}}
  ```

  Si stampino solo le prime due cifre decimali dopo la virgola, ad esempio utilizzando `NumberFormat nf = NumberFormat.getNumberInstance(Locale.UK); nf.setMaximumFractionDigits(2);`

- `AnalizzatoreStudentiVotoMaggiore` i cui oggetti sono costruiti con un voto minimo e, opzionalmente, con un `Analizzatore`. Genera un rapporto la cui `toString` è il rapporto dell'analizzatore fornito in fase di costruzione (o, se non specificato, `l'AnalizzatoreSesso`) dei soli studenti con voto >= di quello minimo specificato in fase di costruzione.
- **Bonus:** Un analizzatore a vostra scelta che abbia una certa complessità, magari aggregando più tipi d'informazione (**fino a +3 punti sul punteggio finale**)

Come anche verificato nel JUnit aggiornato, gli analizzatori devono implementare equals e hashCode in modo appropriato, per evitare inutili duplicati (ma allo stesso tempo conservare istanze effettivamente diverse di analizzatori).

Valutazione. Il punteggio è così composto:

- 18 punti se il modulo compila e si esegue correttamente con la classe JUnit `StudstatsTest` riportata in fondo al documento
- 4 punti per l'uso adeguato di Java 8
- 4 punti per l'utilizzo appropriato delle collection nelle varie classi
- 4 punti per la pulizia e il riuso del codice
- 3 punti per l'uso di Javadoc per commentare i package (file package-info.java nella radice del package), le classi, i metodi e i campi

Consegna

La consegna è fissata per **domenica 20 maggio (24:00 Pacific Time)**. La consegna deve essere effettuata all'indirizzo http://robertonavigli.com/metodologie2018. Analogamente a tutte le altre consegne, si deve inviare un file chiamato

`Studstats_<cognome>.<matricola>.zip` **con cognome MINUSCOLO**. Si devono consegnare la cartella `src` (i sorgenti), la cartella `bin` (i file .class) e la cartella `doc` all'interno del file .zip. **Non dovete consegnare il file con le statistiche degli studenti.** Non sono ammessi percorsi assoluti cablati nel codice.

Plagio

Indicazioni significative di plagio porteranno gli studenti coinvolti all'annullamento di tutti gli esoneri svolti (anche in precedenza), **senza distinguere tra chi ha fatto copiare e chi ha copiato**. Si consiglia caldamente **di non condividere il codice con altri studenti (neanche per 5 minuti)**.

Piccolo compendio sulla gestione dei file

Utilizzando la classe `java.io.File` pre-Java 7:

- `File.isFile` e `.isDirectory` per sapere se si tratta di un file o di una cartella
- `File.listFiles` restituisce un array di File contenuti all'interno dell'oggetto (cartella) su cui è invocato. Ad esempio, `new File(PERCORSO_CARTELLA).listFiles()` restituisce gli oggetti `File` corrispondenti ai file contenuti all'interno della cartella.
- Per leggere un file, potete usare il seguente codice:

```
try(BufferedReader br = new BufferedReader(new FileReader(file)))
{
	while(br.ready())
	{
		// leggi ciascuna riga con br.readLine(),
		// che restituisce una stringa con la linea di testo
	}
}
catch(IOException e)
{
	// gestisci l'eccezione di I/O
}
```

Lettura dei file in Java 7-8 mediante il package `java.nio.file`:

- Potete usare la classe `java.nio.file.Files` per ottenere un oggetto di tipo `BufferedReader` che vi permette di leggere un file di testo. Per farlo, potete usare il metodo statico `Files.newBufferedReader` che richiede in input un `Path`. Il metodo vi restituisce comunque un `BufferedReader` che va gestito con il `try catch` come sopra.
- Per ottenere il `java.nio.file.Path` di un file o di una cartella (l'equivalente Java 8 di un `File`) potete usare il metodo statico della classe `Paths`: `Paths.get(PERCORSO)`

- Potete sempre passare da `Path` a `File` con il metodo `Path.toFile` (viceversa, da `File` a `Path` con `File.toPath`)
- Se dovesse servirvi di scrivere un file, è tutto come sopra, ma `Files.newBufferedWriter` o `new BufferedWriter(new FileWriter(PERCORSO))`.

Classe di test JUnit

NOTA BENE: la classe di test è sufficiente a ottenere 18 se l'output sarà ragionevole rispetto ai dati presenti nel file fornito in input. **Non è possibile modificare questa classe in alcun modo.**

```
package it.uniroma1.lcl.studstats;
import static org.junit.jupiter.api.Assertions.assertEquals;

import java.nio.file.Paths;
import java.util.List;
import java.util.Map;
import java.util.Optional;
import java.util.TreeMap;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import java.util.stream.Collectors;

import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

import it.uniroma1.lcl.studstats.dati.Analizzatore;
import it.uniroma1.lcl.studstats.dati.AnalizzatoreIstituti;
import it.uniroma1.lcl.studstats.dati.AnalizzatoreSegretoSegretissimo;
import it.uniroma1.lcl.studstats.dati.AnalizzatoreSesso;
import it.uniroma1.lcl.studstats.dati.AnalizzatoreStudentiVotoMaggiore;
import it.uniroma1.lcl.studstats.dati.AnalizzatoreVoto;
import it.uniroma1.lcl.studstats.dati.Analizzatori;
import it.uniroma1.lcl.studstats.dati.Rapporto;

/**
 * Classe di test JUnit 5
 * @author navigli
 *
 */
class StudstatsTest
{
    public static final String PERCORSO =
"IMMATRICOLATI_INFORMATICA_SAPIENZA_2018_randomizzato.csv";

    private Studstats stats;
    private int counter;
```

```
    @BeforeEach
    void setUp() throws Exception
    {
        if (counter++ % 2 == 0)
                stats = Studstats.fromFile(PERCORSO);
        else
                stats = Studstats.fromFile(Paths.get(PERCORSO));
    }

    @Test
    void testAdd()
    {
        System.out.print("-- testAdd: ");
        stats.addAll(Analizzatori.allBasic());
        stats.add(Analizzatori.studentiVotoMaggiore(80));
        stats.add(Analizzatori.studentiVotoMaggiore(70));
        try {
                assertEquals(stats.numeroAnalizzatori(), 7);
                System.out.println("OK!");
        } catch(Exception e) { System.out.println("NO"); }    }

    @Test
    void testEqualsHashCode()
    {
        System.out.print("-- testEqualsHashCode: ");
        stats.addAll(Analizzatori.allBasic());
        stats.add(Analizzatori.studentiVotoMaggiore(80));
        stats.add(Analizzatori.studentiVotoMaggiore(80, new AnalizzatoreSesso()));
        stats.add(Analizzatori.studentiVotoMaggiore(80, new
AnalizzatoreIstituti()));
        stats.add(Analizzatori.studentiVotoMaggiore(70));
        try {
                assertEquals(stats.numeroAnalizzatori(), 8);
                System.out.println("OK!");
        } catch(Exception e) { System.out.println("NO"); }
    }

    @Test
    void testBonus()
    {
        System.out.print("-- testBonus: ");
        Optional<Analizzatore> bonus = Analizzatori.analizzatoreBonus();
        bonus.ifPresent(stats::add);
        System.out.print(stats.generaRapporti());
        System.out.println(" OK! [da verificare a mano]");
    }

    @Test
    void testGeneraRapportiBasic()
    {
        System.out.print("-- testGeneraRapportiBasic: ");
```

```
        stats.addAll(Analizzatori.allBasic());
        List<String> l = stats.generaRapporti()
                              .stream()
                              .map(Object::toString)
                              .map(StudstatsTest::toMapString)
                              .collect(Collectors.toList());
        List<String> l2 = List.of("{ANNI_DIPLOMA={2017=234, 2016=37, 2015=13,
2014=11, 2013=4, 2012=7, 2011=1, 2009=2, 2008=2, 2005=1, 2003=1, 2000=1}}",
                                  "{ISTITUTI={ALTRO=16, LABRIOLA=8, NETTUNO=7, MEUCCI=7, G.
PEANO - MONTEROTONDO=7, PACINOTTI=7, =5, MORGAGNI=5, \"I.T.I.G.S. \"\"L. DA
VINCI\"\"\"=5, AVOGADRO=5, ENRICO FERMI=4, NARNI LICEO SCIENTIFICO=4, PLINIO
SENIORE=4, A. VOLTA (TIVOLI)=4, VITO VOLTERRA=4, NEWTON=4, FERRARIS=4, II=3, LICEO
STATALE MARIA MONTESSORI=3, SANDRO PERTINI=3, RIGHI=3, E. FERMI=3, N. COPERNICO=3,
XX - ROMA VIA TEANO, 223=3, ARMELLINI=3, VITERBO=3, \"L.S. \"\"EVANGELISTA
TORRICELLI\"\"\"=3, TALETE=3, FRANCESCO D'ASSISI=3, ITI BARCELLONA COPERNICO=3,
\"LICEO SCIENTIFICO-CLASSICO-SOCIOPEDAGOGICO \"\"LEONARDO DA VINCI\"\"\"=3,
PASTEUR=3, \"I.T.I.S. \"\"G. MARCONI\"\"\"=3, IGNAZIO VIAN=3, ORAZIO=3, MAMIANI=2,
GIORDANO BRUNO=2, VIA P. NENNI,48=2, LICEO ROCCI=2, J.VON NEUMANN=2, VITTORIO
GASSMAN=2, BRUNO TOUSCHEK=2, VALLAURI=2, LIC.SCIENT. MAJORANA GUIDONIA=2,
PACINOTTI - ARCHIMEDE=2, DEMOCRITO=2, \"LICEO SCIENTIFICO STATALE \"\"F.
ENRIQUES\"\"\"=2, P. RUFFINI=2, LICEO STATALE ANTONIO MEUCCI=2, NOBEL=2,
MAJORANA=2, FARADAY=2, MARCONI=2, VIA DEI GIOCHI ISTMICI - ROMA=2, G.CABOTO=2,
G.DI VITTORIO=2, INNOCENZO XII=2, LICEO GAETANO DE SANCTIS=2, ROMA - VIA DELLA
VASCA NAVALE, 58=2, F.DE PINEDO=2, G.VALLAURI=2, LEON BATTISTA ALBERTI=2, PAOLO
BAFFI=2, BIAGIO PASCAL=2, CATTANEO CARLO=1, VITTORIA COLONNA=1, NOMENTANO=1,
S.MARIA=1, \"S.S. \"\"DANTE ALIGHIERI\"\" FIUGGI\"=1, LEONARDO MURIALDO=1,
M.IMMACOLATA=1, PIRANDELLO=1, LIVIA BOTTARDI=1, GIULIO CESARE=1, L.B.ALBERTI=1,
LICEO GINNASIO STATALE VIRGILIO=1, SERAPHICUM=1, CRISTOFORO COLOMBO=1, G.
FILANGIERI=1, ITC L.DE LIBERO=1, PIAZZA RISORGIMENTO, 1 SEGNI=1, \"ISTITUTO
TECNICO INDUSTRIALE \"\"XX\"\"\"=1, FEDERICO CAFFE'=1, BENEDETTO CROCE=1, PIERO
GOBETTI=1, LICEO LINGUISTICO F. HEGEL=1, VIA CASSIA, 931=1, \"\"\"GALILEO
GALILEI\"\" PONTECORVO\"=1, MATTEUCCI=1, LICEO SCIENTIFICO GHILARZA=1, I.I.S.
'PIETRO BONFANTE'=1, \"\"\"PRIMO LEVI\"\"\"=1, DE LORENZO=1, G.PIAZZI=1, LICEO
SCIENTIFICO STATALE XXIII=1, ETTORE MAJORANA=1, TULLIO LEVI-CIVITA D.P.R.=1,
COLONNA=1, PAOLO SEGNERI=1, IMMANUEL KANT=1, \"CONVITTO NAZIONALE  \"\"Vittorio
Emanuele II\"\"\"=1, AZZARITA=1, GALILEI - CIVITAVECCHIA=1, PANTALEONI=1, DANTE
ALIGHIERI=1, \"\"\"A. VOLTA\"\"\"=1, GEORGE BOOLE=1, A. VOLTA GUIDONIA=1, \"LICEO
CLASSICO \"\"LUCREZIO CARO\"\"\"=1, GIORGIO DE CHIRICO=1, EUGENIO MONTALE=1,
G.MARCONI=1, TERNI FEDERICO CESI=1, LIC. SCIENT. SAVIANO=1, G. GALILEI=1, A.
VOLTA=1, ARCHIMEDE=1, RONCIGLIONE=1, A. CALAMO - OSTUNI -=1, \"ISTITUTO TECNICO
COMMERCIALE \"\"ARANGIO RUIZ\"\"\"=1, LICEO CLASSICO - PRAIA A MARE=1, TERNI R.
DONATELLI=1, POLO LICEALE CISTERNA DI LATINA=1, A. LANDI=1, GIOVANNI PAOLO II=1,
ISTITUTO TECNICO COMMERCIALE 'XXIII'=1, LICEO SCIENTIFICO PARITARIO F. HEGEL=1,
LICEO SCIENTIFICO=1, LEONARDO DA VINCI=1, ITIS E LICEO SCIENT TECN  G  MARCONI=1,
BLAISE PASCAL=1, ISTITUTO TALETE=1, \"I.S.S. \"\"GIUSEPPE COLASANTI\"\" (MATURITA'
SCIENTIFICA)\"=1, \"\"\"GALILEI\"\" - TRENTO\"=1, \"LIC.CL.ANNESSO
CONV.NAZ.\"\"V.EMANUELE II\"\"\"=1, LIC. SCIENT. A INDIRIZZO LINGUISTICO=1, CARLO
JUCCI=1, VIVONA=1, \"L. C. NORCIA\"\" SEDE AGGR.\"=1, PIO IX=1, PEANO (ROMA)=1,
IPSAR VINCENZO GIOBERTI=1, LICEO SCIENTIFICO DI ZAGAROLO=1, GIORGI=1, LATINA=1,
\"LIC.SCIENT.\"\"ZALEUCO\"\" LOCRI\"=1, P.L. NERVI VALMONTONE=1, JOHANN VON
NEUMANN=1, LICEO SCIENT. VIA ALBERGOTTI=1, LAZZARO SPALLANZANI=1, CARLO E NELLO
```

```
ROSSELLI=1, ANIENE=1, GIOVANNI SULPICIO=1, \"I.T.COMMERCIALE \"\" L.PILLA\"\"\"=1,
GIOACCHINO PELLECCHIA=1, LUIGI SICILIANI=1, TERNI L. ALLIEVI=1, \"ITC STATALE
\"\"VINCENZO ARANGIO RUIZ\"\"\"=1, MALPIGHI=1, PACIFICI E DE MAGISTRIS=1, C.
BATTISTI=1, \"I.T.COMM.\"\"L.PACIOLI\"\" CASSANO I.\"=1, RENATO CARTESIO=1, VIA
SALVINI, 24=1, G. B. GRASSI=1, BERTRAND RUSSELL=1}}",
                        "{SESSO={F=26, M=288}}",
                        "{TITOLO={SCIENTIFICO=150, PERITO INDUSTRIALE CAPOTECNICO -
specializzazione INFORMATICA=68, CLASSICO=22, RAGIONIERE PERITO COMMERCIALE E
PROGRAMMATORE=13, TECNICO INDUSTRIALE (GENERICO)=7, LINGUISTICO=7, PERITO
INDUSTRIALE CAPOTECNICO - specializzazione ELETTRONICA E TELECOMUNICAZIONE=5,
TITOLO DI STUDIO STRANIERO=5, TECNICO (GENERICO)=5, PERITO INDUSTRIALE CAPOTECNICO
- specializzazione ELETTROTECNICA E AUTOMAZIONE=4, RAGIONIERE E PERITO
COMMERCIALE=4, TECNICO COMMERCIALE (GENERICO)=3, PERITO PER IL TURISMO=2, TECNICO
GESTIONE AZIENDALE INFORMATICA=2, GEOMETRA=2, NAUTICO (GENERICO)=1, SPERIMENTALE
GRAFICO VISIVO=1, LICEO ARTISTICO - DURATA QUINQUENNALE=1, AREONAUTICO
(GENERICO)=1, ASPIRANTE ALLA DIREZIONE DI MACCHINE DI NAVI MERCANTILI=1, MATURITA'
DI LICEO SOCIO-PSICO-PEDAGOGICO=1, PERITO AERONAUTICO SPECIALIZZATO: ASSISTENTE DI
NAVIGAZIONE AEREA=1, TECNICO CINEMATOGRAFIA E TELEVISIONE=1, TECNICO DELLE
INDUSTRIE ELETTRICHE  ED ELETTRONICHE=1, PROFESSIONALE (GENERICO)=1, TECNICO
SERVIZI SOCIALI=1, TECNICO DEI SERVIZI DELLA RISTORAZIONE=1, TECNICO CHIMICO E
BIOLOGICO=1, PERITO AZIENDALE E CORRISPONDENTE IN LINGUE ESTERE=1, SPERIMENTALE
MUSICALE=1}}",
                        "{VOTO={VOTO_MEDIO=76.81, VOTO_MAX=101, VOTO_MIN=60,
VOTO_MEDIANO=75}}")
                                        .stream()
                                        .map(StudstatsTest::toMapString)
                                        .collect(Collectors.toList());
        try {
                assertEquals(l, l2);
                System.out.println("OK!");
        } catch(Exception e) { System.out.println("NO"); }
    }

    @Test
    void testGeneraRapportiTipoSesso()
    {
        stats.addAll(Analizzatori.allBasic());

        System.out.print("-- testGeneraRapportiTipoSesso: ");
        Rapporto r = stats.generaRapporti(Analizzatori.sesso().getTipo())
                                        .get(0);

        try {
                assertEquals(toMap(r.toString()).toString(), "{SESSO={F=26,
M=288}}");
                System.out.println("OK!");
        } catch(Exception e) { System.out.println("NO"); }
    }

    @Test
    void testGeneraRapportiSegretoSegretissimo()
```

```
    {
        System.out.print("-- testGeneraRapportiSegretoSegretissimo: ");
        stats.addAll(Analizzatori.allBasic());
        stats.add(new AnalizzatoreSegretoSegretissimo());
        for (Rapporto r : stats.generaRapporti(Analizzatori.voto().getTipo(), new
AnalizzatoreSegretoSegretissimo().getTipo()))
            System.out.print(r+"; ");
        System.out.println(" OK! [da verificare a mano]");
    }

    static String toMapString(String mappa)
    {
        return toMap(mappa).toString();
    }

    static Map<String, Map<String, String>> toMap(String mappa)
    {
        Pattern re = Pattern.compile("([A-Z_]+) *= *\\{([^\\}]+)\\}");
        Matcher m = re.matcher(mappa);
        Pattern re2 = Pattern.compile("([a-zA-Z0-9-]+) *= *([^,]+)");

        Map<String, Map<String, String>> result = new TreeMap<>();

        while(m.find())
        {
            String key = m.group(1);
            String value = m.group(2);
            Matcher m2 = re2.matcher(value);
            Map<String, String> valueMap = new TreeMap<>();
            while(m2.find())
                valueMap.put(m2.group(1), m2.group(2));
            result.put(key, valueMap);
        }

        return result;
    }
}
```

```
/**
 * Interfaccia che istanzia gli analizzatori di base
 * Questa interfaccia è modificabile PURCHE' giri con il mio Junit
 * (e' possibile sostituire i metodi con una propria
 * implementazione lambda)
 *
 * @author navigli
 *
 */
```

```
public interface Analizzatori
{
    /* è possibile sostituire il codice con le lambda corrispondenti, ad es.:
return () -> { codice dell’analizzatore } */
    static Analizzatore annoDiploma() { return new AnalizzatoreAnnoDiploma(); }
    static Analizzatore istituti() { return new AnalizzatoreIstituti(); }
    static Analizzatore sesso() { return new AnalizzatoreSesso(); }
    static Analizzatore titoloDiStudio() {
                          return new AnalizzatoreTitoloDiStudio(); }
    static Analizzatore voto() { return new AnalizzatoreVoto(); }
    static Analizzatore studentiVotoMaggiore(int voto) {
                          return new AnalizzatoreStudentiVotoMaggiore(voto); }
    static Analizzatore studentiVotoMaggiore(int voto, Analizzatore a) {
                          return new AnalizzatoreStudentiVotoMaggiore(voto, a); }
    /* inserire l'eventuale codice dell'analizzatore bonus
       al posto di return Optional.empty(); si utilizzi Optional.of */
    static Optional<Analizzatore> analizzatoreBonus() { return Optional.empty(); }
    static Analizzatore[] allBasic() { return new Analizzatore[] {
                                                annoDiploma(), istituti(),
                                                sesso(), titoloDiStudio(),
                                                voto() }; }
}
```