

[*www.utpl.edu.ec*](http://www.utpl.edu.ec/)

*Sistemas Informáticos y Computación*

***PROGRAMACIÓN DE ALGORITMOS***

**

*Examples: About Object-oriented programming*

Table of Contents

[La clase empleado *(ejercicio 3.14)*  1](#_Toc465522682)

[*Problema* de la clase empleado 1](#_Toc465522683)

[*Análisis* para las clases empleado y prueba 1](#_Toc465522684)

[*Diseño* del Uml de clases Empleado y PruebaEmpleado 1](#_Toc465522685)

[*Codificación y documentación* del .java para *Empleado* y *PruebaEmpleado* 1](#_Toc465522686)

[La clase *[Nombre del Ejercicio]* *(ejercicio [numeroReferenteLibro])* 4](#_Toc465522687)

[*Problema* [Nombre del Ejercicio del libro *[referencia del libro]*] 4](#_Toc465522688)

[*Análisis* [Nombre del Ejercicio del libro] 4](#_Toc465522689)

[*Diseño* [Nombre del Ejercicio del libro] 4](#_Toc465522690)

[*Codificación y documentación* [Nombre del Ejercicio del libro] 4](#_Toc465522691)

[Bibliography 6](#_Toc465522692)

# **La clase Cuenta *(ejercicio 3.12)*** ***[[1]](#footnote-2)***

## ***Problema* de la clase Cuenta**

Cree una clase llamada Cuenta, que incluya una pieza de información como variables de instancia: saldo *(tipo double)*. Su clase debe tener un constructor que inicialice la variable de instancia. Proporcione un método establecer y un método obtener para cada variable de instancia. Si el salario mensual no es positivo, establézcalo a 0.0. Escriba una aplicación de prueba llamada PruebaEmpleado, que demuestre las capacidades de cada Empleado. Cree dos objetos Empleado y muestre el salario anual de cada objeto. Después, proporcione a cada Empleado un aumento del 10% y muestre el salario anual de cada Empleado otra vez.

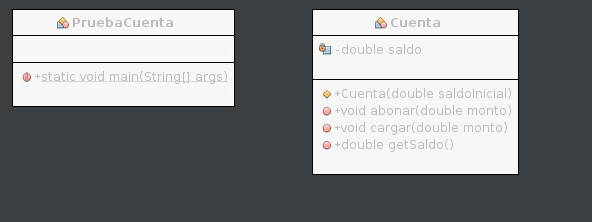
## ***Análisis* para las clases Cuenta y pruebaCuenta**

Cree una clase llamada **Empleado**, que incluya tres piezas de información como variables de instancia: un **primer nombre** *(tipo String)*, un **apellido paterno** *(tipo String)* y un **salario mensual** *(double)*. Su clase debe tener un ***constructor*** que inicialice las tres variables de instancia. Proporcione un método ***establecer*** y un método ***obtener*** para cada variable de instancia. Si el salario mensual no es positivo, establézcalo a 0.0. Escriba una aplicación de prueba llamada **PruebaEmpleado**, que demuestre las capacidades de cada Empleado. Cree dos objetos Empleado y muestre el salario anual de cada objeto. Después, proporcione a cada Empleado un aumento del 10% y muestre el salario anual de cada Empleado otra vez.

|  |  |  |  |
| --- | --- | --- | --- |
| ***CLASES*** | **ATRIBUTOS** | ***MÉTODOS*** | **REQUERIMIENTOS** |
| ***Cuenta*** | saldo | constructor()  establecer(), para todos los atributos.  obtener(), para todos los atributos. | Si el saldo no es positivo, establézcalo a 0.0. |
| ***PruebaEmpleado*** | Cuenta1  cuenta2 | main() |  |

## ***Diseño* del Uml de clases Empleado y PruebaEmpleado**

Desarrollando la lectura analítica del planteamiento del problema se ha obtenido las siguientes clases integradas en UML con el fin de realizar una codificación sencilla en Java. La fecha que sale desde la clase ***PruebaCuenta*** y se une con la clase ***Cuenta***, tiene una relación de “usa un”, es decir, la clase ***PruebaCuenta*** usa la clase ***Cuenta*** para crear sus objetos.



**Fig. 1** | UML de la clase ***Cuenta*** y ***PruebaCuenat***. *(Desarrollado en NetBeans)*

## ***Codificación y documentación* del .java para *Cuenta y pruebaCuenta.***

La siguiente es la clase **Cuenta** *Fig.2*, desarrollada en el lenguaje Java, para describir a una persona ***Cuenta*** con un atributo asociado y sus respectivos métodos/propiedades set/get para obtener y establecer sus valores como una forma alternativa al constructor.

|  |
| --- |
| package Ejercicio3\_12; |
|  |
| /\*\* |
| \* |
| \* @author leonardo |
| \*/ |
| public class Cuenta { |
|  |
| private double saldo; |
|  |
| public Cuenta(double saldoInicial) { |
| if (saldoInicial > 0.0) { |
| saldo = saldoInicial; |
| } |
| } |
|  |
| public void abonar(double monto) { |
| saldo = saldo + monto; |
| } |
|  |
| public void cargar(double monto) { |
| if (monto > saldo) |
| System.out.println("El monto a cargar excede el saldo de la cuenta\n"); |
| else |
| saldo = saldo - monto; |
| } |
|  |
| public double getSaldo() { |
| return saldo; |
| } |
|  |
| } |

**Fig. 2** | Codificación de la clase ***Cuenta*** e implementación de las propiedades set/get y más requerimientos

**Url. 1** | <https://git.taw.utpl.edu.ec/pdirene/PROGRAMAS_CVS-GIT/blob/0c9235e644646daf4c4d39837f1f2b65245662cf/PRO_ALG/MP-ProAlg/Proyect_MP-ProAlg/src/Empleado.java>

Esta clase permite la creación de dos tipos de objetos ***Cuenta*** el primero podría crearse con inicializar sus nombres y salario , y en el segundo tipo de objeto ***Empleado*** enviamos al segundo constructor sobrecargado (línea nro. 28) los valores para los atributos nombre, apellido y salario, éste último siempre se inicializará con un valor positivo dado el requerimiento, por ello, en la línea 31 cuando el objeto setea el valor para el salario, este llama al método *setSalario(double)* de la línea 65 en el cual a través de un ***if ternario*** se asigna 0.0 si se recibe un valor menor a cero en salario, caso contrario se asigna dicho salario positivo.

Los demás métodos set y get, realizan las funciones estándar de establecer y obtener los valores para cada atributo de la clase. No así en el método *aumentoSalario(double)*, el cual realiza un aumento salario (modificación del atributo salario) dado un valor porcentual.

**Fig. 3** | Demostración de la clase ***Cuenta***, de sus constructores, y métodos.

**Url. 1** | <https://git.taw.utpl.edu.ec/pdirene/PROGRAMAS_CVS-GIT/blob/0c9235e644646daf4c4d39837f1f2b65245662cf/PRO_ALG/MP-ProAlg/Proyect_MP-ProAlg/src/Empleado.java>

En la clase **PruebaCuenta** *Fig. 3.* se prueba la clase *Empleado* realizando la declaración de 2 objetos *(variables por referencia)* en las líneas 13 y 14, los dos ya inicializados gracias al constructor sobrecargado de la línea 28 class *Empleado* Fig. 2., aunque por el primer constructor vacío de Empleado Fig. 2. línea 16. es posible la creación de objetos *(empleados)* sin especificar parámetros y luego establecer sus valores con los métodos set de dicha clase.

Una vez declarados los objetos *<empleadoA>* y *<empleadoB>,* se los puede presentar directamente con el nombre del objeto hacia cualquier salida (consola sout, html, otros controles), en este caso la salida a consola sout en las líneas 16, 21, 23 y 27, en las primeras presentaciones de cada objeto (líneas 16, 23) se los puede ver con sus valores asignados en el momento de su creación *(por el segundo constructor de Empleado),* pero ya en las segundas presentaciones (líneas 21 y 27) se verifica que el monto del salario a cambiado gracias a las operaciones matemáticas de la línea 19, y la llamada del método ***aumentoSalario(double)***.

Nota: La fórmula en la línea 20, es más óptima para realizar el cálculo del aumento salarial del 10%, en alternativa a la formula anterior, línea 19.

|  |
| --- |
| Sando de cuenta1: $50,00 |
| Sando de cuenta2: $0,00 |
| Escriba el monto a depositar para cuanta1: |
| 25 |
|  |
| sumando $25,00 al sando de cuanta1 |
|  |
| Saldo cuenta1: $75,00 |
| Saldo cuenta2: $0,00 |
| Escriba el monto a depositar para cuanta2: |
| 50,78 |
|  |
| sumando $50,78 al sando de cuanta1 |
|  |
| Saldo cuenta1: $75,00 |
| Saldo cuenta2: $50,78 |
| Escriba el monto a retirar de cuanta1: |
| 34,67 |
|  |
| restando $34,67 al sando de cuanta1 |
|  |
| Saldo cuenta1: $40,33 |
| Saldo cuenta2: $50,78 |
| Escriba el monto a retirar de cuanta2: |
| 34 |
|  |
| restando $34,00 al sando de cuanta2 |
|  |
| Saldo cuenta1: $40,33 |
| Saldo cuenta2: $16,78 |

**Fig. 4** | Resultados de la ejecución de la clase ***PruebaCuenta***, la cual prueba las capacidades de la clase ***Cuenta***.

Ésta es la ejecución de esta clase ***PruebaCuenta*** Fig. 4. en donde se puede confirmar lo anteriormente citado, la presentación de los objetos *<empleadoA>* y *<empleadoB>* sin el aumento salarial, y luego con la operación matemática en *main()*, y *aumentoSalario(double)* se modifican las copias del atributo ***<salario>*** para cada objeto.

# **La clase *Perfil Medico* *(ejercicio 3.17)***

## ***Problema* Perfil Medico**

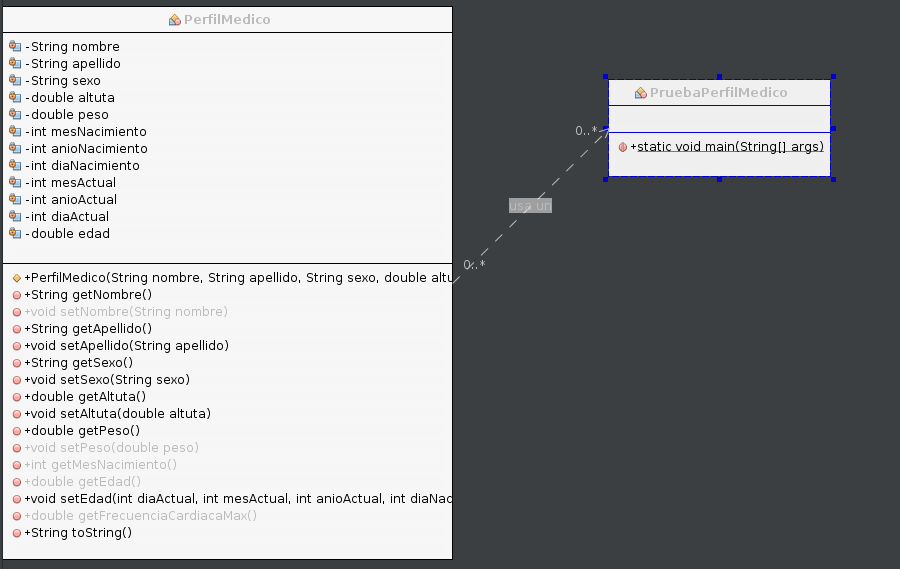
[Enunciado completo y detallado del problema]

## ***Análisis* [Nombre del Ejercicio del libro]**

[Haciendo las lecturas necesarias del enunciado completo del problema: Resaltar, subrayas, hacer otros tipos de maracas para cada elemento de las clases, *mirar el ejemplo adjunto*]

|  |  |  |  |
| --- | --- | --- | --- |
| ***CLASES*** | **ATRIBUTOS** | ***MÉTODOS*** | **REQUERIMIENTOS** |
| ***PerfilMedico*** | Nombre  apellido  sexo  altura  peso  mesNacimiento  anioNacimiento  diaNacimiento  mesActual  anioNacimiento  diaNacimiento  edad | Constructor1()  metodos get/set de cada parametro.  Metodos getFrecuenciaCardiacaMax  toString |  |
| ***PruebaPerfilMedico*** | Pm1 | Main() |  |

## ***Diseño* Cuenta**



**Fig. nro.X** | Nombre y descripción del UML

## ***Codificación y documentación* Cuenta**

[

|  |
| --- |
| package Ejercicio3\_17; |
|  |
| import java.util.Calendar; |
|  |
| /\*\* |
| \* |
| \* @author leonardo |
| \*/ |
| public class PerfilMedico { |
|  |
| private String nombre; |
| private String apellido; |
| private String sexo; |
| private double altuta; |
| private double peso; |
| private int mesNacimiento; |
| private int anioNacimiento; |
| private int diaNacimiento; |
| private int mesActual; |
| private int anioActual; |
| private int diaActual; |
| private double edad; |
|  |
| public PerfilMedico(String nombre, String apellido, String sexo, double altuta, double peso, int mesNacimiento, int anioNacimiento, int diaNacimiento, int mesActual, int anioActual, int diaActual) { |
| this.nombre = nombre; |
| this.apellido = apellido; |
| this.sexo = sexo; |
| this.altuta = altuta; |
| this.peso = peso; |
| this.mesNacimiento = mesNacimiento; |
| this.anioNacimiento = anioNacimiento; |
| this.diaNacimiento = diaNacimiento; |
| this.mesActual = mesActual; |
| this.anioActual = anioActual; |
| this.diaActual = diaActual; |
| } |
|  |
| public String getNombre() { |
| return nombre; |
| } |
|  |
| public void setNombre(String nombre) { |
| this.nombre = nombre; |
| } |
|  |
| public String getApellido() { |
| return apellido; |
| } |
|  |
| public void setApellido(String apellido) { |
| this.apellido = apellido; |
| } |
|  |
| public String getSexo() { |
| return sexo; |
| } |
|  |
| public void setSexo(String sexo) { |
| this.sexo = sexo; |
| } |
|  |
| public double getAltuta() { |
| return altuta; |
| } |
|  |
| public void setAltuta(double altuta) { |
| this.altuta = altuta; |
| } |
|  |
| public double getPeso() { |
| return peso; |
| } |
|  |
| public void setPeso(double peso) { |
| this.peso = peso; |
| } |
|  |
| public int getMesNacimiento() { |
| return mesNacimiento; |
| } |
|  |
| public double getEdad() { |
| double fechaNacimiento = 10000 \* anioNacimiento + 100 \* mesNacimiento + diaNacimiento; |
| double fechaActual = 10000 \* anioActual + 100 \* mesActual + diaActual; |
| return (double) ( fechaActual - fechaNacimiento)/10000; |
| } |
|  |
| public void setEdad(int diaActual, int mesActual, int anioActual, int diaNacimiento, int mesNacimiento, int anioNacimiento) { |
| this.diaActual = diaActual; |
| this.mesActual = mesActual; |
| this.anioActual = anioActual; |
| this.diaNacimiento = diaNacimiento; |
| this.mesNacimiento = mesNacimiento; |
| this.anioNacimiento = anioNacimiento; |
| this.edad = edad; |
| } |
|  |
| public double getFrecuenciaCardiacaMax() { |
| return 220 - getEdad(); |
| } |
|  |
| public String toString() { |
| return String.format("Nombre: %s\t Apellido: %s\nEdad: %.2f\nFrecuencia Cardiaca maxima: %.2f", getNombre(), getApellido(), getEdad(), getFrecuenciaCardiacaMax()); |
| } |
| } |

|  |  |
| --- | --- |
| Nombre: leonardo | Apellido: rodhen |
| Edad: 23,03 |  |
| Frecuencia Cardiaca maxima: 196,97 |  |

# **La clase *Racional* *(ejercicio 8.15)***

## ***Problema* Racional**

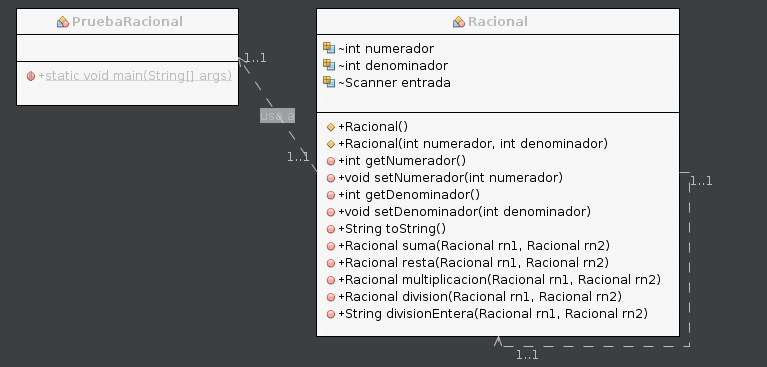
[Enunciado completo y detallado del problema]

## ***Análisis* Racional**

[Haciendo las lecturas necesarias del enunciado completo del problema: Resaltar, subrayas, hacer otros tipos de maracas para cada elemento de las clases, *mirar el ejemplo adjunto*]

|  |  |  |  |
| --- | --- | --- | --- |
| ***CLASES*** | **ATRIBUTOS** | ***MÉTODOS*** | **REQUERIMIENTOS** |
| ***Racional*** | Numerador  denominador | Constructor1()  metodos get/set de cada parametro.  Metodo Racial suma  Metodo Racial resta  Metodo Racial miltiplicacion  Metodo Racial division  Metodo Racial divisonEntera |  |
| ***PruebaRacional*** | Opcion  rn1  rn2  obj1 | main() |  |

## ***Diseño* Cuenta**



**Fig. nro.X** | Nombre y descripción del UML

## ***Codificación y documentación* Cuenta**

|  |
| --- |
| package Ejercicio8\_15; |
|  |
| import java.util.Scanner; |
|  |
| /\*\* |
| \* |
| \* @author leonardo |
| \*/ |
| public class Racional { |
|  |
| int numerador; |
| int denominador; |
| Scanner entrada = new Scanner(System.in); |
|  |
| public Racional() { |
| numerador = 0; |
| denominador = 1; |
| } |
|  |
| public Racional(int numerador, int denominador) { |
| this.numerador = numerador; |
| this.denominador = denominador; |
| } |
|  |
| public int getNumerador() { |
| return numerador; |
| } |
|  |
| public void setNumerador(int numerador) { |
| this.numerador = numerador; |
| } |
|  |
| public int getDenominador() { |
| return denominador; |
| } |
|  |
| public void setDenominador(int denominador) { |
| this.denominador = denominador; |
| } |
|  |
| public String toString() { |
| return String.format("%d/%d\n", numerador, denominador); |
| } |
|  |
| public Racional suma(Racional rn1, Racional rn2) { |
| int a = rn1.numerador; |
| int b = rn1.denominador; |
| int c = rn2.numerador; |
| int d = rn2.denominador; |
| this.setNumerador((a \* d) + (b \* c)); |
| this.setDenominador(b \* d); |
| return this; |
| } |
|  |
| public Racional resta(Racional rn1, Racional rn2) { |
| int a = rn1.numerador; |
| int b = rn1.denominador; |
| int c = rn2.numerador; |
| int d = rn2.denominador; |
| this.setNumerador((a \* d) - (b \* c)); |
| this.setDenominador(b \* d); |
| return this; |
| } |
|  |
| public Racional multiplicacion(Racional rn1, Racional rn2) { |
| int a = rn1.numerador; |
| int b = rn1.denominador; |
| int c = rn2.numerador; |
| int d = rn2.denominador; |
| this.setNumerador(a \* c); |
| this.setDenominador(b \* d); |
| return this; |
| } |
|  |
| public Racional division(Racional rn1, Racional rn2) { |
| int a = rn1.numerador; |
| int b = rn1.denominador; |
| int c = rn2.numerador; |
| int d = rn2.denominador; |
| this.setNumerador(a \* d); |
| this.setDenominador(b \* c); |
| return this; |
| } |
|  |
| public String divisionEntera(Racional rn1, Racional rn2) { |
| int a = rn1.numerador; |
| int b = rn1.denominador; |
| int c = rn2.numerador; |
| int d = rn2.denominador; |
| double divisionEntera = (a \* d) / (b \* c); |
| System.out.println("Ingrese el numero de decimales que desea imprimir"); |
| int presicion = entrada.nextInt(); |
| return String.format("División entera: %.2f", divisionEntera); |
| } |
|  |
| } |

|  |
| --- |
| Fraccion 1: 1/2 |
|  |
| Fraccion 2: 2/5 |
|  |
| Ingrese la Opcion que desea ejecutar: |
| 1. Suma. |
| 2. Resta. |
| 3. Multiplicación. |
| 4. División en fracción. |
| 5. Division entera. |
|  |
| 1 |
| Suma: 9/10 |

# **Bibliography**

DEITEL, P. J., & DEITEL, H. M. (2012). *COMO PROGRAMAR EN JAVA* (9 ed.). (E. PEARSON, Ed.) Naucalpan de Juarez, México.

Kojevnikov, A. (1 de 1 de 2016). *hilite.me*. Recuperado el 26 de 8 de 2016, de Flask and Pygments: http://hilite.me

Universidad Técnica Particular. (1 de 8 de 2016). *EVA*. Recuperado el 26 de 8 de 2016, de Entorno Virtual de Aprendizaje UTPL: https://eva3.utpl.edu.ec/

Universidad Técnica Particular de Loja. (1 de 1 de 2016). *GitLab*. (UTPL, Editor) Recuperado el 29 de 8 de 2016, de GitLab Community Edition: https://git.taw.utpl.edu.e

1. (DEITEL & DEITEL, 2012), *Pag. 101* [↑](#footnote-ref-2)