Ejercicio 1 a)

Definición de la clase :

```
class Registro {
		boolean conc;
		int[] v;

		public Registro(boolean conc, int[] v) {
			this.conc = conc;
			this.v = v;
		}
```

Una solución para el método pedido:

```
public static Registro determinarConcentrica(int[][] m) {
		// PRE: matriz cuadrada
		// Por capas

		int nCapas = (m.length + 1) / 2;
		int[] v = new int[nCapas];
		boolean esConcentrica = true;
		// for capa de 0 a nCapas-1
		int capa = 0;
		while (capa < nCapas && esConcentrica) {
			if (! esCapa (m, capa)) esConcentrica = false;
			capa++;
		}
		if (esConcentrica)
			for (int capa = 0; capa < nCapas; capa++)
				v[capa] = m[capa][capa];
		else
			for (int i = 0; i < v.length; i++) v[i] = 0;
		Registro r = new Registro(esConcentrica, v);
		return r;
	}

public static boolean esCapa(int[][] m, int capa) {
	// DEVUELVE true o false según la "capa" de m sea constante
	// PRE: m cuadrada, capa <= m.length/2
	// la capa c de una matriz nxn se caracteriza por
	//		filas c y n-1-c entre las columnas c y n-1-c
	//		columnas c y n-1-c entre las filas c y n-1-c
	int modelo = m[capa][capa]; // Casilla Noroeste de la capa
	int n = m.length;
	boolean esCapa = true;
	// filas superior e inferior (i=c, i=n-1-c), (incluido el propio modelo)
	// for (int j = capa; j <= n-1-capa; j++)
	int j = capa;
	while ( j <= n-1-capa && esCapa ) {
		if (m[capa][j] != modelo)  esCapa = false;
		if (m[n-1-capa][j] != modelo)  esCapa = false;
		j++;
	}
	// columnas izquierda y derecha (j=c, j=n-1-c), evitando los extremos
	// for (int i=capa+1; i < n-1-capa; i++) {
	int i = capa + 1;
	while ( i<n-1-capa  && esCapa; ) {
		if (m[i][capa] != modelo) esCapa = false;
		if (m[i][n-1-capa] != modelo) esCapa = false;
		i++;
	}
	return esCapa;
}
```

Otra solución

```
public static Registro determinarConcentrica(int[][] m) {
            int nCapas = (m.length + 1) / 2;
            int[] v = new int[nCapas];

            boolean esConcentrica = true;
            int n = m.length;

            int capa = 0 ;
            while (capa < nCapas && esConcentrica){
                  // for  capa de 0 a nCapas-1
                  int modelo = m[capa][capa]; // Casilla Noroeste de la capa
                  v[capa] = modelo;
                  // filas superior e inferior (i=c, i=n-1-c).
                  //Se comprueba de más la casilla modelo
                  int j = capa;
                  while (j <= n - 1 - capa && esConcentrica){
                  // for j de capa a n-1-capa
                        if (m[capa][j] != modelo) esConcentrica = false;
                        if (m[n-1-capa][j] != modelo) esConcentrica = false;
                        j++;
                  }
                  // columnas izquierda y derecha (j=c, j=n-1-c) ,
                  // evitando los extremos ya comprobados
                  int i = capa+1;
                  while (i < n - 1 - capa && esConcentrica) {
                  // for i de capa+1 a n-1-capa-1
                        if (m[i][capa] != modelo) esConcentrica = false;
                        if (m[i][n-1-capa] != modelo) esConcentrica = false;
                        i++;
                  }
                  capa ++;
            }
            if (!esConcentrica)
                  for (int i = 0; i < v.length; i++) v[i] = 0;

            Registro r = new Registro(esConcentrica, v);
            return r;

      }
```

Ejercicio 1 b)

```
public static boolean esCreciente(Registro r) {
      // Devuelve True o False según el vector de r
      // esté compuesto por valores estrictamente crecientes o no
      boolean vamosBien = true;
      int i = 1; // ¡Atención a los límites!
      while (i<r.v.length && vamosBien) {
            if (r.v[i] <= r.v[i-1]) vamosBien = false;
            i++;
      }
      return vamosBien;
}
```

Ejercicio 2 a)

```
public static int valorCadena(String cadena) {
	// PRE: cadena en minúsculas, con otros símbolos que no cuentan
	int  puntos = 0;
	for (int i = 0; i<cadena.length(); i++){
		char c = cadena.charAt(i);
		if (c>='a' && c<= 'z') puntos += (c-'a'+1);
	}

	return puntos;
}
```

Ejercicio 2 b)

```
public static void main(String[] args) {
	final int VALORFIJO=351;
	// Por palabras
	try {
		Scanner texto = new Scanner (new File("Texto.txt"));
		boolean encontrado = false;
		String resultado = "";
		while (texto.hasNext() && ! encontrado) {
			String palabra = texto.next();
			int i = 0;
			while ( valorCadena(resultado)<VALORFIJO
				&& i<palabra.length()) {
				resultado += palabra.charAt(i);
				i++;
			}
			encontrado = valorCadena(resultado)>=VALORFIJO;
			resultado += ' ';  // ESTO mejor no tenerlo en cuenta
		}
		if (encontrado) System.out.println ("...  " + resultado);
		else 		System.out.println ("No lo he encontrado");
		texto.close();
	} catch (FileNotFoundException e) {
		System.out.println ("No pude abrir el archivo");
	}
}
```

Otra solución

```
public static void main(String[] args) {
	final int VALORFIJO=351;
	try {
		Scanner texto = new Scanner (new File("Texto.txt"));
		boolean encontrado = false;
		String resultado = "";
		while (texto.hasNextLine() && ! encontrado) {
			String linea = texto.nextLine();
			int i = 0;
			while (valorCadena(resultado)< VALORFIJO
				&& i<linea.length()) {
				resultado += linea.charAt(i);
				i++;
			}
			encontrado = valorCadena(resultado)>=VALORFIJO;
			resultado +=' ';  // ESTO  no es indispensable
		}
		if (encontrado)   System.out.println ("... " + resultado);
		else              System.out.println ("No lo he encontrado");
		texto.close();
	} catch (FileNotFoundException e) {
		System.out.println ("No pude abrir el archivo");
	}
}
```

Otra solución más (algo más eficiente)

```
try {
		Scanner texto = new Scanner (new File("Texto.txt"));
		boolean encontrado = false;
		String resultado = "";
		int valorLinea=0, valor = 0;
		while (texto.hasNextLine() && ! encontrado) {
			String linea = texto.nextLine();
			valorLinea = valorCadena(linea);
			if (valor+valorLinea >= VALORFIJO) {
				// encontrado
				encontrado = true;
				int i = 0;
				while (valor < VALORFIJO) {
					valor += valorCadena(linea.substring(i, i+1));
					resultado += linea.charAt(i);
					i++;
				}
			} else {
				resultado += linea;
				valor += valorLinea;
			}

			resultado +=' ';  // ESTO no es indispensable
		}
		if (encontrado)        System.out.println ("...: " + resultado);
		else                   System.out.println ("No lo he encontrado");
		texto.close();
	} catch (FileNotFoundException e) {
		System.out.println ("No pude abrir el archivo");
	}
```

Ejercicio 3)

```
public static double potencia(double a, int b) {
	// PRE : b>=0
	if (b == 0)
		return 1;
	else {
		double intermedio = potencia(a, b / 2);
		if (b % 2 == 0)
			return intermedio * intermedio;
		else
			return a * intermedio * intermedio;
	}
}
```

Otra solución

```
public static double potencia_b(double a, int b) {
	/// PRE : b>=0
	if (b == 0)    return 1;
	else if (b==2) return a*a;
	else if (b % 2 == 0)
		return (potencia (potencia (a, b/2), 2));
	else
		return a*(potencia (potencia (a, b/2), 2));
	}
```