Tema 4 Clases y encapsulación

Definición de clases

1. Defínanse dentro del espacio de trabajo las siguientes clases públicas:
   a) clases.figuras.Circunferencia.
   b) clases.Persona
   c) clases.figuras.Rectangulo

Atributos

2. Defínanse los siguientes atributos en cada una de las clases:
   a) Circunferencia.
      - radio → Atributo real que representa el radio de la circunferencia. Valor por defecto de 1
   b) Persona
      - edad → Atributo entero que representa la edad de la persona. Valor por defecto de 0
      - altura → Atributo real que representa la altura del rectángulo. Valor por defecto de 0.5
      - peso → Atributo real que representa el peso de la persona. Valor por defecto de 2.5
   c) Rectángulo
      - ancho → Atributo entero que almacena el valor del ancho del rectángulo. Valor por defecto de 2.
      - alto → Atributo entero que almacena el valor del alto del rectángulo. Valor por defecto de 2
      - borde → Atributo de tipo carácter que almacena el valor del borde del rectángulo. Valor por defecto de '*'
      - relleno → Atributo de tipo carácter que almacena el valor del relleno del rectángulo. Valor por defecto de '+'

Atributos constantes

3. Defínanse los siguientes atributos en cada una de las clases:
   a) Circunferencia.
      - RADIO_INICIAL → Atributo real que representa el radio de la circunferencia al ser creada (como valor por defecto se pondrá el de por defecto del atributo radio para que el código no dé error)
   b) Persona
      - NOMBRE → Constante de cadena (como valor por defecto se pondrá la cadena "" para que el código no dé error)
      - EDAD_INICIAL → Constante entera que almacena la edad que tenia la persona al ser creada (como valor por defecto se pondrá el de por defecto del atributo edad para que el código no de error)

c) Rectángulo
- ANCHO_INICIAL → Constante entera que almacena el ancho que tenía el rectángulo al ser creado (como valor por defecto se pondrá el de por defecto del atributo ancho para que el código no dé error)
- ALTO_INICIAL → Constante entera que almacena el alto que tenía el rectángulo al ser creado (como valor por defecto se pondrá el de por defecto del atributo alto para que el código no dé error)

Métodos get

4. Defínanse los siguientes métodos get en cada una de las clases:
a) Circunferencia
- getRadio() → Devuelve el valor del radio

b) Persona
- getEdad() → Devuelve el valor del atributo edad
- getAltura() → Devuelve el valor del atributo altura
- getPeso() → Devuelve el valor del atributo peso

c) Rectangulo
- getAncho() → Devuelve el valor del atributo ancho
- getAlto() → Devuelve el valor del atributo alto
- getBorde() → Devuelve el valor del atributo borde
- getRelleno() → Devuelve el valor del atributo relleno

Métodos set.

5. Defínase los siguientes métodos set en cada una de las clases. En todos los métodos se devuelve el objeto que invoca el método y se lanzan excepciones "IllegalArgumentException" si el valor a asignar al atributo no cumpliera las restricciones indicadas.
a) Circunferencia
- setRadio(int):Circunferencia → Establece el valor del atributo "radio" siempre que sea positivo

b) Persona
- setEdad(int):Persona → Establece el valor del atributo "edad" siempre que esté entre 0 y 120, ambos inclusive)
- setAltura(double):Persona → Establece el valor del atributo "altura" siempre que tenga como máximo dos decimales y esté entre 0.3 y 2,2, ambos inclusive.
- setPeso(double):Persona → Establece el valor del atributo "peso" siempre que tenga como máximo un decimal y esté entre 1.7 y 140, ambos inclusive.

c) Rectángulo
- setAncho(int):Rectangulo → Establece el valor del atributo "ancho" siempre que sea mayor que 1.

- setAlto(int):Rectangulo → Establece el valor del atributo "alto" siempre que sea mayor que 1.
- setBorde(char):Rectangulo → Establece el valor del atributo "borde" siempre que sea alguno de los siguientes valores: '*', '+' o '-'.
- setRelleno(char):Rectangulo → Establece el valor del atributo "relleno" siempre que sea alguno de los siguientes valores: '*', '+', '-' o ' '.

Métodos set de constantes independientes

6. Defínase los siguientes métodos set de constantes en cada una de las clases

a) Persona
   - setNOMBRE(String):String → Valida el nombre de una persona. Se lanzan excepciónrd "IllegalArgumentException" si el valor del nombre a validar no fuera una palabra en mayúsculas.

Constructor

7. Defínase los siguientes constructores en cada una de las clases

a) Circunferencia
   - Circunferencia(double) → Constructor para inicializar el radio de una circunferencia. La constante RADIO_INICIAL tomará como valor el del radio.

b) Persona
   - Persona(String,int,double,double) → Constructor para inicializar el nombre, edad, altura y peso de una persona. La constante EDAD_INICIAL tomará como valor el de la edad.
   - Persona(String) → Constructor para inicializar el nombre de una persona. La constante EDAD_INICIAL tomará como valor el de la edad.
   - Persona(String, int) → Constructor para inicializar el nombre y la edad de una persona. La constante EDAD_INICIAL tomará como valor el de la edad.
   - Persona(String, double) → Constructor para inicializar el nombre y a altura de una persona. La constante EDAD_INICIAL tomará como valor el de la edad.

c) Rectángulo
   - Rectangulo(int,int,char,char) → Constructor para inicializar el ancho, el alto, el borde y el relleno de un rectángulo. Las constantes ANCHO_INICIAL y ALTO_INICIAL tomarán como valor el del ancho y alto, respectivamente.

Sobrecarga de constructores

8. Defínase los siguientes constructores en cada una de las clases

a) Persona
   - Persona(String,int,double) → Constructor para inicializar el nombre, la edad, y el peso de una persona. La constante EDAD_INICIAL tomará como valor

el de la edad. Sobrecárguese dicho constructor de todas las formas posibles invocando a dicho constructor.

b) Rectangulo
   - Rectangulo(int, int) → Constructor para inicializar el ancho y alto de un rectángulo. Las constantes ANCHO_INICIAL y ALTO_INICIAL tomarán como valor el del ancho y alto, respectivamente.
   - Rectangulo(char, char) → Constructor para inicializar el borde y el relleno de un rectángulo. Las constantes ANCHO_INICIAL y ALTO_INICIAL tomarán como valor el del ancho y alto, respectivamente.
   - Rectángulo(int, char, char) → Constructor para inicializar el ancho. el borde y relleno de un rectángulo. Las constantes ANCHO_INICIAL y ALTO_INICIAL tomarán como valor el del ancho y alto, respectivamente.
   - Rectángulo(int, int, char) → Constructor para inicializar el ancho. el alto y el borde de un rectángulo. Las constantes ANCHO_INICIAL y ALTO_INICIAL tomarán como valor el del ancho y alto, respectivamente.

Uso de this()

9. Defínase los siguientes constructores en cada una de las clases

a) Persona
   - Sobrecárguese el constructor Persona(String,int,double) definido anteriormente de todas las formas posibles invocando a dicho constructor.

b) Rectangulo
   - Sobrecárguese el constructor Rectangulo(int, char, char) definido anteriormente de todas las formas posibles invocando a dicho constructor.
   - Sobrecárguese el constructor Rectangulo(int, int, char) definido anteriormente de todas las formas posibles invocando a dicho constructor.

Constructor implícito y por defecto

10. Defínase los siguientes constructores en cada una de las clases

a) Circunferencia
   - Constructor por defecto. El RADIO_INICIAL tomará como valor el del radio

b) Persona
   - Constructor por defecto. La constante NOMBRE tomará como valor "DESCONOCIDO" y EDAD INICIAL tomará como valor el de la edad.

c) Rectangulo
   - Constructor por defecto. Las constantes ANCHO_INICIAL y ALTO_INICIAL tomarán como valor el del ancho y alto, respectivamente.

Control de errores

11. Léanse números enteros hasta introducir un 0. Se establecerá cada uno de los valores leídos distintos de 0 como radio de una circunferencia. Si no se lanza un error (ya que

el radio es positivo) se muestra el área del círculo; si se lanza un error "IllegalArgumentException" se captura y se muestra un mensaje diciendo que tiene que ser positivo. (ProvocarError)

Aplicaciones con objetos

12. Hágase una aplicación que cree una circunferencia con un radio entre 2 y 5, ambos inclusive. El programa mostrará la longitud y el área de la circunferencia redondeada a tres decimales. (RadioLongitudArea)

13. Hágase un programa que declare una persona cuyo nombre, edad, altura y peso se introduce por consola. El programa obtendrá el índice de masa corporal (peso en kg/estatura en m^2) redondeado a un decimal. (PersonaIMC)

Métodos de cálculo

14. Defínase los siguientes métodos de cálculo en cada una de las clases

a) Circunferencia
   - area():double → Se obtiene el área de la circunferencia (πr^2)
   - longitud():double → Se obtiene la longitud de la circunferencia ($2\pi r$)

b) Persona
   - imc():double → Devuelve el índice de masa corporal de una persona redondeado a un decimal (peso en kg * altura en m^2).

c) Rectangulo
   - area():int → Se obtiene el área del rectángulo (base*altura)
   - perimetro():int → Se obtiene el perímetro del rectángulo (2*base+2*altura)
   - diagonal():double → Se obtiene la diagonal del rectángulo $(base^2 + altura^2)^{1/2}$

Arrays y colecciones de objetos

15. Hágase una aplicación que genere aleatoriamente 10 circunferencias con radio real entre 1 y 10 (el 10 sin incluir). Para cada circunferencia se mostrará el valor del radio, la longitud de la circunferencia y el área del círculo. Los datos se mostrarán ordenados descendentemente por el área de la circunferencia. (ArrayRadioLongitudArea)

16. Hágase una aplicación en la que se define un ArrayList de Personas vacío. Se tendrá un menú que permita insertar personas en el ArrayList y mostrar un informe de todas las personas almacenadas en el ArrayList ordenadas por el nombre o descendentemente por la edad o ascendentemente por la altura. (ArrayListOrdenar)

17. Hágase un menú que permita insertar, eliminar, mostrar todas, ordenar (ascendentemente y descendentemente) circunferencias. (CircunferenciaMenuArray)

Métodos con parámetros de tipo un objeto

18. Créase una clase "Rebajas" en el paquete "clases" que permita descubrir el valor que tenía un producto antes de ser rebajado. La clase tendrá los siguientes miembros:

- Devolver y establecer el valor del precio del producto con la rebaja (será positivo y con dos decimales).
- Devolver y establecer el porcentaje de la rebaja (será un valor entre 10 y 80).
- Obtener el valor anterior del producto sin la rebaja (con dos decimales)
- Inicializar el precio actual y el porcentaje, sólo el precio actual y sólo el porcentaje.
- Inicialmente el precio actual será de 1 y el porcentaje de 10.

Hágase una aplicación que permita mostrar un menú para modificar el precio actual del producto, el porcentaje de rebaja y obtener el que era el precio sin la rebaja. En el último caso se mostrará también el precio actual y el porcentaje de rebaja. (RebajasMenu)

19. Se considera una pila de objetos de tipo "Rebajas". Hágase un menú que permita realizar las siguientes acciones sobre tal pila:

1. Apilar
2. Desapilar
3. Ultima (rebaja con el valor anterior)
4. Vacía
5. Mostrar (cada rebaja con el valor anterior)
6. Vaciar

(MenuPilaRebajas)

Métodos de consulta

20. Defínase los siguientes métodos de consulta en cada una de las clases

a) Circunferencia
   - esUnitaria():boolean → Comprueba si el radio es o no 1.

b) Persona
   - serInfantil():boolean → Devuelve verdadero si la edad está comprendida entre 0 y 6
   - serNiño():boolean → Devuelve verdadero si la edad está comprendida entre 7 y 11.
   - serAdolescente():boolean → Devuelve verdadero si la edad está comprendida entre 12 y 18.
   - serJoven():boolean → Devuelve verdadero si la edad está comprendida entre 19 y 25.
   - serAdulto():boolean → Devuelve verdadero si la edad está comprendida entre 26 y 64.

- serAnciano() → Devuelve verdadero si la edad es superior o igual a 65

c) Rectangulo
- esCuadrado():boolean → Comprueba si el rectángulo es o no un cuadrado (el ancho y el alto coinciden).

Métodos de cambio de estado

21. Defínase los siguientes métodos de cambio de estado en cada una de las clases. En todos los métodos se devuelve el objeto que invoca el método.

a) Circunferencia
- duplicar():Circunferencia → Duplica el radio de la circunferencia.

b) Persona
- cumplir():Persona → Incrementar la edad de la persona en 1 año
- engordar(double):Persona → Aumentar el peso de la persona en la cantidad indicada. Se lanzará una excepción "IllegalArgumentException" si el valor a aumentar fuera negativo.
- adelgazar(double):Persona → Disminuir el peso de la persona en la cantidad indicada. Se lanzará una excepción "IllegalArgumentException" si el valor a disminuir fuera negativo.

c) Rectangulo
- plus():Rectangulo → Incrementa en una unidad tanto el ancho como el alto.
- invertir():Rectangulo → Intercambia el valor del ancho y del alto.
- restaurar():Rectangulo → Restaura el rectángulo a su ancho y alto inicial.

Métodos de conversión de cadenas

22. Defínase los siguientes métodos de conversión de cadenas en cada una de las clases. En todas las clases se definirá el método "toString()" devolviendo una cadena con el valor de cada una de los atributos de objeto precedidos por el nombre del atributo.

a) Circunferencia
- toString():String

b) Persona
- toString():String

c) Rectangulo
- dibujar():String → Obtiene un rectángulo en formato de cadena.
- dimension():String → Obtiene una cadena con el formato "ancho x alto".
- toString():String

Hágase una aplicación en la que se introduzca el ancho, alto, borde y relleno de un rectángulo, y se muestre por consola el valor de la cadena devuelta por el método "toString" para dicho rectángulo. (ConversionACadena)

Sobrecarga de métodos de clase

23. Defínase los siguientes métodos sobrecargados en cada una de las clases.

a) Circunferencia
   - toString(int):String → Sobrecarga del método "toString()" que muestra todos los valores reales con "n" (valor del parámetro) decimales. Se lanzará una excepción "IllegalArgumentException" si el valor del parámetro fuera negativo.

b) Persona
   - engordar():Persona → Sobrecarga del método "engordar(String)" que aumenta el peso de la persona en 1 kg.
   - adelgazar():Persona → Sobrecarga del método "adelgazar(String)" que disminuye el peso de la persona en 1 kg.

c) Rectangulo
   - dibujar(String):String → Sobrecarga del método "dibujar()" que hace lo mismo pero poniendo antes el mensaje.
   - diagonal(int):double→ Sobrecarga del método "diagonal()" que obtiene la diagonal del rectángulo con "n" (valor del parámetro) decimales. El valor del parámetro se convertirá en positivo previamente al cálculo, si fuera negativo.

Aplicaciones con objetos II

24. Hágase una aplicación en la que se lean circunferencias hasta introducir una de radio 1. El programa mostrará el radio, la longitud y el área de cada una de las circunferencias generadas. (LeerCircunferencias)

25. Hágase una aplicación en la que se introduzca los datos de una persona (nombre, edad, altura y peso). Una vez introducida la persona, se la hará cumplir 3 años, se incrementará su peso en 10 Kg, se indicará en qué etapa de la vida se encuentra (infantil, niñez, adolescencia, juventud, adultez o ancianidad) y el valor de su índice de masa corporal. (PersonaAplicacion)

26. Hágase un programa que permita introducir por consola el ancho y alto de un rectángulo. El programa obtendrá el área (base*altura) y el perímetro (2*base + 2*altura) de ese rectángulo, además de la longitud de la diagonal (raíz cuadrada $base^2+altura^2$). (RectanguloAreaPerimetroDiagonal)

27. Hágase una aplicación que lea el ancho y alto de un rectángulo. Lo dibuje, lo invierta, lo muestre, incremente una unidad tanto el ancho como el alto, lo muestre, y compruebe si es o no un cuadrado. (RectanguloAplicacion)

28. Hágase una aplicación en la que se cree un rectángulo de 4x7. Se modifique su ancho y alto introduciendo tales datos por consola; se dibuje; se restaure y se vuelva dibujar. (RectanguloAplicacionConstantes)

Enumerados

29. Defínase en el paquete "clases/enumerados" el enumerado "Mes" con los meses del año.

30. Defínase el siguiente enumerado:

PAQUETE: clases.enumerado ENUMERADO: Semana
MIEMBROS
lunes
martes
miercoles
jueves
viernes
sabado
domingo

Uso de enumerados en clases

31. Hágase un programa que lea del tipo de dato enumerado "Mes" un mes como numérico (del 1 al 12), y se muestre el nombre del mes en mayúsculas. (MesEnumerado)

Enumerados II

32. Defínase la siguiente clase:

PAQUETE: clases CLASE: Repostar			
ACCESO	TIPO	MIEMBROS	
privado	Semana	dia	0
privado	double	cantidad	1
privado	double	precio	0,5
publico	Semana	getDia()	
publico	double	getCantidad()	
publico	double	getPrecio()	
publico	Repostar	setDia(Semana)	
publico	Repostar	setCantidad(double)	
publico	Repostar	setPrecio(double)	
publico		Repostar(Semana,double,double)	
publico		Repostar(double,Semana,double)	
publico		Repostar(double,double,Seaman)	
publico	double	coste(int)	
publico	double	coste()	

- setCantidad → Se lanza la excepción "IllegalArgumentException" si el valor a asignar a "cantidad" fuera menor o igual que 1
- setPrecio → Se lanza la excepción "IllegalArgumentException" si el valor a asignar a "precio" fuera menor que 0,5
- coste → Obtiene el coste (producto de cantidad*precio)
- coste(int) → Obtiene el coste redondeado a los decimales introducidos (>0 hacia la parte decimal; <0 hacia la parte entera)

Hágase una aplicación para que se introduzcan los datos de repostaje de una persona, y luego se muestre el día de la semana que repostó y cuánto dinero pagó. (RepostarAplicacion)

33. Créese una clase de nombre "Consumición" en el paquete "clases" que permita determinar el coste de una determinada consumición. Se definirán los siguientes enumerados y métodos:

- Se tendrá un enumerado "Articulos" con los distintos tipos de productos (caña, refresco, pincho, tapa, bocadillo, vino, etc)
- Devolver y establecer el artículo.
- Devolver y establecer el precio (valor con dos decimales positivo)
- Devolver y establecer el número de unidades (valor positivo)
- Obtener el coste de la consumición.
- Inicializar el artículo, el precio y número de unidades de todas las posibles formas.
- Inicializar el articulo y el precio (en cualquier orden); el artículo y el número de unidades (en cualquier orden); y el precio y el número de unidades (en cualquier orden)
- Inicialmente el artículo toma el valor de "caña"; el precio, "1.00" y el número de unidades, "1".

Hágase una aplicación en la que se genere 10 consumiciones aleatorias válidas. El artículo de cada consumición se generará de forma aleatoria entre todas las posibles consumiciones. El precio de cada consumición será un valor aleatorio entre 0.01 y 10.00, ambos inclusive. El número de unidades de cada consumición será un valor aleatorio entre 1 y 20, ambos inclusive. Se mostrará un informe con cabeceras con el artículo, precio, número de unidades y coste de cada consumición, y una línea final con el total de la consumición. (ConsumicionAplicacion)

34. Un señor cada vez que reposta en la gasolinera apunta el día de la semana (lunes, martes, miércoles, jueves, viernes, sábado y domingo), la cantidad repostada (entre 0.1 a 50.0 litros) y el precio del litro de gasolina (entre 0.001 y 1.500). Se validarán los datos introducidos. El señor puede obtener los siguientes informes con sus correspondientes cabeceras de inicio y fin:

a) El gasto de cada uno de los días en los que repostó (litros, precio y gasto con dos decimales), y una línea de total gasto.

b) El gasto para cada uno de los días de la semana (el día de la semana, el total gastado con dos decimales, el precio medio por litro, el total de litros repostado y el gasto con dos decimales correspondiente del total de litros al precio medio, y la diferencia con el total de gasto real.
c) Número de veces que reposto una cantidad entre 0 y 10 litros, entre 10.1 y 20 litros, entre 20.1 y 30 litros, entre 30.1 y 40 litros, y entre 40.1 y 50 litros.

Hágase un menú para que el señor pueda repostar y mostrar cada uno de los informes. Utilícese la clase "Repostar". (RepostarArray)

Enumerados con constructor

35. Hágase una aplicación en la que se genere de manera aleatoria un valor de tipo "ColorPelo". Muéstrese por consola el valor de la eumelanina y feomelanina para ese color de pelo. (ColorPeloComposicion)

36. Defínase un tipo enumerado de nombre "Materiales" en "clases/enumerados" con los siguientes atributos, métodos y constantes:

- precio → Real. Con valor inicial de 1. Precio por m^2.
- calidad → Entero. Con valor inicial de 1.
- Un constructor sin modificador de acceso para inicializar el precio y la calidad, en cualquier orden.
- Las siguientes constantes:

MATERIAL	PRECIO	CALIDAD
MADERA	0.5	4
LATON	1.2	5
HIERRO	1.8	7
ACERO	2.25	9

(PlacaAplicacion)

Clonar objetos

37. Sobrescríbase en las clases "Circunferencia", "Persona", "Rectangulo", "Repostar", "Consumición" y "Rebajas el método "clone" para realizar una copia superficial.

Hágase una aplicación en la que se introduzca una rebaja, se clone, y se modifique la segunda incrementado el precio actual en una unidad. Muéstrese el precio anterior antes de la rebaja para los objetos anteriores. (Clonar)

38. Hágase la siguiente aplicación:

a) Se genera aleatoriamente 10 circunferencias de radios aleatorios entre 1y 10.
b) Se mostrarán las 10 circunferencias.

c) Se obtendrán un array que sea un clon del anterior. A la primera circunferencia se modifica su radio a 20.
d) A cada circunferencia del primer array se le acumulará el radio de la siguiente, salvo a la última que se le acumulará la de la primera. Se vuelven a mostrar las 10 circunferencias.
e) Se vuelven a mostrar las 10 circunferencias.
f) Se mostrarán las circunferencias del array clonado.

(CircunferenciaArray)

Sobreescribir el método equals

39. Sobrescríbase en las clases "Circunferencia", "Persona", "Rectangulo", "Repostar", "Consumición" y "Rebajas" el método "equals" de forma que dos objetos en dichas clases sean iguales si coinciden todos sus atributos.

40. Considérese un conjunto hash de objetos de la clase "Rectangulo". Procese dicho conjunto hash con un menú que permita insertar (rectángulos distintos) en la colección, mostrar todos los rectángulos (dibujándolos) de la colección, eliminar (un rectángulo) de la colección, vaciar la colección y mostrar el número de rectángulos que hay en la colección. (MenuHashSetRectangulos)

Comparar objetos

41. Impleméntese la interfaz "Comparable<T>" en las clases "Circunferencia", "Persona", "Rectangulo" "Repostar", "Consumición" y "Rebajas comparando los objetos por el radio, por el nombre, por el porcentaje de rebaja, por el coste de la consumición, por el coste del repostaje y por el área, respectivamente.

42. Defínase la siguiente clase "Mascota":

PAQUETE: poo CLASE: Mascota IMPLEMENTA: Comparable<Mascota>			
ACCESO	TIPO	MIEMBROS	
publico	enumerado	Animal	
private	String	nombre	""
private	Animal	animal	1
publico	String	getNombre()	
publico	Animal	getAnimal()	
publico	Mascota	setNombre(String)	
publico	Animal	setAnimal(Animal)	
publico		Mascota(String, Animal)	
publico		Mascota(String)	
publico		Mascota(Animal)	
publico		Mascota(Mascota)	
REEMPLAZAMIENTO DE METODOS			
publico	String	toString()	

publico	Mascota	clone()	
publico	int	compareTo(Mascota)	
publico	booelan	equals(Object)	

- Anumal → Enumerado con los valores “perro”, “gato”, “tortuga”, “hámster”, “pájaro”, “pez”
- nombre → Atributo de tipo cadena que almacena el nombre de la mascota. Inicailizado a “”
- animal → Atributo de tipo Animal que almacena el tipo de animal. Inicializado a “perro”
- getNombre() → Devuelve el valor del atributo “nombre”
- getAnimal() → Devuelve el valor del atributo “animal”
- setNombre(String) → Establece el valor del atributo “nombre” al valor pasado. Se lanza un erro de tipo “llegalArgumentException” si no es una palabra o la cadena vacía.
- setAnimal(Animal) → Estable el valor del atributo “animal” al valor pasado.
- Mascota(String, Animal) → Inicializada una mascota con su nombre y tipo de animal.
- Mascota(String) → Inicializada una mascota con su nombre.
- Mascota(Animal) → Inicializada una mascota con el tipo de animal.
- Mascota(Mascota) → Constructor de copia.
- toString() → Obtiene una cadena con los valores de los atributos precedidos de un texto en líneas distintas.
- clone() → Clona la mascota que invoca el método.
- compareTo(Mascota) → Compara dos mascotas por su nombre.
- equals(Object) → Comprueba si dos mascotas son o no iguales en función del valor de su nombre.

(MascotaComparar)

Atributos de objeto

43. Defínase la clase “ClienteBar” con las siguientes características y métodos:

- Nombre de la persona. Inicialmente vale la cadena vacía.
- Consumición realizada. Inicialmente vale “null”.
- Devolver y establecer el nombre de la persona. El nombre de la persona será una palabra que comienza por mayúsculas y el resto en minúsculas o la cadena vacía. Se lanzará un error del tipo “IllegalArgumentException” si no fuera así.
- Devolver y establecer la consumición realizada.
- Constructor para poder inicializar el nombre de la persona y la consumición.
- Constructor para poder inicializar sólo el nombre de la persona.
- Constructor para poder inicializar sólo la consumición.
- Constructor por defecto

- Obtener una línea de informe con las siguientes características: nombre de la persona con un tamaño de 20 y alineado a la izquierda; tipo de la consumición con un tamaño de 10 y alineado a la izquierda; unidades con un tamaño de 8; precio con un tamaño de 6 y dos decimales; y coste con un tamaño de 8 y dos decimales.
- Reemplazar el método "toString" obteniendo una cadena con el nombre de la persona precedida por el texto "nombre" y si la consumición no es null, un salto de línea seguido de la conversión a cadena con el método "toString" del valor de la consumición.
- Reemplazar el método "clone".
- Implementar la interfaz "Comparable" comparando los objetos por el nombre.
- Reemplazamiento del método "equals" de forma que dos objetos son iguales si coincide sus consumiciones.

Hágase una aplicación en la que se introduzca tres objetos de tipo "ClienteBar". Se hará lo siguiente:

a) Clonar el primer objeto "ClienteBar" introducido en un cuarto objeto. Incrementar en 1 las unidades de la consumición de este nuevo objeto.
b) Obtener un informe con cabecera y pie de los cuatro objetos con las líneas que devuelve el método definido en la clase.
c) Comprobar si son o no iguales el primer y cuarto objeto.

(ClienteBarAtributoObjeto)

44. Modifíquese la clase "Persona" añadiendo las siguientes características y métodos:

- Atributo "mascota" con el valor inicial de "null"
- Devolver y establecer el valor del atributo "mascota"
- Constructor que inicializa el nombre de la persona y su mascota.
- Modificar el constructor de copia para que se copie también la mascota.
- Modificar el método "toString" para que muestre además la mascota si la tuviera.

Hágase una aplicación que haga lo siguiente:

a) Crear una mascota "m" de tipo tortuga con el nombre "Ati".
b) Crear una persona "p1" con el nombre de "Carlos" y como mascota "m".
c) Obténgase una persona "p2" que sea un clon de "p1".
d) Compárese las mascotas "m" y la mascota de "p1".
e) Modifíquese el nombre de la mascota "p1".
f) Compárese las mascotas "m" y la mascota de "p2"
g) Muéstrese las personas "p1" y "p2".

(PersonaAtributoObjeto)

45. Considéres la clase "Alumnos" añadiendo las siguientes características y métodos:

- Nombre del alumno. Cadena de un máximo de 20 caracteres con palabras separadas por un espacio y cuyas palabras tienen la primera letra en mayúsculas y el resto en minúsculas.
- Una lista de notas de tipo entero. La lista no puede ser null, aunque puede estar vacía. Las notas tomarán valores entre 0 y 10, ambos inclusive.
- Devolver y establecer el nombre del alumno.
- Reemplazar el método "clone"
- Devolver y establecer la lista de notas.
- Devolver y establecer el valor de una nota.
- Constructor que permita inicializar el nombre del alumno y la lista de notas a vacia.
- Añadir una nota a la lista.
- Eliminar una nota en una determinada posición de la lista
- Eliminar una nota de la lista (en todas las posiciones donde aparezca)
- Obtener el número de sobresalientes (notas superiores a 8)
- Obtener el número de suspensos (notas inferiores a 5)
- Obtener la media de las notas con un decimal.
- Reemplazar el método "toString" mostrando el nombre del alumno y cada una de sus notas en líneas distintas y con un mensaje previo del tipo "NOMBRE: ", "NOTA 1: ", NOTA 2: ", etc

Hágase una aplicación en la que se define una lista de alumnos. Se tendrá un menú con las siguientes opciones:

a) Añadir un alumno a la lista (que no se repita su nombre)
b) Añadir una nota a un alumno de la lista.
c) Modificar una nota de un alumno.
d) Eliminar una nota de un alumno en una determinada posición.
e) Clonar el primer alumno, modificar el nombre del alumno clonado y añadirlo a la lista de alumnos.
f) Mostrar un informe con todas las notas de cada alumno.
g) Mostrar un informe con las notas medias de cada alumno, el número de sobresalientes y el número de suspensos.
h) Mostrar para un alumno sus notas. Si no tuviera notas se mostrará un mensaje del tipo "No tiene notas"

(MenuNotas)

46. Considérese una clase "Alumno" en "clases" con las siguientes características y métodos:

- Constante estática con los nombres de los módulos de primero de DAM
- Constante estática con las siglas de los módulos de primero de DAM
- Nombre y apellidos. Inicialmente toma la cadena vacía.

- Notas finales de los módulos. Todos los módulos tienen una nota de 1.
- Devolver y establecer el nombre y los apellidos. El formato será "Apellido2 Apellido2, Nombre" o la cadena vacía.
- Asignar las notas. Se validará que toman valores enteros entre 1 y 10, ambas inclusive.
- Devolver una copia de las notas y de forma privada devolver las notas.
- Modificar una nota de uno de los módulos
- Obtener una nota de uno de los módulos.
- Asignar la misma nota a todos los módulos.
- Comparar dos alumnos por su nombre completo.
- Dos alumnos serán iguales si tienen el mismo nombre.
- Buscar un alumno en un array dinámico de alumnos.
- Obtener la media de todas las notas de los módulos con 1 decimal.
- Obtener un informe para ese estudiante con las notas finales de cada módulo y su nombre, y con la nota final.
- Método estático que permita leer una nota válida (entre 1 y 10, ambos inclusive).

Hágase un menú para actualizar alumnos almacenados en un array dinámico. El menú tendrá las siguientes opciones:

a) Insertar alumnos (sólo el nombre y apellidos)
b) Modificar todas las notas de un alumno
c) Asignar una nota a todos los módulos de un alumno
d) Mostrar un informe con las notas finales y media de un alumno
e) Mostrar las notas de todos los alumnos ordenadas por el nombre
f) Mostrar las notas de todos los alumnos ordenados por su nota media de forma descendente.

(AlumnoMenu)

47. Un señor va a una cierta gasolinera a repostar. Anota en una libreta sus repostajes (día de la semana, litros y precio del litro). Créese una clase "LibretaRepostaje" que simule la libreta de dicho señor. En dicha clase se tendrá un atributo de tipo array dinámico "Repostajes" inicializado a vacío. El señor podrá añadir un nuevo repostaje, modificar un repostaje que hizo un día determinado (el primer dia -1-, el segundo día -2-, etc), eliminar un repostaje de un día determinado, añadir un repostaje antes de un determinado día, eliminar todos aquellos repostajes que superen un determinado coste, vaciar la libreta de repostajes, hacer una copia de seguridad de la libreta actual, restaurar la libreta actual a la última copia de seguridad y mostrar un informe con cada uno de los repostajes y el gasto total de la libreta actual. Defínanse métodos en la clase para obtener las acciones anteriores, y hágase un menú que permita realizar cada una de las opciones anteriores. (LibretaMenu)

48. Añádase a la clase "Persona" las siguientes características y métodos:

ACCESO	TIPO	MIEMBROS		EXCEPCIONES
privado	int[]	cuentas	*	(*) 10 cuentas a 0
privado	int[]	getCuentas()		
publico	n	lengthCuentas()		
publico	int	getCuentas(int[])		
publico	Persona	setCuentas(int, int)		IllegalArgumentException
publico	Persona	setCuentas(int[])		IllegalArgumentException
publico	Persona	Persona(Persona)		
publico	String	toString()		

- cuentas → Atributo de array de 10 enteros inicializado a 0
- getCuentas() → Método privado que devuelve el valor del atributo "cuentas"
- lengthCuentas() → Devuelve el número de cuentas
- getCuentas(int) → Devuelve el valor de "iesima" cuenta.
- setCuentas(int[]) → Establece el valor del atributo "cuentas" al del parámetro clonándolo. Se lanzará un error del tipo de tipo "IllegalArgumentException" si el array no tuviera 10 componentes o el valor de cada componente no fuera positivo o 0.
- setCuentas(int, int) → Establecer en una posición del atributo "cuentas" el valor de una cuenta. Se lanzará un error del tipo de tipo "IllegalArgumentException" si la posición estuviera fuera del rango válido o el valor de la cuenta fuera negativo.
- Persona(Persona) → Se modificará para clonar las cuentas de una persona.
- toString() → Se modificará para convertir a cadenas las cuentas.

Hágase una aplicación que cree una persona de nombre "Pedro". Se rellenará de forma aleatoria con enteros entre 100 y 200 todas las cuentas de "Pedro". Se mostrará por consola cada una de las cuentas de "Pedro". (PersonaCuentas)

49. Almacénese en un array dinámico líneas correspondientes a una factura. Cada línea tendrá el nombre del artículo (20 caracteres como máximo), el precio (entre 0 y 500.00) y el número de unidades (entre 0 y 10). Se podrá actualizar (eliminar, insertar y modificar) cualquiera de las líneas de la factura. Se podrá recorrer todas las líneas de una factura (primero, último, siguiente, anterior y buscar). Se podrá mostrar un informe con el coste de cada artículo y el total de la factura a pagar. Créese una clase "Linea" y una clase "Factura" (con el código de la factura y un array dinámico de líneas) y utilícense para resolver el problema.

Métodos estáticos

50. Añádese a la clase "Circunferencia" los siguientes miembros estáticos:

- UNO → Constante estática con el valor de una circunferencia de radio 1.

- crear(double) → Devuelve una circunferencia de radio el valor pasado al método.
- sumar(Circunferencia, Circunferencia):Circunferencia → Devuelve una circunferencia cuyo radio es la suma de los radios de las circunferencias dadas.
- producto(Circunferencia, Circunferencia):Circunferencia → Devuelve una circunferencia cuyo radio es el producto de los radios de las circunferencias dadas.
- esUNO(Circunferencia):boolean → Devuelve verdadero si una circunferencia es la circunferencia UNO.

Hágase una aplicación que lea circunferencias hasta introducir la circunferencia UNO. De haber introducido más de una circunferencia, se obtendrá la circunferencia resultante de sumar la primera y la última, y se mostrará.

(CircunferenciaEstaticos)

51. Añádase a la clase "Persona" los siguientes miembros:

- "NUMERO_DE_OJOS" → Constante estática con el valor de 2
- "ESPERANZA_DE_VIDA → Constante estática cuyo valor se extraerá en su momento del fichero "fic/esperanzaDeVida.txt". Ahora se obtendrá de forma aleatoria en cada ejecución con un valor entre 80 y 90, ambos inclusive.
- static → Constructor estático para inicializar el valor de la constante ESPERANZA_DE_VIDA.
- leer():int → Método estático que en su momento extraerá del fichero "fic/espearnzaDeVida.txt" el valor de la esperanza de vida y lo devuelve. Ahora devolverá un número aleatorio entre 80 y 90, ambos inclusive.

52. Defínase en la clase "Circunferencia" un método público estático que obtenga la mayor de las circunferencias que se pasan como argumentos. Hágase una aplicación en la que se lean tres circunferencias, y se muestre el área de la mayor de todas. (CircunferenciaMayor)

Métodos y constantes estáticas

53. Añádase a la clase "Persona" los siguientes miembros:

- totalEdad → atributo entero privado estático que almacena la suma de las edades de los objetos de persona existentes.
- getTotalEdad():int → Método estático que devuelve el valor del atributo totalEdad
- setTotalEdad(int):void → Método estático privado que establece el valor del atributo totalEdad al valor del parámetro. De ser dicho parámetro negativo, se lanzará una excepción del tipo IllegalArgumentException.

- setEdad(int):Persona → Se modificará dicho método para dejar un valor coherente del atributo totalEdad.

Clase Comparator

En los siguientes ejercicios se harán aplicaciones para introducir dos objetos de la clase en cuestión, y compararlos de todos los modos posibles.

54. Defínase constantes estáticas de tipo Comparator en las siguientes clases:

a) Circunferencia. Criterios de comparación: radio (CircunferenciaComparator)
b) Persona: Criterios de comparación: nombre, edad, altura, peso. (PersonaComparator)
c) Consumición. Criterios de comparación: precio, número de unidades, coste de la consumición (ConsumicionComparator)
d) Rebajas. Criterios de comparación: precio del producto sin la rebaja, precio del producto con la rebaja, porcentaje de rebaja. (RebajasComparator)
e) Rectángulo: Criterios de comparación: ancho, alto, área, perímetro, área descendente y perímetro ascendente. (RectanguloComparator)
f) Repostar. Criterios de comparación: día de la semana, cantidad de litros repostados, precio del litro. (RepostarComparator)
g) ClienteBar: Criterios de comparación: nombre del cliente o el coste de la consumición

Finalizador y método de close

55. Añádase a la clase "Persona" los siguientes miembros:

- fin → Atributo finalizador. En el finalizador se actualizará correctamente el valor del total de las edades de los objetos vivos.
- Modifiquese el método "clone" para que se mantenga correctamente el total de la edad de los objetos vivos.
- close()→ Método de la interfaz AutoCloseable que invoca el método para limpiar el objeto.

Hágase una aplicación en las que se lean personas (nombre y edad) hasta introducir una persona cuya edad coincide con la de la esperanza de vida. Se hará lo siguiente:

a) Después de leer cada persona se mostrará el valor del atributo "totalEdad".
b) Se modificará la edad de la primera persona haciendo que cumpla años dos veces.
c) Se mostrará el valor del atributo "totalEdad"
d) Se contabilizará el número de personas cuya edad es inferior a la de la ESPERANZA_DE_VIDA
e) Para cada una de las personas se hará los siguiente: se le asignará el valor de null, se invocará el recolector de basura, se detendrá l segundo el programa, y se mostrará el valor del atributo "totalEdad"

(PersonaEstaticos)

56. Defínase una clase “Placa” en “clases” con las siguientes características y métodos:

- Constante estática con el valor de ancho y alto mínimo.
- Constante estática con el valor del ancho y alto máximo.
- Constructor estático para leer el mínimo y máximo del ancho y alto. Si no se sabe procesar ficheros, el mínimo será un valor aleatorio entre 3 y 10, ambos inclusive; y el máximo, un valor aleatorio entre 80 y 100, ambos inclusive.
- Constantes con los valores iniciales de la definición del objeto.
- La inscripción o texto.
- Ancho de la placa.
- Alto de la placa.
- Material en el que está elaborada la placa.
- Devolver y establecer el texto. El texto son palabras en mayúsculas separadas por un espacio. Lanzar excepciones cuando proceda.
- Devolver y establecer el ancho de la placa. El ancho es un valor entre el mínimo y máximo, ambos inclusive. Lanzar excepciones cuando proceda.
- Devolver y establecer el alto de la placa. El alto es un valor entre el mínimo y el máximo, ambos inclusive. Lanzar excepciones cuando proceda.
- Devolver y establecer el material.
- Constructor para inicializar el texto, el ancho, el alto y el material.
- Restaurar la placa a sus valores iniciales.
- Reemplazar el método “clone”
- Reemplazar el método “toString” devolviendo una cadena con cada uno de los atributos en líneas distintas. Para cada atributo se pondrá un texto previo describiéndolo. En el caso del material, se mostrará también el precio y la calidad.
- Implementar la interfaz “Comparable” comparando dos objetos por su texto.
- Reemplazar el método “equals” de forma que dos objetos son iguales si tienen el mismo texto.
- Método limpiador o finalizador que muestre el objeto que se está eliminando con una línea previa de “********* PLACA DESTRUIDA **********” y una línea final “**********”.

Hágase una aplicación en la que se introduzca dos objetos de tipo “Placa”, se comparen por los distintos criterios, se comprueban si son o no iguales, se asigna el valor null a los dos objetos y se invoca el recolector de basura deteniendo la ejecución del programa el tiempo suficiente para ver la salida del método “finalize”. (PlacaAplicacion)

Gestores de eventos de cambio y cambiándose

57. Defínase para cada una de las clases siguientes los gestores de eventos de cambio y cambiándose para cada uno de los atributos indicados:

a) Circunferencia: radio
b) Persona: edad, peso, altura
c) Rectángulo: ancho, alto, borde, relleno

Escuchadores asociados a objetos

58. Defínase en una aplicación dos objetos de tipo Circunferencia cuyos radios sean 1 y 2, respectivamente, y un rectángulo de ancho 3, alto 5, borde "+" y relleno " ". En la aplicación se hará lo siguiente:

a) El evento de que vaya a cambiarse el valor del radio de sendas circunferencias será escuchado por el mismo escuchador. El controlador asociado a dicho escuchador cancelará la asignación del nuevo radio si fuera menor que el que tiene actualmente el objeto Circunferencia, y se mostrará por consola un mensaje indicándolo.
b) El evento de que se haya cambiado el valor del radio de sendas circunferencias será escuchado por el mismo escuchador. El controlador asociado a dicho escuchador incrementará el ancho del rectángulo en una unidad, si el objeto que lanza el evento es el primer objeto circunferencia; y se incrementará en una unidad el alto del rectángulo, si el objeto que lanzará el evento fuera el segundo objeto.
c) El evento de que se haya cambiado el valor del ancho o alto del rectángulo será escuchado por el mismo escuchador, En el controlador asociado a dicho escuchador mostrará por consola el dibujo del rectángulo.
d) En la aplicación principal se genera de forma aleatoria 10 números aleatorios entre 0 y 10, ambos inclusive. Para cada valor generado se decide de forma aleatoria cuál de los dos objetos Circunferencia tendrá como nuevo radio dicho valor.

(EscuchadoresEventos)

Documentar los miembros de una clase

59. Créese un proyecto "Documentar" con el enumerado "Semana" y la clase "Repostar". Documéntese todos los miembros de dicho enumerado y de la clase "Repostar". Genérese con "javadoc" los correspondientes archivos web en una carpeta "doc" en dicho proyecto. Obténgase en dicho proyecto un archivo .jar de la "Repostar" (el nombre del archivo será "repostar.jar". (Documentar)