Programación en Lenguaje Java

Práctica 7.1. Paquetes. Señales movimiento ondulatorio

Michael González Harbour

Mario Aldea Rivas

Departamento de Matemáticas,
Estadística y Computación

Práctica 7-1: Paquetes. Señales movimiento ondulatorio

Objetivos

- Practicar la creación y uso de paquetes Java.
- Practicar la escritura y generación de la documentación de las clases.
- Practicar los recorridos en arrays.
- Practicar la escritura y uso de métodos estáticos.

Desarrollo

Tomando como base el documento de requisitos que aparece a continuación:

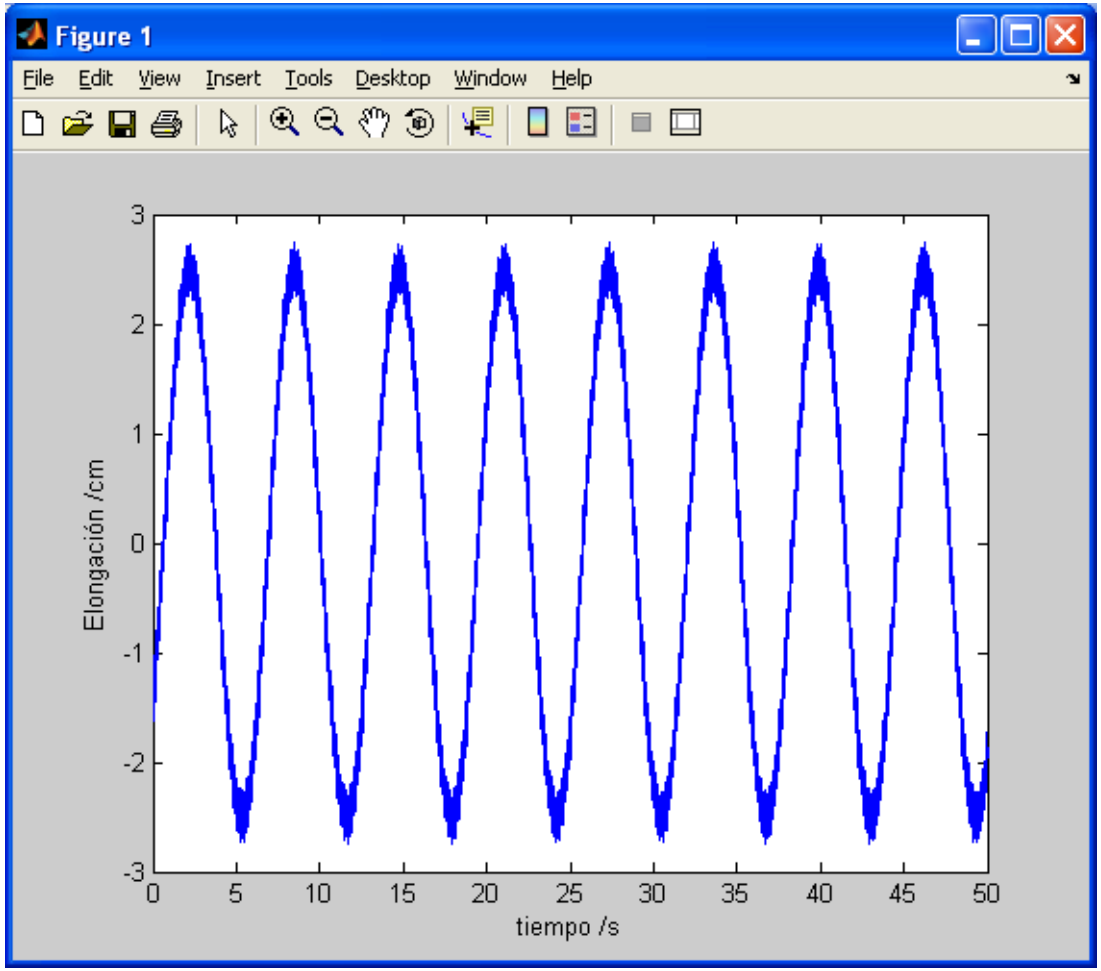
1. Finalizar la implementación del paquete `senhales`.
2. Generar la documentación de `javadoc` del paquete.
3. Encapsular el paquete en un `jar`.
4. Escribir un programa sencillo de prueba del paquete desarrollado.

Documento de requisitos

Un laboratorio de mecánica dispone de un instrumento electrónico capaz de monitorizar el movimiento vibratorio armónico simple de un cuerpo suspendido de un muelle.

Dicho instrumento genera una señal que muestra la elongación del muelle en función del tiempo pero, debido a que el dispositivo no es perfecto y que hay interferencias en el entorno, muestra una señal sinusoidal con ruido. El dispositivo toma 100 medidas por segundo y genera un conjunto de medidas conteniendo las lecturas correspondientes a un intervalo de tiempo determinado. La siguiente figura muestra un ejemplo de las lecturas obtenidas en un intervalo de 50 segundos

Ejemplo de lecturas



Requisitos (cont.)

Se quiere implementar el paquete Java `senhales` que obtenga y filtre la señal proveniente del instrumento de medida y calcule los parámetros que caracterizan el movimiento vibratorio.

Este paquete constará de 3 clases, que se exponen a continuación

Clase `Medida` (ya implementada)

Clase muy sencilla que contiene los dos atributos públicos de una medida (valor de la medida y el instante de tiempo en que se tomó).

Clase `Instrumento` (parcialmente implementada) (Alumno A)

Clase que representa el instrumento de medida del laboratorio. Contiene los siguientes métodos:

```
static public Medida[] generaSenal(double duracion)
```

- Retorna un array de medidas en el intervalo indicado. *(Ya está implementado)*.
- Parámetros: duracion- duración del intervalo durante el que se desean obtener las medidas.

Clase `Instrumento` (cont.)

```
static public Medida[] filtraSenal(Medida[] senal)
```

- Realiza el filtrado de las medidas, sustituyendo cada medida por la media de los valores anterior y posterior en el array. Los valores primero y último se dejarán como estaban.
- Parámetros:
  - senal: señal que se desea filtrar.
- Retorna: array de medidas filtrado.

Clase `Instrumento` (cont.)

```
static public Medida[] filtraSenal(Medida[] senal,
int nVeces)
```

- Hace nVeces llamadas al método `filtraSenal(Medida[] senal)` anterior para que el filtrado sea apreciable (por ejemplo, con nVeces=100).
- Parámetros:
  - senal: señal que se desea filtrar.
  - nVeces: número de veces que se invocará el método `filtraSenal(Medida[] senal)`.
- Retorna: array de medidas filtrado.

Clase **Parametros** (hacer por el alumno) (Alumno B)

Esta clase contiene métodos para calcular la amplitud, el periodo y la fase inicial de la forma de onda sinusoidal:

```
f(t)=Amplitud*sin(2*pi*t/periodo-fase)
```

```
static public double calculaAmplitud(Medida[] senal)
```

- Calcula y retorna la amplitud de la señal. Se calcula como la diferencia entre los valores máximo y mínimo dentro del array.
- Parámetros:
  - senal: señal de la que se desea calcular su amplitud.
- Retorna: amplitud de la señal.

Clase Parametros (cont.)

`static public double calculaPeriodo(Medida[] senal)`

- Calcula y retorna el periodo de la señal. Se calcula como la diferencia entre los dos primeros instantes de tiempo en que la señal pasa por el valor 0 con pendiente ascendente.
- Parámetros:
  - senal: señal de la que se desea calcular su periodo.
- Retorna: periodo de la señal.

`static public double calculaFase(Medida[] senal)`

- Calcula y retorna la fase inicial. Corresponde al instante del primer cruce por cero con pendiente ascendente dividido entre el `Periodo` y multiplicado por `2*PI`.
- Parámetros:
  - senal: señal de la que se desea calcular su fase.
- Retorna: fase de la señal.

Encapsular el proyecto

Encapsular el proyecto en un fichero jar (**Alumnos A y B**), para ello:

1. Botón derecho sobre el paquete `senhales -> Export -> Java -> JAR file`
2. Elegimos las clases que queremos añadir, damos nombre a la librería.
3. `Finish`.

Documentación

Generar la documentación del paquete (**Alumnos A y B**), para ello:

1. Elegir `Project -> Generate Javadoc.`
2. Elegir las clases de las que vamos a generar la documentación y el tipo de métodos y atributos que queremos que se incluyan (privados, protegidos, públicos) y pulsar `Finish`
3. La documentación se genera en un directorio `doc` dentro del proyecto.

Para consultar la documentación abrir el fichero `index.html` (puede abrirse desde el propio Eclipse pulsando con el botón derecho sobre `index.html` y eligiendo `Open With -> Web Browser`).

Prueba

Hacer un pequeño programa de prueba (**Alumnos A y B**) en otro proyecto que genere una señal, aplique el filtro 100 veces, dibuje las gráficas correspondientes antes y después de filtrar y, a continuación, llame a los métodos para calcular la amplitud el periodo y la fase inicial a partir de la señal filtrada y muestre los valores calculados.

Para dibujar las gráficas se debe usar la clase Gráfica del paquete fundamentos (ver como se usa en la página web de dicho paquete).

Material proporcionado

- Clases `Medida` e `Instrumento` (parcialmente implementada).

Entregar

- Código desarrollado, fichero `jar` y documentación generada por `javadoc` (a través del moodle).