Material para
a Formación Profesional inicial

A02. Clases e Obxectos

Familia profesional	IFC	Informática e comunicacións
Ciclo formativo	CSIFC02 CSIFC03	Desenvolvemento de aplicacións multiplataforma Desenvolvemento de aplicacións web
Grao		Superior
Módulo profesional	MP0485	Programación
Unidade didáctica	UD03	Introdución á POO
Actividade	**A02**	**Clases e obxectos**
Autores		Silvia Framiñán Fondevila Marta Rey López
Nome do arquivo		CSIF02_MP0485_V000302_UD03_A02_Clases_e_obxectos.odt

Índice

1. Ficha técnica

Contexto da actividade

Módulo	Duración	Unidade didáctica.	Sesións 60´	Actividades	Sesións 60´
MP0485. Programación.	**240**	UD01. Identificación dos elementos dun programa informático.	24	A01. Metodoloxía da programación.	5
				A02. Introdución á linguaxe de programación Java.	7
				A03. Elementos básicos dun programa Java.	12
		UD02. Uso de estruturas de control.	20	A01. Estruturas de control selectivas.	8
				A02. Estruturas de control repetitivas.	12
		UD03. Introdución á POO.	**28**	A01. Conceptos de POO.	8
				A02. Clases e obxectos.	**20**
		U04. Conceptos avanzados de POO.	20	A01. Obxectos predefinidos e métodos estáticos.	10
				A02. Constantes e métodos estáticos.	5
				A03. Visibilidade e empaquetaxe.	5
		UD05. Lectura e escritura de información.	34	A01. Fluxos de entrada e saída.	4
				A02. Uso de ficheiros.	15
				A03. Interfaces gráficas de usuario.	15
		UD06. Aplicación das estruturas de almacenamento.	37	A01. Uso de cadeas.	8
				A02. Uso de arrais.	9
				A03.Uso de coleccións e outras estruturas	12
				A04. Manipulación de documentos XML.	8
		UD07. Xerarquías de clases e excepcións.	37	A01. Introdución á herdanza.	10
				A02. Uso avanzado da herdanza.	10
				A03. Interfaces.	10
				A04. Control de código. Excepcións.	7
		UD08. Mantemento da persistencia dos obxectos.	18	A01. Instalación dun SXBDOO e almacenamento básico de obxectos.	8
				A02. Almacenamento e recuperación de obxectos nun SXBDOO e operacións con datos complexos.	10
		UD09. Xestión dos datos almacenados nas bases de datos relacionais.	22	A01. Conexión con sistemas xestores de bases de datos relacionais.	6
				A02. Operacións de lectura nunha base de datos relacional.	7
				A03. Operacións de escritura nunha base de datos relacional e uso de transaccións.	9

NOTA: Esta actividade está vinculada á programación recollida no arquivo CSIFC02_MP0485_V0003 00_UD03_Introducion_POO.pdf

Título da actividade

Nº	Título	Descrición	Duración
A02	Clases e obxectos.	Nesta actividade implementaranse clases e instanciaranse obxectos desas clases. Empregaranse os construtores dos obxectos e realizaranse chamadas aos seus métodos. Ademais tomarase contacto co contorno integrado de desenvolvemento para a implementación e proba do código.	20

Resultados de aprendizaxe do currículo

Resultados de aprendizaxe do currículo	Completo
▪ RA2: Escribe e proba programas sinxelos, para o que recoñece e aplica os fundamentos da programación orientada a obxectos.	**NON**
▪ RA4. Desenvolve programas organizados en clases, para o que analiza e aplica os principios da programación orientada a obxectos.	**NON**

Obxectivos didácticos e título e descrición da actividade

Obxectivos específicos		Actividade		Descrición básica	Duración
O2.1	Escribir programas simples.	A02	Clases e obxectos.	Nesta actividade implementaranse clases e instanciaranse obxectos desas clases. Empregaranse os construtores dos obxectos e realizaranse chamadas aos seus métodos. Ademais tomarase contacto co contorno integrado de desenvolvemento para a implementación e proba do código.	20
O2.2	Utilizar métodos e propiedades dos obxectos.				
O2.3	Utilizar parámetros na chamada a métodos.				
O2.4	Utilizar construtores.				
O2.5	Utilizar o contorno integrado de desenvolvemento na creación e na compilación de programas simples.				
O2.6	Recoñecer a sintaxe, a estrutura e os compoñentes típicos dunha clase.				
O2.7	Definir clases.				
O2.8	Definir propiedades e métodos.				
O2.9	Definir construtores.				
O2.10	Desenvolver programas que instancien e utilicen obxectos das clases creadas anteriormente.				

Criterios de avaliación

Criterios de avaliación
▪ CA2.2. Escribíronse programas simples.
▪ CA2.4. Utilizáronse métodos e propiedades dos obxectos.
▪ CA2.6. Utilizáronse parámetros na chamada a métodos.
▪ CA2.8. Utilizáronse construtores.
▪ CA2.9. Utilizouse o contorno integrado de desenvolvemento na creación e na compilación de programas simples.
▪ CA4.1. Recoñeceuse a sintaxe, a estrutura e os compoñentes típicos dunha clase.
▪ CA4.2. Definíronse clases.
▪ CA4.3. Definíronse propiedades e métodos.
▪ CA4.4. Definíronse construtores.
▪ CA4.5. Desenvolvéronse programas que instancien e utilicen obxectos das clases creadas anteriormente.

Contidos

Contidos
▪ BC2. Uso de obxectos: – Instanciación de obxectos. – Argumentos dun método. Valores devoltos. – Chamada aos métodos: mensaxes. Operador punto. – Identificador de obxecto actual. – Construtores. – Destrución de obxectos e liberación de memoria. ▪ BC4. Desenvolvemento de clases: – Concepto de clase. – Estrutura e membros dunha clase. – Tipos de atributos, métodos e construtores.

Actividades de ensino e aprendizaxe e de avaliación, métodos, recursos e instrumentos de avaliación (exemplo)

Que e para que	Como			Con que	Como e con que se valora	
Actividade (título e descrición)	**Profesorado (en termos de tarefas)**	**Alumnado (tarefas)**	**Resultados ou produtos**	**Recursos**	**Instrumentos e procedementos de avaliación**	
A02. Clases e obxectos. ▪ Nesta actividade implementaranse clases e instanciaranse obxectos desas clases. Empregaranse os construtores dos obxectos e realizaranse chamadas aos seus métodos. Ademais tomarase contacto co contorno integrado de desenvolvemento para a implementación e proba do código.	▪ **Tp2.1** - Explicación da creación de clases en Java. ▪ **Tp2.2** - Exposición dos construtores e a instanciación de obxectos en Java.	▪ **Ta2.1** - Creación e instanciación da clase Conta Bancaria. ▪ **Ta2.2** - Creación e emprego do construtor para a clase Conta Bancaria	▪ Proxecto Java.	▪ Ordenador persoal con conexión a Internet. ▪ Contorno de desenvolvemento NetBeans. ▪ Java SE Development Kit. ▪ Apuntamentos da profesora. ▪ Proxector.		16
	▪ **Tp2.3** - Descrición da creación e emprego de métodos nas clases. ▪ **Tp2.4** - Exposición dos métodos *getters* e *setters*.	▪ **Ta2.3** - Creación de métodos getters e setters para a clase Conta Bancaria. ▪ **Ta2.4** - Creación de métodos para a clase Conta Bancaria. ▪ **Ta2.5** - Creación da clase Persoa.	▪ Proxecto Java.			
	▪ **Tp2.5** - Explicación da recolección de lixo e do método `finalize`.	▪ **Ta2.6** - Toma de apuntamentos e formulación de dúbidas.	▪ Apuntamentos.			
		▪ **Ta2.7** - Tarefa de avaliación, combinando: – Táboa de indicadores sobre un proxecto Java onde se escriben programas simples. – Táboa de indicadores sobre un proxecto Java onde se utilizan métodos e propiedades de obxectos. – Táboa de indicadores sobre un proxecto Java onde se empregan parámetros na chamada a métodos. – Táboa de indicadores sobre un proxecto Java onde se empregan construtores. – Táboa de indicadores sobre un proxecto Java onde se emprega o contorno integrado de desenvolvemento na creación e na compilación de programas simples. – Táboa de indicadores sobre un proxecto Java onde se recoñeza a sintaxe, a estrutura e os compoñentes típicos dunha clase. – Táboa de indicadores sobre un proxecto Java onde se definan clases. – Táboa de indicadores sobre un proxecto Java onde se definan propiedades	▪ Proxecto Java.		▪ OU 1. Táboa de indicadores sobre un proxecto Java onde se escriben programas simples. ▪ OU 2. Táboa de indicadores sobre un proxecto Java onde se utilizan métodos e propiedades de obxectos. ▪ OU 3. Táboa de indicadores sobre un proxecto Java onde se empregan parámetros na chamada a métodos. ▪ OU 4. Táboa de indicadores sobre un proxecto Java onde se empregan construtores. ▪ OU 5. Táboa de indicadores sobre un proxecto Java onde se emprega o contorno integrado de desenvolvemento na creación e na compilación de programas simples. ▪ OU 6. Táboa de indicadores sobre un proxecto Java onde se recoñeza a sintaxe, a estrutura e os compoñentes típicos dunha	4

		e métodos. – Táboa de indicadores sobre un proxecto Java onde se definan construtores. – Táboa de indicadores sobre un proxecto Java onde se desenvolvan programas que instancien e utilicen obxectos das clases creadas anteriormente.			clase. ▪ OU 7. Táboa de indicadores sobre un proxecto Java onde se definan clases. ▪ OU 8. Táboa de indicadores sobre un proxecto Java onde se definan propiedades e métodos. ▪ OU 9. Táboa de indicadores sobre un proxecto Java onde se definan construtores. ▪ OU 10. Táboa de indicadores sobre un proxecto Java onde se desenvolvan programas que instancien e utilicen obxectos das clases creadas anteriormente.	

2. A02. Clases e obxectos.

2.1 Introdución

Na actividade que nos ocupa aprenderanse os seguintes conceptos e manexo de destrezas sobre a Programación Orientada a Obxectos:

- Creación e instanciación de clases.
- Especificadores de acceso.
- Creación de obxectos e construtores.
- Métodos das clases e invocación.
- Destrución de obxectos.

2.2 Actividade

2.2.1 Convención de nomes

- Os nomes das clases deben comenzar con maiúsculas. No caso de ter un nome composto por varias palabras, cada inicial debe estar tamén sempre en maiúsculas. Esta sintaxe se coñece como `upperCamelCase`.

```
public class ProfesorDeSecuntaria {
}
```

- Os nomes dos métodos, dos atributos e dos obxectos deben comenzar por minúscula. No caso de ter un nome composto por varias palabras, cada inicial debe estar en maiúsculas. Esta sintaxe se coñece como `lowerCamelCase`.

```
public class Persoa {
    public String dataNacemento;
    public void irATraballar() {
    }
}

Persoa manuel;
```

- Os nomes das constantes deben escribirse con maiúsculas, se o seu nome ten máis de unha palabra, estas deben separarse mediante un guión baixo.

```
public static final double PI = 3.1416;
```

2.2.2 Creación de clases

Java é unha linguaxe orientada a obxectos pura, polo tanto, todo o código se introduce sempre dentro dunha clase. A definición dunha clase en Java ten a seguinte estrutura:

```
[acceso] class nombreDeClase {
  [acceso] [static] tipo atributo1;
  [acceso] [static] tipo atributo2;
  [acceso] [static] tipo atributo3;
```

```
  ...

  [access] [static] tipo nombreMétodo1([listaDeArgumentos]) {
    ...código del método...
  }
  [access] [static] tipo nombreMétodo2([listaDeArgumentos]) {
    ...código del método...
  }
  ...
}
```

Vexamos un exemplo coa clase Persoa introducida na actividade anterior:

```
public class Persoa {
    /*Atributos da clase*/
    public String nome;
    public String dataNacemento;
    public double peso; //peso en kg
    public int altura; //altura en cm

    /*Métodos da clase*/
    public void durmir() {
           System.out.println("Vou durmir");
    }
    public void comer() {
        System.out.println("Teño fame, vou comer");
    }
    public void camiñar() {
        System.out.println("Estou camiñando");
    }
}
```

2.2.3 Especificadores de acceso

Na actividade anterior presentabamos o concepto de encapsulación, por medio do cal se protexen ás clases, aos atributos e métodos. Polo momento, veremos dous do total de catro especificadores de visibilidade que ofrece Java:

- `public`: Calquera obxecto de calquera clase pode acceder ao atributo ou método.
- `private`: Só o propio obxecto pode acceder ao atributo ou método.

2.2.4 Identificador de obxecto actual

Para acceder aos atributos e métodos desde o propio obxecto, debemos empregar a palabra reservada `this`. Por exemplo, se queremos acceder ao atributo nome do propio obxecto da clase persoa debemos facelo empregando

```
this.nome
```

O uso deste só é obrigatorio cando o nome do método ou parámetro coincide con outro no mesmo ámbito. Por exemplo, un método da clase que recibe un parámetro que se chame igual que un atributo da clase. Veremos estes casos nos exemplos dos seguintes apartados.

2.2.5 Creación de obxectos dunha clase

A creación de obxectos dunha clase, coñecida como instanciación de obxectos, lévase a cabo declarando ese obxecto, do mesmo xeito que se declara unha variable:

```
Clase obxecto;
```

Por exemplo:

```
Persoa manuel;
```

O que declara un obxecto `manuel`, dunha clase `Persoa`, que debe ter sido definida previamente.

Deste xeito unicamente se declara unha referencia ao obxecto, pero non se crea o propio obxecto, para crealo debemos empregar a palabra reservada `new`:

```
manuel = new Persoa();
```

Tamén se pode declarar e crear o obxecto nunha soa liña:

```
Persoa manuel = new Persoa();
```

Para acceder aos métodos e atributos dunha clase (os que permita a súa visibilidade) emprégase o operador punto (.).

```
manuel.nome;
manuel.camiñar();
```

Realiza a tarefa 2.1 “Creación e instanciación da clase Conta Bancaria”, na que se creará a estrutura dunha clase só con atributos e se instanciará a mesma.

2.2.6 Construtores

No momento de crear un obxecto (coa palabra reservada `new`) chámase a un método especial da clase chamado construtor. O construtor da clase é un método que ten o mesmo nome que a clase. É de acceso público, polo que se defínese coa palabra reservada `public`. Non é obrigatorio definir nunha clase un construtor, xa que todas as clases teñen por defecto un construtor baleiro.

```
public class Persoa {
    /*Atributos da clase*/
    public String nome;
    public String dataNacemento;
    public double peso; //peso en kg
    public int altura; //altura en cm

    /*Construtor da clase*/
    public Persoa(String nome, String dataNacemento, double peso, int altura) {
        this.nome = nome;
        this.dataNacemento = dataNacemento;
        this.peso = peso;
        this.altura = altura;
    }

    /*Métodos da clase*/
    ...
}
```

O construtor emprégase habitualmente para inicializar os valores dos atributos do obxecto.

Neste caso, no construtor é obrigatorio empregar o identificador `this`, posto que os parámetros que se lle pasan ao construtor chámanse igual que os atributos da clase. Se non fose así, o uso de `this` sería opcional, como se pode ver a continuación:

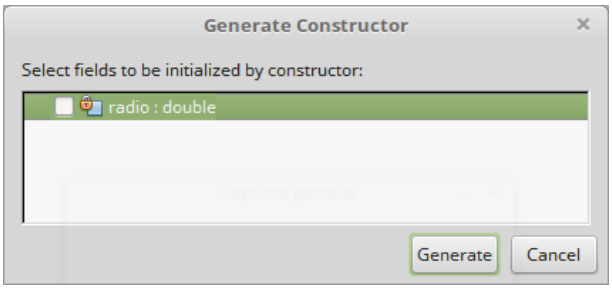
```
public class Persoa {
    /*Atributos da clase*/
    private String nome;
    private String dataNacemento;
    private double peso; //peso en kg
    private int altura; //altura en cm

    /*Construtor da clase*/
    public Persoa(String n, String dN, double p, int a) {
```

```
        nome = n;
        dataNacemento = dN;
        peso = p;
        altura = a;
    }

    /*Métodos da clase*/
    ...
}
```

No contorno de programación NetBeans podemos crear de xeito automático a estrutura do construtor a partir dos atributos da clase. Para iso facemos clic co botón dereito do rato e escollemos *Insert code...*, a continuación escollemos a opción *Constructor...* e escollemos os atributos aos que queremos dar valor no construtor.



Tamén se pode acceder a dita opción desde o menú *Source* → *Insert code...*

Realiza a tarefa 2.2 “Creación e emprego do construtor para a clase Conta Bancaria”, na que se creará un construtor para dar valores aos atributos no momento de creación da clase.

2.2.7 Métodos

Para declarar un método, indicaremos, por esta orde:

- O nivel de visibilidade do método.
- O tipo de dato que devolve como resultado, ou `void` se non devolve ningún resultado.
- O nome do método.
- A lista de parámetros que recibe o método, separados por comas. Para cada parámetro indicaremos o tipo de dato e o nome do mesmo, que nos permitirá acceder a el dentro do código do método.

Despois da declaración, podemos introducir a implementación do método, englobando o código do mesmo entre chaves (`{}`).

O valor de retorno do método, devólvese mediante a palabra reservada `return`. O valor devolto debe coincidir en tipo co indicado na declaración do método. Calquera método con tipo devolto distinto de `void` debe conter polo menos unha sentencia `return`.

O método rematará a súa execución cando:

- Acade o seu final.
- Atope unha sentenza `return`.

- Lance unha excepción (verémolo en actividades posteriores).

O que suceda antes.

Vexamos este exemplo dunha clase Círculo, que ten como atributo o seu radio e métodos para calcular a área, a lonxitude da súa circunferencia e para agrandar o círculo nunha determinada proporción:

```
public class Círculo {
    /* Atributo */
    private double radio;

    /* Construtor */
    public Círculo(double radio) {
        if (radio > 0) {
            this.radio = radio;
        }
    }

    /* Métodos */
    public double área() {
        return Math.PI * Math.pow(radio, 2);
    }

    public double lonxitude() {
        return 2 * Math.PI * radio;
    }

    public void agrandar(double proporcion) {
        radio = radio * proporcion;
    }
}
```

2.2.7.1 Setters e getters

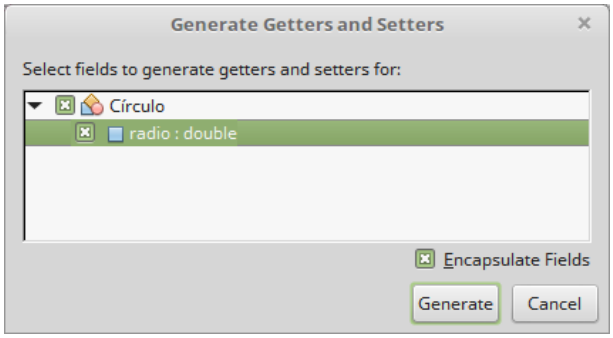
Como xa comentamos con anterioridade, as clases deben evitar que se acceda ás propiedades dos obxectos sen control, porque se poden escribir neles valores erróneos. No exemplo do círculo, vemos como a clase, antes de escribir o valor do radio no construtor comproba que este é maior que cero, xa que o contrario non tería sentido. Esta comprobación debe facerse tamén cando se modifican estes valores. Para que isto sexa posible, o acceso aos atributos debe facerse a través de métodos, coñecidos como *setters* (para escribir) e *getters* (para ler):

```
/* Getters e setters */
    public double getRadio() {
        return radio;
    }

    public void setRadio(double radio) {
        if (radio > 0) {
            this.radio = radio;
        }
    }
```

No contorno de programación NetBeans podemos crear de xeito automático a estrutura destes métodos a partir dos atributos da clase. Para iso facemos clic co botón dereito do rato e escollemos *Insert code...*, a continuación escollemos *Getter and Setter...* (ou unha soa das opcións se así o preferimos) e na seguinte pantalla escollemos os atributos para os que queremos crear ditos métodos.

Tamén se pode acceder a dita opción desde o menú *Source* → *Insert code...*



A opción *Encapsulate fields...* fai que se modifique de forma automática o acceso aos atributos como `private` e que se modifiquen os accesos que a eles se facía de forma directa por métodos de acceso (*getters*).

Realiza a tarefa 2.3 "Creación de métodos getters e setters para a clase Conta Bancaria", na que se modificará a privacidade dos atributos da clase e se crearán métodos para a súa lectura e escritura.

2.2.7.2 Invocación de métodos

Os obxectos pásanse mensaxes a través da invocación dos seus métodos. Para invocar aos métodos dunha clase, teremos empregar o operador punto (.) para o que escribiremos o nome do obxecto seguido dun punto e o nome do método que queremos executar, indicando entre parénteses o valor para todos os parámetros que se indican na declaración dese método, separados por comas. Os valores indicados para os distintos parámetros terán que coincidir co tipo de dato establecido na declaración do método para eses parámetros.

```
public class Xeometría {
    public static void main(String[] args) {
        Círculo c = new Círculo(4);
        System.out.println("Área: " + c.área());
        System.out.println("Lonxitude: " + c.lonxitude());
        c.agrandar(2);
        System.out.println(c.getRadio());
    }
}
```

Igual que pasa cos atributos, dende o código de outra clase só podemos chamar a aqueles métodos que teñen nivel de visibilidade pública.

Realiza a tarefa 2.4 "Creación de métodos para a clase Conta Bancaria", na que se crearán métodos para xestionar o comportamento dunha conta bancaria.

Realiza a Tarefa 2.5 "Creación da clase Persoa", na que aplicaremos todos os conceptos aprendidos nesta actividade para a creación e uso dunha clase Persoa.

2.2.8 Destrución de obxectos e liberación de memoria

En Java non existen destrutores para as clases, xa que dispón do colector de lixo que libera automaticamente os obxectos non utilizados e a memoria ocupada por estes. O colector de lixo elimina aqueles obxectos que xa non están referenciados en ningún dos elementos do programa.

Se queremos realizar algunha liberación explícita de recursos, podemos utilizar o método `finalize` que o colector de lixo invocará inmediatamente antes de liberar o obxecto. Porén, este método é rara vez necesitado no código.

2.3 Tarefas

As tarefas propostas para esta actividade son as seguintes:

- Tarefa 2.1 - **Creación e instanciación da clase Conta Bancaria**, na que se creará a estrutura dunha clase só con atributos e se instanciará a mesma.
- Tarefa 2.2 - **Creación e emprego do construtor para a clase Conta Bancaria**, na que se creará un construtor para dar valores aos atributos no momento de creación da clase.
- Tarefa 2.3 - **Creación de métodos *getters* e *setters* para a clase Conta Bancaria**, na que se modificará a privacidade dos atributos da clase e se crearán métodos para a súa lectura e escritura.
- Tarefa 2.4 - **Creación de métodos para a clase Conta Bancaria**, na que se crearán métodos para xestionar o comportamento dunha conta bancaria.
- Tarefa 2.5 - **Creación da clase Persoa**, na que aplicaremos todos os conceptos aprendidos nesta actividade para a creación e uso dunha clase Persoa.

2.3.1 Tarefa 2.1. Creación e instanciación da clase Conta Bancaria

Nesta tarefa crearase unha clase Conta Bancaria con dous atributos: titular e saldo, e instanciarase a mesma.

Enunciado

Crea unha clase en Java para unha Conta Bancaria, os seus atributos serán o titular e o saldo que, de momento serán públicos.

Solución

- ContaBancaria.java

```
package banco;

public class ContaBancaria {
    public String identificador;
    public String titular;
    public double saldo;
}
```

- Banco.java

```
package banco;

public class Banco {
    public static void main(String[] args) {
        ContaBancaria conta = new ContaBancaria();
        System.out.println("Número: " + conta.identificador);
        System.out.println("Titular: " + conta.titular);
        System.out.println("Saldo: " + conta.saldo);
    }
}
```

- O resultado da execución do programa sería:

```
Número: null
Titular: null
```

```
Saldo: 0.0
```

Como non lle demos valor aos atributos, estes teñen os valores por defecto que asigna Java aos tipos aos que pertencen.

2.3.2 Tarefa 2.2. Creación e emprego do construtor para a clase Conta Bancaria

Nesta actividade engadiremos á clase Conta Bancaria un construtor para poder pasarlle os valores dos atributos da clase.

Enunciado

Engade á clase Conta Bancaria creada no apartado anterior, un construtor ao que se lle pasen como parámetros os valores dos atributos do obxecto. Modifica o programa principal para que empregue o construtor creado.

Soluciónao de dous maneiras distintas, empregando o identificador do propio obxecto `this` e sen que faga falta empregalo.

Solución

- Se empregamos a axuda de NetBeans para crear o construtor de forma automática (Botón dereito → *Insert Code...* → *Constructor* e marcamos todos os parámetros, xérase este código na clase ContaBancaria que é perfectamente válido:

```
package banco;

public class ContaBancaria {
    public String identificador;
    public String titular;
    public double saldo;

    public ContaBancaria(String identificador, String titular, double saldo) {
        this.identificador = identificador;
        this.titular = titular;
        this.saldo = saldo;
    }
}
```

- Se quixeramos modificalo para non empregar o identificador this, teríamos que cambiar os nomes dos parámetros do construtor para que non coincidisen cos nomes dos atributos da clase, do seguinte xeito:

```
public ContaBancaria(String id, String t, double s) {
    identificador = id;
    titular = t;
    saldo = s;
}
```

- No programa principal, habería que incluír os parámetros do construtor:

```
package banco;

public class Banco {
    public static void main(String[] args) {
        ContaBancaria conta = new ContaBancaria("ES59 1234 5678 1234 5678", "Pepe Pérez", 3450.03);
        System.out.println("Número: " + conta.identificador);
        System.out.println("Titular: " + conta.titular);
        System.out.println("Saldo: " + conta.saldo);
    }
```

```
}
```

- E o resultado da execución sería:

```
Número: ES59 1234 5678 1234 5678
Titular: Pepe Pérez
Saldo: 3450.03
```

2.3.3 Tarefa 2.3. Creación de métodos *getters* e *setters* para a clase Conta Bancaria

Nesta tarefa incluiremos os métodos *getters* e *setters* para poder ler e modificar os atributos da clase. Modificaremos ademais a privacidade dos atributos da clase.

Enunciado

Modifica a clase ContaBancaria para que os seus atributos sexan privados e só se poida acceder a eles a partir dos seus métodos.

Solución

- ContaBancaria.java

Podemos crear de forma automática estes métodos, e encapsular á súa vez os atributos facendo clic co botón dereito do rato e escollendo *Insert code...*, a continuación escollemos *Getter and Setter...* e na seguinte pantalla escollemos todos os atributos para os que queremos crear ditos métodos e marcamos a opción *Encapsulate field*s para que se modifique a privacidade dos atributos e se fagan as modificacións necesarias no resto do código para que se empreguen métodos no seu acceso.

```
package banco;

public class ContaBancaria {
    private String identificador;
    private String titular;
    private double saldo;

    public ContaBancaria(String identificador, String titular, double saldo) {
        this.identificador = identificador;
        this.titular = titular;
        this.saldo = saldo;
    }

    public double getSaldo() {
        return saldo;
    }

    public void setSaldo(double saldo) {
        this.saldo = saldo;
    }

    public String getIdentificador() {
        return identificador;
    }

    public void setIdentificador(String identificador) {
        this.identificador = identificador;
    }

    public String getTitular() {
        return titular;
    }

    public void setTitular(String titular) {
```

```
        this.titular = titular;
    }
}
```

Neste caso non fixemos ningunha comprobación á hora de modificar os atributos, pero sería interesante facela para comprobar que o identificador de conta bancaria fose válido, que non facemos porque se precisa o manexo de cadeas de caracteres, que se introducirá en unidades sucesivas.

- Banco.java

```
package banco;

public class Banco {
    public static void main(String[] args) {
        ContaBancaria conta = new ContaBancaria("ES59 1234 5678 1234 5678", "Pe-
pe Pérez", 3450.03);
        System.out.println("Despois de crear o obxecto:");
        System.out.println("Número: " + conta.getIdentificador());
        System.out.println("Titular: " + conta.getTitular());
        System.out.println("Saldo: " + conta.getSaldo());
        conta.setIdentificador("ES35 3456 0987 5432 1768");
        conta.setTitular("Xiana López");
        conta.setSaldo(2456.06);
        System.out.println("Despois de modificar os valores:");
        System.out.println("Número: " + conta.getIdentificador());
        System.out.println("Titular: " + conta.getTitular());
        System.out.println("Saldo: " + conta.getSaldo());
    }
}
```

- E o resultado da execución sería:

```
Despois de crear o obxecto:
Número: ES59 1234 5678 1234 5678
Titular: Pepe Pérez
Saldo: 3450.03
Despois de modificar os valores:
Número: ES35 3456 0987 5432 1768
Titular: Xiana López
Saldo: 2456.06
```

2.3.4 Tarefa 2.4. Creación de métodos para a clase Conta Bancaria

Nesta tarefa crearanse métodos coas operacións habituais dunha conta bancaria.

Enunciado

Engade á clase `ContaBancaria` os seguintes métodos:

- `movemento()` que terá tres parámetros: o tipo de movemento (debe ou haber), o valor do movemento e o seu concepto. Debe actualizarse o saldo da conta con este movemento e almacenar nun novo atributo do obxecto o concepto do último movemento. O ideal sería facer un histórico de movementos, pero as estruturas precisas para facelo estudaranse en unidades sucesivas. Devolve `true` se o saldo é positivo e `false` se a conta quedou en números vermellos.

- `amosarInformacion()`, amosa por pantalla os valores de todos os atributos.

Proba o seu funcionamento desde o programa principal.

Solución

- contaBancaria.java (inclúese unicamente o código xerado nesta tarefa)

```
package banco;

public class ContaBancaria {
    private String identificador;
    private String titular;
    private double saldo;
    private String ultimoMovemento;

    //Aquí iría o código do construtor e os getters e setters dos atributos cre-
ados en tarefas anteriores

    public String getUltimoMovemento() {
        return ultimoMovemento;
    }

    public void setUltimoMovemento(String ultimoMovemento) {
        this.ultimoMovemento = ultimoMovemento;
    }

    public boolean movemento(boolean debe, double valor, String concepto) {
        if (debe) {
            saldo -= valor; //restamos o valor ao saldo
        } else {
            saldo += valor; //sumamos o valor ao saldo
        }
        ultimoMovemento = concepto;
        if (saldo >= 0) {
            return true;
        } else {
            return false;
        }
    }

    public void amosarInformacion() {
        System.out.println("-------------------------------------");
        System.out.println("Estado actual da conta:");
        System.out.println("Número de conta: " + identificador);
        System.out.println("Titular: " + titular);
        System.out.println("Saldo: " + saldo + "€");
        System.out.println("Último movemento: " + ultimoMovemento);
        System.out.println("-------------------------------------");
    }

}
```

- Banco.java (inclúese unicamente o código xerado nesta tarefa)

```
package banco;

public class Banco {
    public static void main(String[] args) {
        ContaBancaria conta = new ContaBancaria("ES59 1234 5678 1234 5678", "Pe-
pe Pérez", 3450.03);
        conta.amosarInformacion();
        conta.movemento(true,200,"Factura teléfono");
        conta.amosarInformacion();
    }
}
```

- Resultado:

```
-------------------------------------
Estado actual da conta:
Número de conta: ES59 1234 5678 1234 5678
Titular: Pepe Pérez
Saldo: 3450.03€
```

```
Último movemento: null
-----------------------------------
-----------------------------------
Estado actual da conta:
Número de conta: ES59 1234 5678 1234 5678
Titular: Pepe Pérez
Saldo: 3250.03€
Último movemento: Factura teléfono
```

2.3.5 Tarefa 2.5. Creación da clase Persoa

Nesta tarefa aplicaremos todos os conceptos aprendidos nesta actividade para a creación e uso dunha clase `Persoa`.

Enunciado

Implementa unha clase chamada `Persoa` coas seguintes condicións:

- Os seus atributos son: nome, idade, DNI, sexo ('H' home, 'M' muller), peso e altura. Non queremos que se accedan directamente a eles.
- Terá un construtor que recibirá todos os valores dos atributos nos parámetros e os incorporará aos mesmos. Comprobará que a idade teña un valor entre 0 e 150 e que o sexo sexa 'H' ou 'M'. De non ser así, non lle asignará valor e imprimirá unha mensaxe por pantalla.
- Os seus métodos son:
  - `calcularIMC()`: Devolverá o Índice de Masa Corporal (IMC), que se calcula como:

$$IMC = \frac{peso(kg)}{altura(m)^2}$$

  - `eMaiorDeIdade()`: indica se a persoa é maior de idade, devolve un valor booleano.
  - `eCorrectoSexo(char sexo)`: comproba que o sexo introducido é correcto. Devolve un valor booleano. Non será visible ao exterior.
  - `eCorrectaIdade(int idade)`: comproba que a idade introducida atópase entre 0 e 150. Devolve un valor booleano. Non será visible ao exterior.
  - `amosarInformacion()`: amosa por pantalla toda a información do obxecto.
  - *Getters* e *setters* de todos os atributos, coas comprobacións pertinentes.

Crea unha clase executable que probe todas as funcionalidades da clase `Persoa`.

Solución

- Persoa.java

```
package tarefa25;

public class Persoa {

    private String nome;
    private int idade;
    private String DNI;
    private char sexo;
    private double peso;
```

```
    private double altura;

    public Persoa(String nome, int idade, String DNI, char sexo, double peso,
double altura) {
        this.nome = nome;
        if (eCorrectaIdade(idade)) {
            this.idade = idade;
        }
        this.DNI = DNI;
        if (eCorrectoSexo(sexo)) {
            this.sexo = sexo;
        }
        this.peso = peso;
        this.altura = altura;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public int getIdade() {
        return idade;
    }

    public void setIdade(int idade) {
        if (eCorrectaIdade(idade)) {
            this.idade = idade;
        }
    }

    public String getDNI() {
        return DNI;
    }

    public void setDNI(String DNI) {
        this.DNI = DNI;
    }

    public char getSexo() {
        return sexo;
    }

    public void setSexo(char sexo) {
        if (eCorrectoSexo(sexo)) {
            this.sexo = sexo;
        }
    }

    public double getPeso() {
        return peso;
    }

    public void setPeso(double peso) {
        this.peso = peso;
    }

    public double getAltura() {
        return altura;
    }

    public void setAltura(double altura) {
        this.altura = altura;
    }

    public double calcularIMC() {
        return peso / (altura * altura);
    }
```

```
    public boolean eMaiorDeIdade() {
        if (idade >= 18) {
            return true;
        } else {
            return false;
        }
    }

    private boolean eCorrectoSexo(char sexo) {
        if (sexo == 'H' || sexo == 'M') {
            return true;
        } else {
            System.out.println(sexo + " non é un valor válido para o sexo.");
            return false;
        }
    }

    private boolean eCorrectaIdade(int idade) {
        if (idade >= 0 && idade <= 150) {
            return true;
        } else {
            System.out.println(idade + " non é un valor válido para a idade.");
            return false;
        }
    }

    public void amosarInformacion() {
        System.out.println("----------------------");
        System.out.println("Información da persoa");
        System.out.println("Nome: " + nome);
        System.out.println("Idade: " + idade);
        System.out.println("DNI: " + DNI);
        System.out.println("Sexo: " + sexo);
        System.out.println("Peso: " + peso);
        System.out.println("Altura: " + altura);
        System.out.println("----------------------");
    }
}
```

- Tarefa25.java

```
package tarefa25;

public class Tarefa25 {

    public static void main(String[] args) {
        Persoa p = new Persoa("Pepe", 35, "12345678Z",'H',70,174);
        p.amosarInformacion();
        System.out.println("IMC: " + p.calcularIMC());
        System.out.println("É maior de idade: " + p.eMaiorDeIdade());
        p.setIdade(5);
        p.amosarInformacion();
        System.out.println("É maior de idade: " + p.eMaiorDeIdade());
        p.setSexo('U');
        p.amosarInformacion();
        p.setIdade(-14);
        p.amosarInformacion();
    }

}
```

- Resultado

```
----------------------
Información da persoa
Nome: Pepe
Idade: 35
DNI: 12345678Z
Sexo: H
Peso: 70.0
```

```
Altura: 174.0
---------------------
IMC: 0.0023120623596247854
É maior de idade: true
---------------------
Información da persoa
Nome: Pepe
Idade: 5
DNI: 12345678Z
Sexo: H
Peso: 70.0
Altura: 174.0
---------------------
É maior de idade: false
U non é un valor válido para o sexo.
---------------------
Información da persoa
Nome: Pepe
Idade: 5
DNI: 12345678Z
Sexo: H
Peso: 70.0
Altura: 174.0
---------------------
-14 non é un valor válido para a idade.
---------------------
Información da persoa
Nome: Pepe
Idade: 5
DNI: 12345678Z
Sexo: H
Peso: 70.0
Altura: 174.0
---------------------
```

3. Materiais

3.1 Documentos de apoio ou referencia

- Especificación da linguaxe Java: http://docs.oracle.com/javase/specs/
- Java SE 8 API Documentation: https://docs.oracle.com/javase/8/docs/api/index.html
- The Java Tutorials: https://docs.oracle.com/javase/tutorial/
- V. RESÚA EIRAS. POOJava. http://iespazodamerce.es/wiki/index.php?title=POOJava
- Java static. Atributos y métodos estáticos o de clase. http://puntocomnoesunlenguaje.blogspot.com.es/2013/02/java-static.html
- J. SÁNCHEZ. Manual de Java. http://jorgesanchez.net/java
- P.A. SZNAJDLEDER. Java a fondo. Alfaomega. 2ª edición.
- Ejercicios propuestos y resueltos programación orientado a objetos Java. https://www.discoduroderoer.es/ejercicios-propuestos-y-resueltos-programacion-orientado-a-objetos-java/
- J. BOBADILLA SANCHO. Java a través de ejemplos. Ra-Ma. Ed. 2003.
- Wikipedia. https://www.wikipedia.org

3.2 Recursos didácticos

- Ordenador persoal con conexión a Internet.
- Contorno de desenvolvemento NetBeans.
- Java SE Development Kit.
- Apuntamentos da profesora.
- Proxector.

4. Avaliación

Criterios de avaliación seleccionados para esta actividade	Evidencia de aprendizaxe	Instrumento de avaliación	Peso na cualificación da UD
▪ CA2.2. Escribíronse programas simples.	▪ Proxecto Java	▪ OU 1. Táboa de indicadores sobre un proxecto Java onde se escriben programas simples.	10%
▪ CA2.4. Utilizáronse métodos e propiedades dos obxectos.	▪ Proxecto Java	▪ OU 2. Táboa de indicadores sobre un proxecto Java onde se utilizan métodos e propiedades de obxectos.	10%
▪ CA2.6. Utilizáronse parámetros na chamada a métodos.	▪ Proxecto Java	▪ OU 3. Táboa de indicadores sobre un proxecto Java onde se empregan parámetros na chamada a métodos.	7%
▪ CA2.8. Utilizáronse construtores.	▪ Proxecto Java	▪ OU 4. Táboa de indicadores sobre un proxecto Java onde se empregan construtores.	5%
▪ CA2.9. Utilizouse o contorno integrado de desenvolvemento na creación e na compilación de programas simples.	▪ Proxecto Java	▪ OU 5. Táboa de indicadores sobre un proxecto Java onde se emprega o contorno integrado de desenvolvemento na creación e na compilación de programas simples.	6%
▪ CA4.1. Recoñeceuse a sintaxe, a estrutura e os compoñentes típicos dunha clase.	▪ Proxecto Java	▪ OU 6. Táboa de indicadores sobre un proxecto Java onde se recoñeza a sintaxe, a estrutura e os compoñentes típicos dunha clase.	8%
▪ CA4.2. Definíronse clases.	▪ Proxecto Java	▪ OU 7. Táboa de indicadores sobre un proxecto Java onde se definan clases.	10%
▪ CA4.3. Definíronse propiedades e métodos.	▪ Proxecto Java	▪ OU 8. Táboa de indicadores sobre un proxecto Java onde se definan propiedades e métodos.	8%
▪ CA4.4. Definíronse construtores.	▪ Proxecto Java	▪ OU 9. Táboa de indicadores sobre un proxecto Java onde se definan construtores.	6%
▪ CA4.5. Desenvolvéronse programas que instancien e utilicen obxectos das clases creadas anteriormente.	▪ Proxecto Java	▪ OU 10. Táboa de indicadores sobre un proxecto Java onde se desenvolvan programas que instancien e utilicen obxectos das clases creadas anteriormente.	10%

4.1 Modelo de proba

Enunciado

O código RGB representa as cores por medio de 3 compoñentes: vermello (R, *red*), verde (G, *green*) e azul (B, *blue*). Así unha cor queda definida por medio de tres valores, RGB. Neste exercicio a cada compoñente asignarémoslle un valor real entre 0.0 e 1.0, por exemplo:

(R, G, B)	cor
(1.0, 0.0, 0.0)	vermello
(0.0, 1.0, 0.0)	verde
(0.0, 0.0, 1.0)	azul
(1.0, 1.0, 1.0)	branco
(0.0, 0.0, 0.0)	negro

Deberemos programar unha clase Cor coas seguintes características:

- Tres atributos privados para os compoñentes R, G e B. Antes de modificar os seus valores, débese comprobar que estes estean entre 0 e 1. Se se trata de modificar un compoñente cun valor negativo, o valor deixarase en 0; ou cun valor maior que 1, deixarase en 1.
- Un construtor ao que se lle pasan os valores dos tres compoñentes.
- Métodos para a escritura e lectura dos compoñentes.
- Un método `gris()` que modifica os atributos privados para pasar a branco e negro: debe calcular o valor medio dos tres compoñentes e poña os tres compoñentes ao devandito valor.
- Un método `filtro(filtror, filtrog, filtrob)` que modifica os atributos privados aplicando un filtro: recibe como parámetros tres coeficientes, o de vermello, o de verde e o de azul. O método multiplica cada compoñente polo coeficiente correspondente; así, `filtro(1.0, 0.0, 0.0)` sería un filtro de vermello, que só deixa pasar o compoñente R. Se os coeficientes non se atopan entre 0 e 1, debe devolver `false`, noutro caso devolverá `true`.
- Un método `amosarInformacion()` que amose por pantalla os valores dos tres compoñenentes.

Crea un pequeno programa principal que probe todas as opcións.

Indica empregando comentarios as distintas partes da estrutura dunha clase.

Solución

- Cor.java

```
package rgb;

public class Cor {      /* Cabeceira da clase */
    /* Atributos */
    private double r;
    private double g;
    private double b;

    /* Construtor */
    public Cor(double r, double g, double b) {
        if (r < 0) {
            this.r = 0;
        } else if (r > 1) {
            this.r = 1;
        } else {
            this.r = r;
        }

        if (g < 0) {
            this.g = 0;
        } else if (g > 1) {
            this.g = 1;
        } else {
            this.g = g;
        }

        if (b < 0) {
            this.b = 0;
        } else if (b > 1) {
```

```
            this.b = 1;
        } else {
            this.b = b;
        }
    }

    /* Métodos getters and setters */
    public double getR() {
        return r;
    }

    public void setR(double r) {
        if (r < 0) {
            this.r = 0;
        } else if (r > 1) {
            this.r = 1;
        } else {
            this.r = r;
        }
    }

    public double getG() {
        return g;
    }

    public void setG(double g) {
        if (g < 0) {
            this.g = 0;
        } else if (g > 1) {
            this.g = 1;
        } else {
            this.g = g;
        }
    }

    public double getB() {
        return b;
    }

    public void setB(double b) {
        if (b < 0) {
            this.b = 0;
        } else if (b > 1) {
            this.b = 1;
        } else {
            this.b = b;
        }
    }

    /* Outros métodos*/
    public void gris() {
        double media = (r+g+b)/3;
        r = media;
        g = media;
        b = media;
    }

    public boolean filtro(double filtror, double filtrog, double filtrob) {
        if (!(filtror >= 0 && filtror <= 1) || !(filtrog >= 0 && filtrog <= 1)
|| !(filtrob >= 0 && filtrob <= 1)) {
            return false; //Algún dos parámetros non está no rango [0,1]
        } else { //Actualizamos os valores
            r *= filtror;
            g *= filtrog;
            b *= filtrob;
            return true;
        }
    }

    public void amosarInformacion() {
        System.out.println("--------------------------------");
```

```
        System.out.println("R: " + r);
        System.out.println("G: " + g);
        System.out.println("B: " + b);
        System.out.println("--------------------------------");
    }
}
```

- Programa principal. RGB.java

```
package rgb;

public class RGB {

    public static void main(String[] args) {
        Cor c = new Cor(0.5,0.4,0.3);
        c.amosarInformacion();
        //Probamos os getters e setters para B, débese repetir para R e G
        c.setB(3);
        c.amosarInformacion();
        c.setB(-2);
        c.amosarInformacion();
        System.out.println("B: " + c.getB());
        if (c.filtro(0.5, 0.25, 0.5)) {
            c.amosarInformacion();
        } else {
            System.out.println("Valores dos coeficientes non válidos");
        }
        if (c.filtro(0.5, 1, 2)) { //Neste caso non actualiza e devolve false
            c.amosarInformacion();
        } else {
            System.out.println("Valores dos coeficientes non válidos");
        }
        c.gris();
        c.amosarInformacion();
    }

}
```

- Resultado

```
--------------------------------
R: 0.5
G: 0.4
B: 0.3
--------------------------------
--------------------------------
R: 0.5
G: 0.4
B: 1.0
--------------------------------
--------------------------------
R: 0.5
G: 0.4
B: 0.0
--------------------------------
B: 0.0
--------------------------------
R: 0.25
G: 0.1
B: 0.0
--------------------------------
Valores dos coeficientes non válidos
--------------------------------
R: 0.11666666666666665
G: 0.11666666666666665
B: 0.11666666666666665
--------------------------------
```

Táboas de indicadores

OU 1. Táboa de indicadores sobre un proxecto Java onde se escriben programas simples.

Nome:		Data:	
CA2.2. Escribíronse programas simples.			
Indicadores	**Puntuación base**	**Puntuación obtida**	**Observacións**
▪ Crea a estrutura da clase	3		
▪ Indica ben o nivel de acceso aos atributos	1,5		
▪ Elixe ben o tipo de datos dos atributos	1,5		
▪ É capaz de instanciar un obxecto desa clase	2		
▪ Emprega as regras de estilo para os nomes de clases, atributos e métodos.	2		

OU 2. Táboa de indicadores sobre un proxecto Java onde se utilizan métodos e propiedades de obxectos.

Nome:		Data:	
CA2.4. Utilizáronse métodos e propiedades dos obxectos.			
Indicadores	**Puntuación base**	**Puntuación obtida**	**Observacións**
▪ Emprega o método filtro().	2		
▪ Emprega o método gris().	2		
▪ Emprega os métodos getters e setters.	2		
▪ Accede correctamente aos atributos desde o propio obxecto.	2		
▪ Emprega axeitadamente o identificador this para acceder aos métodos e propiedades do obxecto cando é preciso.	2		

OU 3. Táboa de indicadores sobre un proxecto Java onde se empregan parámetros na chamada a métodos.

Nome:		Data:	
CA2.6. Utilizáronse parámetros na chamada a métodos.			
Indicadores	**Puntuación base**	**Puntuación obtida**	**Observacións**
▪ Especifica os parámetros e os seus tipos no método filtro().	2,5		
▪ Comproba os valores adecuados dos parámetros no método filtro().	2,5		
▪ Especifica os parámetros e os seus tipos nos setters.	2,5		
▪ Comproba os parámetros e asigna os valores, se procede, nos setters.	2,5		

OU 4. Táboa de indicadores sobre un proxecto Java onde se empregan construtores.

Nome:		Data:	
CA2.8. Utilizáronse construtores.			

Indicadores	Puntuación base	Puntuación obtida	Observacións
▪ Crea a estrutura do construtor da clase.	1,5		
▪ Indica ben os parámetros do construtor.	1,5		
▪ Escolle apropiadamente os tipos de datos dos parámetros do construtor.	1,5		
▪ Asigna adecuadamente os parámetros aos atributos.	1,5		
▪ Fai as comprobacións pertinentes antes de asignar os parámetros ao construtor.	2		
▪ É capaz de instanciar un obxecto desa clase empregando o construtor correctamente.	2		

OU 5. Táboa de indicadores sobre un proxecto Java onde se emprega o contorno integrado de desenvolvemento na creación e na compilación de programas simples.

Nome:		Data:	
CA2.9. Utilizouse o contorno integrado de desenvolvemento na creación e na compilación de programas simples.			
Indicadores	**Puntuación base**	**Puntuación obtida**	**Observacións**
▪ Crea o proxecto no IDE.	2		
▪ Crea novas clases no IDE.	2		
▪ Emprega a opción de inserción de código do IDE para crear o construtor.	2		
▪ Emprega a opción de inserción de código do IDE para crear os getters e setters.	2		
▪ Executa o proxecto no IDE.	2		

OU 6. Táboa de indicadores sobre un proxecto Java onde se recoñeza a sintaxe, a estrutura e os compoñentes típicos dunha clase.

Nome:		Data:	
CA4.1. Recoñeceuse a sintaxe, a estrutura e os compoñentes típicos dunha clase.			
Indicadores	**Puntuación base**	**Puntuación obtida**	**Observacións**
▪ Sinalou a cabeceira da clase.	2		
▪ Sinalou os atributos da clase.	2		
▪ Sinalou o construtor da clase.	2		
▪ Sinalou os métodos da clase.	2		
▪ Sinalou os métodos getters e setters da clase.	2		

OU 7. Táboa de indicadores sobre un proxecto Java onde se definan clases.

Nome:		Data:	
CA4.2. Defiríronse clases.			
Indicadores	**Puntuación base**	**Puntuación obtida**	**Observacións**

▪ Crea a clase nun ficheiro co seu nome.	2		
▪ Define ben a sintaura da clase.	2		
▪ Crea os atributos da clase.	2		
▪ Crea a sintura do construtor da clase.	2		
▪ Crea a sinatura dos métodos.	2		

OU 8. Táboa de indicadores sobre un proxecto Java onde se definan propiedades e métodos.

Nome:		Data:	
CA4.3. Defiríronse propiedades e métodos.			
Indicadores	**Puntuación base**	**Puntuación obtida**	**Observacións**
▪ Crea a estrutura do método filtro().	0,75		
▪ Devolve valores correctamente no método filtro().	1		
▪ Implementa a funcionalidade do método filtro().	1,25		
▪ Devolve correctamente os valores no método filtro().	1		
▪ Crea a estrutura do método gris().	0,75		
▪ Implementa a funcionalidade do método gris().	1,25		
▪ Crea a estrutura do métodos getters e setters.	0,75		
▪ Devolve valores correctamente nos getters.	1		

OU 9. Táboa de indicadores sobre un proxecto Java onde se definan construtores.

Nome:		Data:	
CA4.4. Defiríronse construtores.			
Indicadores	**Puntuación base**	**Puntuación obtida**	**Observacións**
▪ Creou adecuadamente a cabeceira do construtor da clase Cor.	3		
▪ Escolleu correctamente os parámetros que recibe o construtor.	3,5		
▪ Implementou correctamente a funcionalidade do construtor.	3,5		

OU 10. Táboa de indicadores sobre un proxecto Java onde se desenvolvan programas que instancien e utilicen obxectos das clases creadas anteriormente.

Nome:		Data:	
CA4.5. Desenvolvéronse programas que instancien e utilicen obxectos das clases creadas anteriormente.			
Indicadores	**Puntuación base**	**Puntuación obtida**	**Observacións**
▪ Creou o programa principal de xeito funcional.	2		
▪ Creou obxectos da clase cor.	2		
▪ Empregou correctamente o construtor.	2		
▪ Empregou correctamente os getters e setters.	2		

▪ Empregou correctamente os métodos.	2		