Módulo: Entornos de Desarrollo

Unidad 4 - Optimización y documentación

Mejoras en código aplicando refactorización

Descripción:

Durante esta unidad se han estudiado los distintos patrones de refactorización, que permiten transformar el código desarrollado en nuestra aplicación sin alterar la funcionalidad que ofrece dicho código, para hacerlo más legible, con una mejor organización y diseño y para que goce de un mejor mantenimiento futuro. Además de ver las buenas prácticas que hacen de mejor calidad nuestro código.

Para practicar estos conceptos se propone refactorizar y aplicar buenas prácticas de programación a la clase Fecha siguiente:

```
public class Fecha {
private int dia;
private int mes;
private int anio;
public Fecha(int dia, int mes, int anio) {
this.dia = dia;
this.mes = mes;
this.anio = anio;
}
public boolean valida () {
if (dia < 1 || dia > 31)
return false;
if (mes < 1 || mes > 12)
return false;
// determinamos la cantidad de días del mes:
int diasMes = 0;
switch (mes) {
case 1:
case 3:
case 5:
case 7:
case 8:
case 10:
case 12: diasMes = 31; break;
case 4:
case 6:
case 9:
```

```
case 11 : diasMes = 30; break;
case 2 : // verificación de año bisiesto
if ( (anio % 400 == 0) ||
( (anio % 4 == 0) && (anio % 100 != 0) ) )
diasMes = 29;
else diasMes = 28;
break;
}
if (dia > diasMes)
return false;
else return true;
}
}
```

Objetivos:

- Aplicar buenas prácticas de programación al código.
- Elaborar las pruebas asociadas a la refactorización.
- Aplicar patrones de refactorización con las herramientas que proporciona el entorno de desarrollo.

Recursos:

- Acceso a internet.
- NetBeans.

Resolución:

- Primero vamos a identar (o tabular) el código para poder entenderlo mejor:

```
public class Fecha {
	private int dia;
	private int mes;
	private int anio;

	public Fecha(int dia, int mes, int anio) {
		this.dia = dia;
		this.mes = mes;
		this.anio = anio;
	}
	public boolean valida () {
		if (dia < 1 || dia > 31)
			return false;
		if (mes < 1 || mes > 12)
			return false;
		// determinamos la cantidad de días del mes:
		int diasMes = 0;
		switch (mes) {
			case 1:
			case 3:
			case 5:
			case 7:
			case 8:
			case 10:
			case 12:
				diasMes = 31; break;
			case 4:
			case 6:
			case 9:
			case 11 :
				diasMes = 30; break;
			case 2 : // verificación de año bisiesto
				if ( (anio % 400 == 0) || ( (anio % 4 == 0) && (anio % 100 != 0) ) )
					diasMes = 29;
				else
					diasMes = 28;
				break;
			}
		if (dia > diasMes)
			return false;
		else
			return true;
	}
}
```

- Ahora que lo podemos leer con más claridad vemos que el método valida() es poco legible y difícil de seguir. Deberíamos separar en un método aparte el cálculo de la cantidad de días del mes.

```
public class Fecha {
	private int dia;
	private int mes;
	private int anio;

	public Fecha(int dia, int mes, int anio) {
		this.dia = dia;
		this.mes = mes;
		this.anio = anio;
	}
	public boolean valida () {
		if (dia < 1 || dia > 31)
			return false;
		if (mes < 1 || mes > 12)
			return false;
		if (dia > diasMes())
			return false;
		else
			return true;
	}

	// determinamos la cantidad de días del mes:
	private int diasMes () {
		int diasMes = 0;
		switch (mes) {
			case 1:
			case 3:
			case 5:
			case 7:
			case 8:
			case 10:
			case 12:
				diasMes = 31; break;
			case 4:
			case 6:
			case 9:
			case 11 :
				diasMes = 30; break;
			case 2 : // verificación de año bisiesto
				if ( (anio % 400 == 0) || ( (anio % 4 == 0) && (anio % 100 != 0) ) )
```

```
                                diasMes = 29;
                        else
                                diasMes = 28;
                        break;
                }
                return diasMes;
        }
}
```

- Si seguimos desgranándolo vemos que lo podemos leer con más claridad pero que la parte del año bisiesto no esta muy clara. Así que hagamos un método llamado bisiesto().

```
public class Fecha {
        private int dia;
        private int mes;
        private int anio;

        public Fecha(int dia, int mes, int anio) {
                this.dia = dia;
                this.mes = mes;
                this.anio = anio;
        }
        public boolean valida () {
                if (dia < 1 || dia > 31)
                        return false;
                if (mes < 1 || mes > 12)
                        return false;
                if (dia > diasMes())
                        return false;
                else
                        return true;
        }

        // determinamos la cantidad de días del mes:
        private int diasMes () {
                int diasMes = 0;
                switch (mes) {
                        case 1:
                        case 3:
                        case 5:
                        case 7:
                        case 8:
                        case 10:
                        case 12:
                                diasMes = 31; break;
                        case 4:
```

```
                    case 6:
                    case 9:
                    case 11 :
                            diasMes = 30; break;
                    case 2 : // verificación de año bisiesto
                            if (bisiesto() )
                                    diasMes = 29;
                            else
                                    diasMes = 28;
                            break;
                    }
                    return diasMes;
        }

        private boolean bisiesto() {
                if ( (anio % 400 == 0) || ( (anio % 4 == 0) && (anio % 100 != 0) ) )
                    return true;
                else
                    return false;
        }

}
```

- Si esto lo hago por medio de un entorno de desarrollo me va a facilitar mucho el poder llevar a cabo este proceso.