Programación I

Examen final - 9-Enero-2015

Un banco nos ha pedido gestionar la paga extraordinaria de los mejores empleados del Bajo Deba. Para ello se debe implementar la clase Empleado.

[3 puntos] Clase Empleado

Esta clase representa a cada empleado de una de las cuatro sucursales del Bajo Deba que gestiona este programa. La información que se registra de cada empleado es:

- Nombre, cadena de caracteres, de al menos 5 caracteres.
- Sueldo base mensual, número real, entre 600 y 2400 euros.
- Código, una cadena codificada de la siguiente forma:
    - XX, dos caracteres que representan la oficina (“ER”-Ermua, “EI”-Eibar, “ME”-Mendaro, “DB”-Deba)
    - 99, dos dígitos que representan el número de empleado.
    - X, un carácter que representa el rango (‘A’, ‘B’, ‘C’ o ‘D’)
- Puntos, es un entero que indica el número de puntos obtenidos por el empleado, se debe inicializar a cero.

Para manejar estos datos, la clase Empleado facilita los siguientes métodos:

- **Constructor** sin argumentos, que inicializa todos los atributos con valores válidos por defecto (cada programador debe elegir cuáles son estos valores)
- **Constructor** con parámetros, que inicializa todos los atributos con valores pasados como parámetro, comprobando que son valores válidos. En caso de que no lo sean, se inicializarán con valores por defecto (iguales a los del constructor anterior).
- Métodos **set y get** para todos los atributos de la clase, excepto el Código, que no se puede modificar. IMPORTANTE: cada vez que se cambie el valor de un atributo de la clase, se debe comprobar que el nuevo valor a asignar sea válido, en caso contrario se dejará el valor que tenía antes (no se debe mostrar ningún mensaje por pantalla)
- Además se deberán programar los siguientes métodos get, obtenidos a partir del código del empleado (XX99X, *pe.”DB25B”*):
    - **getOficina**, devuelve la cadena con los dos caracteres que conforman el código de la oficina (en el ejemplo, devolvería “DB”)
    - **getCodigoNum**, devuelve el entero que representa el código de empleado (en el ejemplo, devolvería 25)

    - **getRango**, devuelve el carácter con el rango del empleado (en el ejemplo devolvería ‘B’)

- Método **leer**, que lee por teclado todos los atributos de la clase (salvo los puntos, que los pone a cero), y comprueba que sean correctos. En caso de no serlo los seguirá pidiendo una y otra vez.

Para facilitar la validación del código, se debe programar en la clase Empleado, un método estático denominado **esCorrecto**, que recibe un código y devuelve *true* si responde a las características de un código correcto, y *false* en caso contrario.

Para gestionar la paga extra disponemos además de la clase Operacion, **ya implementada**, que representa las operaciones realizadas por cada uno de los empleados. El código de la clase Operación se muestra al final del enunciado.

Clase GestionPagaExtra

Una vez creada la clase Empleado, se deben programar los siguientes métodos para gestionar la paga extra de los cinco mejores empleados de cada una de las cuatro sucursales.

La información de todos los empleados está en un array denominado *vEmpleados*, que ya está cargado.

[3 puntos] Método cargarOperaciones

Este método lee por teclado los datos de todas las operaciones de los empleados, que serán un máximo de 1000, y los almacena en el array *vOperaciones* en el orden en que se introducen. Después de introducir cada operación se preguntará al usuario si quiere introducir más (contemplar las mayúsculas y minúsculas). El proceso finaliza al introducir 1000 operaciones o cuando el usuario indique que no quiere introducir más.

Por cada operación, debe realizar las siguientes comprobaciones:

- Se lee el código del empleado, y se comprueba:
    - Que el código es correcto en cuanto a formato, y que corresponda a un empleado existente. Para ello se implementará el método **leerCodigoCorrectoExistente [1 de los 3 puntos]**, que realiza las comprobaciones siguientes, y devuelve el código que cumpla ambas condiciones: es correcto y existe un empleado con dicho código
        - Que es correcto, que sigue el formato “XX99X”. En caso de que no lo sea se muestra un mensaje de error y se vuelve a pedir. Para realizar esta comprobación se debe utilizar el método ya implementado en la clase Empleado (esCorrecto).
        - A continuación se debe comprobar que el código introducido corrresponde a un empleado de alguna de las cuatro sucursales, buscando dicho código en el array de Empleados. En caso de que exista, este será el código que se devuelve, en caso de que no exista se vuelve a pedir un código y se hacen de nuevo todas las comprobaciones.
- A continuación se leen y validan el tipo de operación (H-hipoteca, F-fondos, C-cuenta, P-plan de pensiones) y el importe (un número > 0)

Ejemplo de ejecución:

```
Código de empleado: BI34Z
Error, el código es incorrecto!
Código de empleado: BI34A
No existe un empleado con ese código!
Código de empleado: ER03B
Tipo de operación (H, F, C, P): Z
Tipo de operación (H, F, C, P): F
Importe (>0): -25
Importe (>0): 1200
¿Desea introducir más? (S/N): s
Código de empleado: BI34Z
Error, el código es incorrecto!
Código de empleado: BI34A
No existe un empleado con ese código!
Código de empleado: EI06A
Tipo de operación (H, F, C, P): P
Importe (>0): 9000
¿Desea introducir más? (S/N): N
```

[2 puntos] Método asignarPuntos

En base a las operaciones que haya realizado cada empleado, se le asigna un número de puntos:

- H: +3 puntos por euro de la operación
- F: +1 punto por euro de la operación
- C: +2 puntos por cada 100 euros de la operación
- P: +3 puntos por cada 1500 euros de la operación

Se deben procesar todas las operaciones contenidas en el array *vOperaciones* y actualizar como corresponda los puntos de los empleados del array *vEmpleados*. Tened en cuenta que sabemos que todas las operaciones contenidas en el array *vOperaciones* corresponde a algún empleado del array *vEmpleado*.

NOTA: para convertir un dato de tipo *double* a *int*, se puede hacer de la siguiente forma:

```
double p = 5.25;
int a = (int) p; //a tomará el valor 5
```

[2 puntos] Método puntosPorSucursalTipo

Este método muestra por pantalla el número total de puntos acumulados por cada una de las cuatro sucursales y los cuatro tipos de empleados existentes. Para ello será necesario utilizar un array de dos dimensiones, donde se almacene esta información.

[1 punto-opcional] Método asignarPagaExtra

Este método saca por pantalla los empleados de cada rango (A, B, C, D) que hayan conseguido más puntos, ya que será a los que se asigne una paga extra de 1000 euros. Ejemplo de ejecución:

```
La paga de 1.000 euros corresponde a:
A - Iratxe Eguíluz - 20 puntos
B - Luis Gutiérrez - 150 puntos
C - Olga Rayón - 6000 puntos
D - Almudena Gamiz - 1250 puntos
```

```
      public class Operacion {
          private String codigo;
          private char tipo;
          private double importe;
          public Operacion(String codigo, char tipo, double importe) {
            this.codigo = codigo;
            this.tipo = tipo;
            this.importe = importe;
          }
          public String getCodigo() {       return codigo;     }
          public void setCodigo(String codigo) {
            this.codigo = codigo;
          }
          public char getTipo() {      return tipo;    }
          public void setTipo(char tipo) {
            this.tipo = tipo;
          }
          public double getImporte() {      return importe;     }
          public void setImporte(double importe) {
            this.importe = importe;
          }
      }
  public static void main(String[] args) {
   Operacion vOperaciones[] = new Operacion[1000];
   Empleado vEmpleados[] = {new Empleado("Ana Pastor", "ER01A", 2300),
              new Empleado("Luis Gutiérrez", "ER02B", 2000),
              new Empleado("Andrés Agirre", "ER03B", 1200),
              new Empleado("Kepa Sarria", "ER05C", 1000),
              new Empleado("Almudena Gamiz", "ER06D", 900),

              new Empleado("Jon Arregi", "EI01B", 2300),
              new Empleado("Diego Gómez", "EI02B", 2200),
              new Empleado("Ainhoa Elgeta", "EI03B", 1500),
              new Empleado("Begoña Besoita", "EI05C", 1200),
              new Empleado("Igor Azurmendi", "EI06A", 1900),

              new Empleado("Luis Pagei", "ME01D", 2300),
              new Empleado("Ana Larreategi", "ME02C", 1000),
              new Empleado("Kepa Bilbao", "ME03C", 1200),
              new Empleado("Kepa Ortuzar", "ME05C", 1100),
              new Empleado("Alfonso Gamiz", "ME06D", 750),

              new Empleado("Jon Punset", "DB01A", 2300),
              new Empleado("Alex Ortiz de Guinea", "DB02C", 2200),
              new Empleado("Javi Menchaca", "DB03C", 1500),
              new Empleado("Iratxe Eguíluz", "DB05A", 1200),
              new Empleado("Olga Rayón", "DB06C", 1900), };

              cargarOperaciones(vOperaciones, vEmpleados);
              asignarPuntos(vOperaciones, vEmpleados);
              puntosPorSucursalTipo(vEmpleados);
              asignarPaga(vEmpleados);
          }
```