EXAMEN DE PROGRAMACIÓN
GRADO EN INGENIERÍA INFORMÁTICA
Leganés, Enero de 2015

Universidad Carlos III de Madrid

LEA ATENTAMENTE ESTAS INSTRUCCIONES ANTES DE COMENZAR LA PRUEBA:

- **Rellene todas las hojas a bolígrafo, no utilice lápiz ni bolígrafo rojo**
- **El tiempo máximo de realización es de 3 horas**
- **Se permiten apuntes y/o libros para la realización del examen. No se permiten dispositivos electrónicos de ningún tipo**

Se quiere crear un programa que ayude al usuario a invertir en Bolsa, para ello se crearán una serie de clases. Cada ejercicio se puede realizar por separado, suponiendo que existen las clases creadas en el ejercicio anterior. De igual forma, cada método se puede realizar por separado, asumiendo que existen los métodos de los apartados anteriores.

Pregunta 1 (4,5 puntos).- Crear la clase `Acción` con las siguientes características:

a) **(0,3 puntos)** Atributos privados: `nombre`, `valor`, `valorAnterior`, que deben guardar respectivamente el nombre de la acción, su valor actual y el valor en la sesión anterior. Elegir el tipo más adecuado para estos atributos.

b) **(0,3 puntos)** Métodos `set` y `get` para cada uno de los atributos. Tanto `valor` como `valorAnterior` deben ser siempre mayores que cero.

c) **(0,4 puntos)** Un constructor completo que recibe valores para los tres atributos (debe usar los métodos `set` anteriores)

d) **(0,4 puntos)** Un constructor sin parámetros que use el constructor anterior y que cree una `Acción` de Inditex, con valor actual 22.72 y valor anterior 22.95.

e) **(0,4 puntos)** Un método `actualizar` que reciba el nuevo `valor` de la `Acción`, lo coloque en el atributo correspondiente y ponga el `valor` antiguo en `valorAnterior`. Si el valor recibido no es mayor o igual que cero, no hará nada.

f) **(1 punto)** Un método **privado** `truncar` que recibe un `double` y devuelve un `String` resultado de truncar el número recibido a 2 decimales. Asumir que el número tiene siempre más de 2 decimales. Ejemplo, recibe `2.14932423` y devuelve `"2.14"`. Debe funcionar para cualquier número.

g) **(0,8 puntos)** Un método `variación` que devuelva como `String` con dos decimales el porcentaje de variación de una acción de un día respecto al anterior.

h) **(0,5 puntos)** Un método `toString` que devuelva: `"<nombre de acción>;<valor>;<variación respecto al día anterior>%"` Por ejemplo, para la `Acción` creada por defecto devolvería: `"Inditex;22.72;-1.00%"`. Tanto `valor` como variación deben tener 2 decimales.

i) **(0,4 puntos)** Un método `equals` que recibe por parámetro otro objeto `Acción` y devuelve `true` si ambas son iguales y `false` en caso contrario. Dos acciones son iguales si tienen el mismo `nombre`.

```java
public class Accion {
	private String nombre;
	private double valor;
	private double valorAnterior;

	public String getNombre() {
		return nombre;
	}
	public void setNombre(String nombre) {
		this.nombre = nombre;
	}
	public double getValor() {
		return valor;
	}
	public void setValor(double valor) {
		if (valor>0)
			this.valor = valor;
	}
	public double getValorAnterior() {
		return valorAnterior;
	}
	public void setValorAnterior(double anterior) {
		if (anterior>0)
			this.valorAnterior = anterior;
	}
	public Accion (String n, double v, double a){
		setNombre(n);
		setValor(v);
		setValorAnterior(a);
	}
	public Accion (){
		this("Inditex",22.72,22.95);
	}
	public void actualizar (double nuevoValor){
		if (nuevoValor>0){
			setValorAnterior(valor);
			setValor(nuevoValor);
		}
	}
	//Este método supone que siempre va a haber al menos 2 decimales
	private String truncar (double num){
		//Convertimos a String
		String res = ""+num;
		/* La forma ortodoxa sería:
		 * String res = String.valueOf(num);
		*/
		//Buscamos la posición del punto
		int pos = res.indexOf(".");
		//Devolvemos la cadena hasta dos elementos más
		//como en substring el segundo índice está excluido
		//ponemos 3 en lugar de 2
		res = res.substring(0,pos+3);
		return res;
	}
	public String variacion (){
		//Calculamos la variación
		double var;
		var = (valor-valorAnterior)*100/valorAnterior;
		//y llamamos al método truncar
		return truncar(var);
	}
```

```
	public String toString (){
		String res;
		res = nombre+";"+truncar(valor)+";"+variacion()+"%";
		return res;
	}
	public boolean equals (Accion a){
		//Como no dice nada, asumimos que tienen que coincidir
		//mayúsculas y minúsculas. También podríamos haber usado
		//equalsIgnoreCase
		return a.nombre.equals(nombre);
	}
}
```

Pregunta 2 (4 puntos).- Crear la clase `Bolsa` con las siguientes características:

a) **(0,2 puntos)** Atributo privado: `cotizaciones`, de tipo array de `Acción`.

b) **(0,2 puntos)** Un método `setCotizaciones` y un método `getCotizaciones`.

c) **(1 punto)** Un método `partirString` que recibe un `String` como parámetro. El parámetro tendrá siempre un formato como "`Inditex;22.72;22.95 Telecinco;10.02;11.02`" (es decir, los `nombres` de cada `Acción`, su `valor` actual y su valor anterior, separando cada `Acción` por un espacio). Devuelve una matriz de `String` en el que hay tantas filas como acciones en el array y cada fila tiene 3 columnas, la primera es el `nombre`, la segunda el `valor` y la tercera el valor anterior. El método debe funcionar para cualquier número de acciones dentro del `String`.

d) **(1 punto)** Un método `creaAcciones` que recibe un `String` como el del método anterior y devuelve un array con los objetos `Acción` correspondientes

e) **(0,4 puntos)** Un constructor que recibe un `String` como el de los dos métodos anteriores y crea el array `cotizaciones` con el contenido de ese `String`.

f) **(0,5 puntos)** Un constructor sin parámetros que crea un array de 10 posiciones para `cotizaciones` y crea sus elementos usando el constructor sin parámetros de `Acción`.

g) **(0,5 puntos)** Un método `buscar` que reciba un `String` con el `nombre` de una `Acción` y si la `Acción` está en `cotizaciones` imprima el `nombre`, `valor` y variación respecto al día anterior de esa `Acción`. Debe devolver también la posición en la que se encuentra esa `Acción` en el array o -1 si no está.

h) **(0,2 puntos)** Un método `toString` que devuelva el `nombre`, `valor` y variación con el día anterior de cada elemento de `cotizaciones`, cada uno en una línea.

```
import java.util.ArrayList;

public class Bolsa {
	private Accion [] cotizaciones;

	public Accion []  getCotizaciones () {
		return cotizaciones;
	}
	public void setCotizaciones (Accion [] c) {
		//Esto no es muy ortodoxo porque c y cotizaciones ahora
		//son el mismo array, pero para el examen no se pedía más.
		cotizaciones = c;
		//Lo suyo es copiar los elementos de uno en otro:
			//cotizaciones = new Accion[c.length];
			//System.arraycopy(c, 0, cotizaciones, 0, c.length);
	}
```

```
	public String [][] partirString (String acc) {
		String [] [] res;
		//Primero partimos por el espacio
		String [] aux = acc.split(" ");
		//Con esto sabemos el número de filas de la matriz
		res = new String[aux.length][];
		//Ahora partiremos por el ;
		for (int ii=0; ii<aux.length;ii++){
			res[ii] = aux[ii].split(";");
		}
		return res;
	}
	public Accion [] creaAcciones (String acc){
		//Partimos el String
		String [][] aux = partirString(acc);
		//Creamos un array de Accion con tantos elementos como
		//el número de filas de la matriz
		Accion [] array = new Accion[aux.length];
		//Ahora crearemos cada objeto del array de Accion
		for (int ii=0; ii<aux.length;ii++){
			String nombre = aux[ii][0];
			double valor = Double.parseDouble(aux[ii][1]);
			double anterior = Double.parseDouble(aux[ii][2]);
			//Creamos el objeto
			array[ii] = new Accion(nombre,valor,anterior);
			}
		return array;
	}
	public Bolsa (String acc){
		cotizaciones = creaAcciones(acc);
	}

	public Bolsa () {
		cotizaciones = new Accion[10];
		for (int ii=0;ii<cotizaciones.length;ii++){
			cotizaciones[ii] = new Accion();
		}
	}
	public int buscar (String nombre){
		for (int ii=0; ii<cotizaciones.length; ii++){
			if (cotizaciones[ii].getNombre().equals(nombre)){
				System.out.println(cotizaciones[ii]);
				return ii;
			}
		}
		//Si ha llegado hasta aquí es porque no la
		//ha encontrado
		return -1;
	}

	public String toString (){
		String res="";
		for (int ii=0; ii<cotizaciones.length;ii++)
			res = res + cotizaciones[ii]+"\r\n";
		return res;
	}
}
```

Pregunta 3 (1,5 puntos).- Crear una clase `Prueba` con las siguientes características:

a) **(0,5 puntos)** Un método `ordenar`, que reciba un array de `Acción` y utilizando el algoritmo de la **burbuja** lo ordene de forma que las acciones de menor `valor` sean las primeras.

b) Un método `main` que haga lo siguiente:

   a) **(0,5 puntos)** Pedir al usuario que por teclado introduzca los datos de una lista de acciones en el formato: `nombre;valor;valor anterior`. Para terminar debe teclear la palabra "fin" (será válida cualquier combinación de mayúsculas o minúsculas). Guardar las acciones en un `String` separándolas por un espacio.

   b) **(0,2 puntos)** Crear un objeto `Bolsa` usando el `String` anterior.

   c) **(0,2 puntos)** Usar el método `ordenar` para ordenar las acciones del atributo `cotizaciones` del objeto `Bolsa` creado.

   d) **(0,1 puntos)** Imprimir el objeto `Bolsa`.

Ejemplo de ejecución:

```
Introduce las acciones. Introduce "fin" para terminar
Inditex;22.72;22.95
Telecinco;10.02;11.02
Antena3;12.12;12.04
FiN
Telecinco;10.02;-9.07%
Antena3;12.12;0.66%
Inditex;22.72;-1.00%
```

```
import java.util.Scanner;

public class Prueba {

	//Usamos el método de la burbuja más simple de los dos que hay en
	//Aula Global
	public static void ordenar (Accion [] lista){
		Accion aux;
		//bucle exterior (elementos-1) pasadas como máximo)
		for (int i=1; i<lista.length; i++){
			//Bucle interior (n-i comparaciones)
			for (int j=0; j<lista.length-i; j++){
				// si el elemento de índice inferior es mayor que el de índice
				// superior cambiamos
				if (lista[j].getValor()>lista[j+1].getValor()){
					//hacemos el cambio usando la variable auxiliar
					aux = lista[j+1];
					lista[j+1] = lista[j];
					lista[j] = aux;
				}//fin if
			}//fin for j
		}//fin for i
	}// fin burbuja

	public static void main(String[] args) {
		Scanner sc = new Scanner(System.in);
		String acc="";
		String respuesta;
		System.out.println("Introduce las acciones. Introduce \"fin\" para terminar");
```

```
		do {
			respuesta = sc.next();
			if (!respuesta.toLowerCase().equals("fin")) {
				acc = acc+respuesta+" ";
			}
		}
		while (!respuesta.toLowerCase().equals("fin"));
		//Creamos el objeto
		//Usamos trim() para quitar el último espacio
		Bolsa b = new Bolsa(acc.trim());
		ordenar(b.getCotizaciones());
		System.out.println(b);
		sc.close();
	}
}
```