

### 3.2.5 Time series

In statistics, signal processing and financial mathematics, a time series is a sequence of data points  $\{X_0, X_1, \dots, X_n\}$ , measured typically at successive times  $\{t_0, t_1, \dots, t_n\}$  spaced at uniform time intervals such that  $t_i - t_{i-1} = \Delta t$ . For analysing time series some basic methods are necessary in order to extract meaningful statistics and other characteristics of the data.

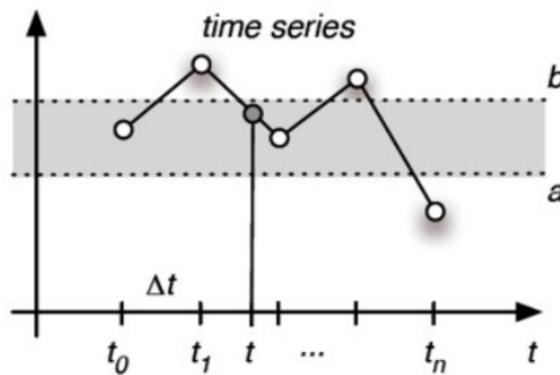
**getDuration** The duration of a time series can be calculated by  $t_n - t_0$ .

**getTime** Returns the point in time  $t_i$  of the  $i$ -th measurement.

**count** Counts the number of data points  $X_i$  lying inside a band between  $a$  and  $b$  such that  $a \leq X_i \leq b$ .

**valueAt** Returns the linearly interpolated data point for a specified point in time  $t$ .

**isMonotonous** Tests whether the time series is either ascending or descending. The series is ascending if  $X_i \leq X_{i+1}$  for  $i = 0, \dots, n-1$  and descending if  $X_i \geq X_{i+1}$  for  $i = 0, \dots, n-1$ .



#### Class design

Design a new class `TimeSeries`. The attributes are the array of data values,  $\Delta t$  and  $t_0$ . Specify a constructor that takes values for all attributes as arguments. Define the signatures of the above mentioned methods. Create the UML diagram of the `TimeSeries` class. Develop the Nassi-Schneiderman diagrams of the methods `valueAt` and `isMonotonous`.

#### Implementation

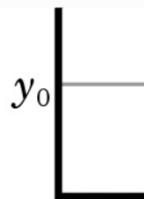
Implement the class `TimeSeries` in Java. Specify some useful test cases (test programs, test units) to verify the results of your implemented methods.

### Linear Interpolation (used formula):

If the two known points are given by the coordinates  $(x_0, y_0)$  and  $(x_1, y_1)$ , the **linear interpolant** is the straight line between these points. For a value  $x$  in the interval  $(x_0, x_1)$ , the value  $y$  along the straight line is given from the equation of slopes

$$\frac{y - y_0}{x - x_0} = \frac{y_1 - y_0}{x_1 - x_0},$$

which can be derived geometrically from the figure on the right. It is a special case of [polynomial interpolation](#) with  $n = 1$ .



Given the two I  
interpolant bet  
may be found I

# UML diagram

## TimeSeries

- elements : double[]
- t0: double
- = DELTA : double

- + TimeSeries (elements:double[], DELTA:double, t0:double)
- + getDuration () : double
- + getTime (i:int) : double
- + print() : void
- + count (a:double, b:double) : int
- + valueAt (t : double) : double
- + isMonotonic () : boolean

## Nassi - Shneiderman diagrams

## 1) valueAt method

valueA (+:double) : double

$$f_e = 1$$

$$Pe = 0$$

`n = this.elements.length`

~~t > this.getTime(n-1) OR  
t < this.getTime(0)~~

return 0

8

i=0; i<n; i++

$+ > \text{this.getTime}(i) \text{ AND}$   
 $+ < \text{this.getTime}(i+1) \text{ AND}$   
 $i != n-2$

$$fe = i+1$$

$t > \text{this.deltaTime}(i)$  AND  
 $t < \text{this.getTime}(i+1)$   
AND  $i \neq n-2$

T

三

$$f_e = n - 1$$

```

res = ((this.elements[fe] - this.elements[pe]) *
       (+- this.getTime(Pe))
     / (this.getTime(fe) - this.getTime(p))
     + this.elements[pe])

```

return res

# Nassi-Shneiderman diagram (isMonotonous method)

isMonotonous():boolean

check Asc = false

check Desc = false

n = this.elements.length

i=0; i<n-1; i++

this.elements[i]  $\geq$  this.elements[i+1]

T

F

check Asc = true

check Asc = false  
break

i=0; i<n-1; i++

this.elements[i]  $\leq$  this.elements[i+1]

T

F

check Desc = true

check Desc = false  
break

check Asc == true OR

check Desc == true

T

F

return true

return false

# My code

```
package chapter1;

public class TimeSeries {

    private double [] elements;
    private double DELTA;
    private double t0;

    //Constructor for the class TimeSeries. We assign parameters to the attributes of the object
    public TimeSeries (double [] x, double t0, double DELTA) {
        this.elements=x;
        this.DELTA = DELTA;
        this.t0 = t0;
    }

    //Method to calculate the duration of the series.
    public double getDuration () {
        int n = this.elements.length;
        //Duration is the period from time of the last element minus the time of the first element
        double duration = this.getTime(n-1) - this.getTime(0);
        System.out.println("\n"+ "Duration of the time series is:" + duration);
        return duration;
    }

    //Method that return the point in time of the i-th measurement.
    public double getTime (int i) {
        //We have to define the corresponding time values for each element in the data array
        int n = this.elements.length;
        double [] ts = new double [n+1];
        for (int j=0; j<n; j++) {
            ts[j] = this.t0 + this.DELTA*j; }
        //We just return the time value from the corresponding position
        return ts[i];    }

    //Method for printing the time series with their corresponding time values
    void print () {
        int n = this.elements.length;
        System.out.println("Defined timeseries is:");
        for ( int i=0; i<n; i++ ) {
            System.out.println("Element at the measurement number " + i + " has the time " + this.getTime(i) + " and it's value is " + this.elements[i]);}

    //Method that counts the numbers of data points between two values a and b
    public int Count (double a, double b) {
        int number = 0;
        //If one data point lies between this two values, the number increases by one
        for (int i=0; i<this.elements.length; i++) {
            if (this.elements[i] >=a && this.elements[i]<=b) {
                number = number + 1;
            }}
        System.out.println("\n"+ "Number of data points between values " + a + " and " + b + " is " + number);
        return number;
    }

    public double valueAt (double t) {
        int fe=1;
        int pe=0;
        int n = this.elements.length;

        //We check between which measurements does the inserted value lie
        if (t> this.getTime(n-1) || t<this.getTime(0)){
            System.out.println("Entered value of t is out of the limits of this time series.");
            return 0;
        } else {
            for (int i = 0; i<n-1; i++) {
                if ( t>=this.getTime(i) && t<=this.getTime(i+1) && i!=this.elements.length-2) {
                    fe = i+1; pe = i;
                }
                else if (t>=this.getTime(i) && t<=this.getTime(i+1) && i==this.elements.length-2) {
                    fe = this.elements.length-1;
                    pe = this.elements.length -2;    }}}
            //Printing this results for checking
            System.out.println("\n"+ "Entered value is between values of time " + this.getTime(pe) + " and " + this.getTime(fe) + " , which correspond to the
            + fe + "\n");
            //Calculating the value of the data point at the time t according to linear interpolation formulae
            double res;
            res = ((this.elements[fe]-this.elements[pe])*(t-this.getTime(pe)))/(this.getTime(fe)-this.getTime(pe))+this.elements[pe];
            System.out.println("Calculated value is " + res);
            return res;
    }
}
```

```

public boolean isMonotonous () {
    boolean checkAsc=false;
    boolean checkDesc=false;
    int n = this.elements.length;
    //For loop to check if our time series is ascending
    for (int i = 0; i<n-1; i++) {
        if (this.elements[i]>=this.elements[i+1]) {
            checkAsc=true;} else {
            checkAsc=false;
            break;}}
    //For loop to check if our time series is descending
    for (int i = 0; i<n-1; i++) {
        if (this.elements[i]<=this.elements[i+1]) {
            checkDesc=true;} else {
            checkDesc=false;
            break;}}
    //If our time series is either ascending or descending we say it is monotonous
    if (checkAsc==true || checkDesc==true) {
        System.out.println("Time series is montonous.");
        return true; }
    else {
        System.out.println("Time series is not montonous.");
        return false;}}
}

public static void main (String [] args) {

    //For loop to create a time series with random values
    double [] x = new double [10];
    for (int i = 0; i<10; i++) {
        x [i] = Math.random()*10;}
    TimeSeries ts = new TimeSeries (x, 1.0, 0.5);

    //For loop to create an ascending time series
    double [] x1= new double [10];
    for (int i = 0; i<10; i++) {
        x1 [i] = i;}
    TimeSeries tsA= new TimeSeries (x1, 1.0, 0.5);

    //For loop to create a descending time series
    double [] x2= new double [10];
    for (int i = 0; i<10; i++) {
        x2 [i] = (10-i);}
    TimeSeries tsD= new TimeSeries (x2, 1.0, 0.5);

    //Testing our methods
    ts.print();
    ts.getDuration();
    int measurement = 4;
    double f = ts.getTime(measurement);
    System.out.println("\n"+ "Value of time at the position " + measurement + " is " + f);
    ts.Count(2.4, 3.8);
    ts.valueAt(3.8);
    ts.isMonotonous();
    tsA.isMonotonous();
    tsD.isMonotonous();
}

```

# Console output

```
Problems @ Javadoc Declaration Console X Call Hierarchy
<terminated> TimeSeries [Java Application] C:\Users\User\Desktop\Programming\Eclipse\jre1.8.0_201\jre1.8.0_201\jre1.8.0_201\bin\javaw.exe (0)
Defined timeseries is:
Element at the measurement number 0 has the time 1.0 and it's value is 2.2643518780696956
Element at the measurement number 1 has the time 1.5 and it's value is 7.296225860740112
Element at the measurement number 2 has the time 2.0 and it's value is 7.703064323315382
Element at the measurement number 3 has the time 2.5 and it's value is 6.706116678757245
Element at the measurement number 4 has the time 3.0 and it's value is 9.055514492402404
Element at the measurement number 5 has the time 3.5 and it's value is 3.856874709895546
Element at the measurement number 6 has the time 4.0 and it's value is 2.397131378700851
Element at the measurement number 7 has the time 4.5 and it's value is 2.166423780017579
Element at the measurement number 8 has the time 5.0 and it's value is 2.8438788069386645
Element at the measurement number 9 has the time 5.5 and it's value is 7.672091516259794

Duration of the time series is:4.5

Value of time at the position 4 is 3.0

Number of data points between values 2.4 and 3.8 is 1

Entered value is between values of time 3.5 and 4.0 , which correspond to the number of measurement 5 and 6

Calculated value is 2.9810287111787295
Time series is not monotonous.
Time series is monotonous.
Time series is monotonous.
```