Isabel Moreno

Actividades UT3

1. Realiza una clase *finanzas* que convierta dólares a euros y viceversa. Codifica los métodos *dolaresToEuros* y *eurosToDolares*. Prueba que dicha clase funciona correctamente haciendo conversiones entre euros y dólares.

La clase tiene que tener:

 Un constructor *finanzas()* por defecto, el cual establecerá el cambio Dólar-Euro en 1.36.

 Un constructor *finanzas(double)*, el cual permitirá configurar el cambio dólar-euro.

1. Realiza una clase *minumero* que proporcione el doble, triple y cuádruple de un número proporcionado en su constructor (realiza un método para doble, otro para triple y otro para cuádruple). Haz que la clase tenga un método *main* y comprueba los distintos métodos.
2. Realiza una clase número que almacene un número entero y tenga las siguientes características:

 Constructor por defecto que inicializa a 0 el número interno.

 Constructor que inicializa el número interno.

 Método aniade que permite sumarle un número al valor interno.

 Método resta que resta un número al valor interno.

 Método getValor. Devuelve el valor interno.

 Método getDoble. Devuelve el doble del valor interno.

 Método getTriple. Devuelve el triple del valor interno.

 Método setNumero. Inicializa de nuevo el valor interno.

1. Modificación del ejercicio resuelto número 4.

public class satelite {

private double meridiano;

private double paralelo;

private double distancia\_tierra;

satelite (double m,double p,double d){

meridiano=m;

paralelo=p;

distancia\_tierra=d;

}

satelite (){

meridiano=paralelo=distancia\_tierra=0;

}

public void setPosicion(double m,double p,double d){

meridiano=m;

paralelo=p;

distancia\_tierra=d;

}

public void printPosicion(){

System.out.println(“El satélite se encuentra en el paralelo” + paralelo+ “Meridiano” +meridiano+ “a

una distancia de la tierra de” +distancia\_tierra+ “Kilómetros”);

}

}

Modifica la clase satélite y añádele los siguientes métodos:

 Método *void variaAltura(double desplazamiento)*: Este método acepta un parámetro que será positivo o negativo dependiendo de si el satélite tiene que alejarse o acercarse a La Tierra.

 Método *boolean enOrbita()*: Este método devolverá *false* si el satélite está en tierra, y *true* en caso contrario.

 Método *void variaPosicion(double variap, double variam)*: Este método permite modificar los atributos de posición (meridiano y paralelo) mediante los parámetros variap y variam. Estos parámetros serán valores positivos o negativos relativos que harán al satélite modificar su posición.

1. Crea la clase *peso*, la cual tendrá las siguientes características:

 Deberá tener un atributo donde se almacene el peso de un objeto en kilogramos.

 En el constructor se le pasará el peso y la medida en la que se ha tomado (‘Lb’ para libras, ‘Li’ para lingotes, ‘Oz’ para onzas, ‘P’ para peniques, ‘K’ para kilos, ‘G’ para gramos y ‘Q’ para quintales).

 Deberá de tener los siguientes métodos:

– getLibras. Devuelve el peso en libras.

– getLingotes. Devuelve el peso en lingotes.

– getPeso. Devuelve el peso en la medida que se pase como parámetro (‘Lb’ para libras, ‘Li’ para lingotes, ‘Oz’ para onzas, ‘P’ para peniques, ‘K’ para kilos, ‘G’ para gramos y ‘Q’ para quintales).

Para la realización del ejercicio toma como referencia los siguientes datos:

1 Libra = 16 onzas = 453 gramos.

1 Lingote = 32,17 libras = 14,59 kg.

1 Onza = 0,0625 libras = 28,35 gramos.

1 Penique = 0,05 onzas = 1,55 gramos.

1 Quintal = 100 libras = 43,3 kg.

Crea además un método *main* para testear y verificar los métodos de esta clase.

1. Crea una clase con un método *millasAMetros()* que toma como parámetro de entrada un valor en millas marinas y las convierte a metros.

Una vez tengas este método escribe otro *millasAKilometros()* que realice la misma conversión, pero esta vez exprese el resultado en kilómetros.

Nota: 1 milla marina equivale a 1852 metros.

1. Crea la clase *coche* con dos constructores. Uno no toma parámetros y el otro sí. Los dos constructores inicializarán los atributos marca y modelo de la clase. Crea dos objetos (cada objeto llama a un constructor distinto) y verifica que todo funciona correctamente.
2. Implementa una clase consumo, la cual forma parte de la centralita electrónica de un coche y tiene las siguientes características:

 Atributos:

– kms. Kilómetros recorridos por el coche.

– litros. Litros de combustible consumido.

-vmed. Velocidad media.

– pgas. Precio de la gasolina.

 Métodos:

– getTiempo. Indicará el tiempo empleado en realizar el viaje.

– consumoMedio. Consumo medio del vehículo (en litros cada 100 kilómetros).

– consumoEuros. Consumo medio del vehículo (en euros cada 100 kilómetros).

No olvides crear un constructor para la clase que establezca el valor de los atributos. Elige el tipo de datos más apropiado para cada atributo.

1. Para la clase anterior implementa los siguientes métodos, los cuales podrán modificar los valores de los atributos de la clase:

 setKms

 setLitros

 setVmed

 setPgas

1. El restaurante mejicano de Israel cuya especialidad son las papas con chocos nos pide diseñar un método con el que se pueda saber cuántos clientes pueden atender con la materia prima que tienen en el almacén. El método recibe la cantidad de papas y chocos en kilos y devuelve el número de clientes que puede atender el restaurante teniendo en cuenta que por cada tres personas, Israel utiliza un kilo de papas y medio de chocos.
2. Modifica el programa anterior creando una clase que permita almacenar los kilos de papas y chocos del restaurante.

Implementa los siguientes métodos:

 public void addChocos(int x): Añade x kilos de chocos a los ya existentes.

 public void addPapas(int x): Añade x kilos de papas a los ya existentes.

 public int getComensales(): Devuelve el número de clientes que puede atender el restaurante (este es el método anterior).

 public void showChocos(): Muestra por pantalla los kilos de chocos que hay en el almacen.

 public void showPapas(): Muestra por pantalla los kilos de papas que hay en el almacén.