```
/**
 * Classe di Test.
 *
 * @author Filippo Fontanelli , Francesca Brogi
 *
 */

public class testClassLezione {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        try {
            System.out
                    .println("\n================================\nProvo la classe
'Lezione'...\n================================");
            Lezione[] dip = new Lezione[5];
            System.out.println(".");
            // PROVO COSTRUTTORI
            try {
                // Caso Null
                dip[0] = new Lezione(null, "Lezione", "Argomento",
                        new RegStudenti(), 10);
                System.out
                        .println("Attento: il costrutture deve lanciare una
'NullPointerException' \nse la data della lezione e' null: correggi e riprova.");
            } catch (NullPointerException e) {
            }

            try {
                dip[0] = new Lezione(new Data(12, 05, 2008), null, "Argomento",
                        new RegStudenti(), 10);
                System.out
                        .println("Attento: il costrutture deve lanciare una
'NullPointerException' \nse il nome della lezione e' la stringa vuota: correggi e
riprova.");
            } catch (NullPointerException e) {
            }

            try {
                dip[0] = new Lezione(new Data(12, 05, 2008), "Lezione", null,
                        new RegStudenti(), 10);
                System.out
                        .println("Attento: il costrutture deve lanciare una
'NullPointerException' \nse l'argomento della lezione e' null: correggi e riprova.");
            } catch (NullPointerException e) {
            }

            try {
                dip[0] = new Lezione(new Data(12, 05, 2008), "Lezione",
                        "Argomento", null, 10);
                System.out
                        .println("Attento: il costrutture deve lanciare una
'NullPointerException' \nse il registro lezioni e' null: correggi e riprova.");
```

```
            } catch (NullPointerException e) {
            }
            // Caso Stringa vuota ed interi < 0
            try {
                dip[0] = new Lezione(new Data(12, 05, 2008), "", "Argomento",
                        new RegStudenti(), 10);
                System.out
                        .println("Attento: il costrutture deve lanciare una
'IllegalArgumentException' \nse il cognome dello studente e' la stringa vuota:
correggi e riprova.");
            } catch (IllegalArgumentException e) {
            }

            try {
                dip[0] = new Lezione(new Data(12, 05, 2008), "Lezione", "",
                        new RegStudenti(), 10);
                System.out
                        .println("Attento: il costrutture deve lanciare una
'IllegalArgumentException' \nse l'argomento e' la stringa vuota: correggi e riprova."
);
            } catch (IllegalArgumentException e) {
            }

            try {
                dip[0] = new Lezione(new Data(12, 05, 2008), "Lezione",
                        "Argomento", new RegStudenti(), -1);
                System.out
                        .println("Attento: il costrutture deve lanciare una
'IllegalArgumentException' \nse il numero presenze e' minore di 0: correggi e riprova
.");

            } catch (IllegalArgumentException e) {
            }

            System.out.println("..");

            // PROVO GETTERS
            dip[0] = new Lezione(new Data(12, 05, 2008), "Lezione",
                    "Argomento", new RegStudenti(), 10);
            if (!dip[0].getNome().equals("Lezione"))
                System.out
                        .println("Attento: il valore restituito da getNome()()\n non
corrisponde al valore passato al costruttore");
            if (!dip[0].getArgomento().equals("Argomento"))
                System.out
                        .println("Attento: il valore restituito da getArgomento()\n
non corrisponde al valore passato al costruttore");
            if (dip[0].getDate().compareTo((Data) new Data(12, 05, 2008)) != 0)
                System.out
                        .println("Attento: il valore restituito da getDate()\n non
corrisponde al valore passato al costruttore");
            if (dip[0].getNumPresenze() != 10)
                System.out
```

```
                        .println("Attento: il valore restituito da getNumPresenze()\n
non corrisponde al valore passato al costruttore");

            // PROVO COMPARETO
            dip[0] = new Lezione(new Data(12, 05, 2008), "Lezione",
                    "Argomento", new RegStudenti(), 1); //
            dip[1] = new Lezione(new Data(13, 05, 2008), "Lezione",
                    "Argomento", new RegStudenti(), 10);//
            dip[2] = new Lezione(new Data(14, 05, 2008), "Lezione",
                    "Argomento", new RegStudenti(), 10); // 14
            dip[3] = new Lezione(new Data(15, 05, 2008), "Lezione",
                    "Argomento", new RegStudenti(), 110); // 10
            dip[4] = new Lezione(new Data(15, 05, 2008), "Lezione",
                    "Argomento", new RegStudenti(), 10); // 10

            System.out.println("...");

            if (dip[1].compareTo(dip[2]) >= 0 || dip[2].compareTo(dip[1]) <= 0
                    || dip[0].compareTo(dip[1]) >= 0
                    || dip[1].compareTo(dip[0]) <= 0
                    || dip[2].compareTo(dip[3]) >= 0
                    || dip[3].compareTo(dip[2]) <= 0
                    || dip[3].compareTo(dip[4]) != 0)
                System.out
                        .println("Attento: sembra che compareTo() non confronti\n
correttamente gli Studenti");

            System.out.println("....");

            // PROVO EQUALS
            try {
                if (dip[0].equals(null) || dip[0].equals("ciao")
                        || dip[0].equals(dip[1]) || dip[0].equals(dip[2])
                        || dip[0].equals(dip[3]) || !dip[3].equals(dip[4]))
                    System.out
                            .println("Attento: sembra che equals() non confronti\n
correttamente gli account");
            } catch (RuntimeException e) {
                System.out
                        .println("Attento: ho provato il metodo equals, e ho
catturato una "
                                + e
                                + ".\n Equals non deve mai lanciare un'eccezione!!!
Controlla...");
            }

            System.out.println(".....");

            // PROVO addElemList
            try {
                dip[0].addElemList(null);
                System.out
                        .println("Attento: il metodo addElemList deve lanciare una
```

```
'IllegalArgumentException' \nse tento di inserire uno studente null: correggi e
riprova. ");

            } catch (NullPointerException e) {
            }
            Studente app = new Studente("Mara", "Rossi", 2, new RegLezioni(),
                    10);
            try {
                dip[0].addElemList(app);
                dip[0].addElemList(app);
                System.out
                        .println("Attento: il metodo addElemList deve lanciare una
'DuplicatedLezioniException' \nse tento di inserire uno studentee che e' gia presente
: correggi e riprova. ");
            } catch (DuplicatedLezioniException e) {
            }

            System.out.println("......");

            Studente app2 = new Studente("Mario", "Rossi", 25,
                    new RegLezioni(), 10);
            dip[0].addElemList(app2);
            if (dip[0].getList().getElemReg().indexOf((Studente) app2) < dip[0]
                    .getList().getElemReg().indexOf((Studente) app))
                System.out
                        .println("Attento: il metodo addElemList deve mantenere l
'elenco degli studenti ordinato : correggi e riprova. ");

            System.out.println(".......");

            // PROVO removeElemList
            try {
                dip[0].removeElemList(null);
                System.out
                        .println("Attento: il metodo removeElemList deve lanciare una
'IllegalArgumentException' \nse tento di rimuovere uno studente null: correggi e
riprova. ");

            } catch (NullPointerException e) {
            }
            Studente rem = new Studente("Mario", "Bianchi", 25,
                    new RegLezioni(), 10);
            try {
                dip[0].removeElemList(rem);
                System.out
                        .println("Attento: il metodo removeElemList deve lanciare una
'NullPointerException' \nse tento di rimuovere uno studente che non e' presente nell
'archivio : correggi e riprova. ");
            } catch (NotExistentStudentiException e) {
            }

            dip[0].addElemList(rem);
            int prima = dip[0].getNumPresenze();
```

```
            dip[0].removeElemList(rem);
            int dopo = dip[0].getNumPresenze();
            if (prima != (dopo + 1))
                System.out
                        .println("Attento: il metodo removeElemList deve diminuire il
numero delle presenze dopo ogni cancellazione : correggi e riprova. ");

        } catch (Exception ecc) {
            System.out
                    .println("\n OOPS!!! Stavo provando la tua classe 'Lezione', \nma
si e' verificata un'eccezione:");
            ecc.printStackTrace();
            System.exit(-1);
        }

        System.out.println("OK: non sono riuscito a trovare alcun errore.");
        System.out.println("===============================================");

    }
}
```