UAX

Ejercicios Tema 3

Somos
#**UAX**makers

UAX

1 Ejercicios

1. (**5 minutos**) Escribe un programa que cree una clase para representar un objeto punto en tres dimensiones. Proporcionar un constructor que inicialice los valores del punto al origen de coordenadas y otro que permita especificar las coordenadas del punto. Sobrescribe su método toString() para que muestre información sobre los puntos. Usa la clase en un programa donde crees objetos que representen los puntos (12, 13, 18) y (8, 14, 0) y los muestres por consola.
2. (**5 minutos**) Crea una clase fecha que almacene el día, el mes y el año de una fecha. Proporciona funciones miembro para acceder a estos atributos (getDia(), getMes() y getAño() y para modificarlos (setDia(int dia), setMes(int mes) y setAño(int año)). Sobreescribe su método toString(). Crea la fecha 20/10/2018. Muéstrala por pantalla. Después cambia el año 2019. Muéstrala por pantalla.
3. (**20 minutos**) Crear una clase que represente un número racional que permita, al menos, sumar, multiplicar y simplificar números racionales. Proporcionar un constructor por defecto, un constructor de copia (esto es, un constructor al que se le pasa una instancia de la clase número racional y crea otro número racional idéntico), y otro que permita indicar los valores del

numerador y del denominador. Usando esta clase, crea una calculadora que permita operar con números racionales, seleccionando las operaciones de un menú.

4. (**10 minutos habiendo hecho el anterior. Si no, 20 minutos**) Repetir el ejercicio anterior, pero creando una clase que represente a un número complejo.
5. (**30 minutos**) Haz una clase llamada Persona con atributos: nombre, edad, DNI, sexo (usa una enumeración), peso y altura. Crea métodos para acceder y modificar todos los atributos.

Por defecto, todos los atributos menos el DNI tendrán valores por defecto según su tipo (0 números, cadena vacía para String, etc.). Sexo será mujer por defecto. La clase deberá tener los siguientes constructores constructores:

- Un constructor por defecto.
- Un constructor con el nombre, edad y sexo, el resto por defecto.
- Un constructor con todos los atributos como parámetro, menos el DNI.

La clase deberá tener los siguientes métodos:

- calcularIMC(): calcula el índice de masa corporal de la persona (peso en kg/(altura^2 en m))
- valorarPesoCorporal() devuelve un -1 si está por debajo de su peso ideal, un 0 si está en su peso ideal y un 1 si tiene sobrepeso. Sobrepeso se define como IMC > 25 y se considera que se está por debajo del peso ideal si IMC <18.
- esMayorDeEdad(): indica si es mayor de edad, devuelve un booleano.
- toString(): devuelve toda la información de la persona como una cadena de caracteres.
- generaDNI(): genera un número aleatorio de 8 cifras que será el DNI de la persona. Este método no será visible desde el exterior. Este método deberá invocarse desde cualquier constructor para generar el DNI.
- Métodos set de cada parámetro, excepto de DNI.

Ahora, crea una clase ejecutable que haga lo siguiente:

- Pide por teclado el nombre, la edad, sexo, peso y altura.
- Crea 3 objetos de la clase anterior, el primer objeto obtendrá las anteriores variables pedidas por teclado, el segundo objeto obtendrá todos los anteriores menos el peso y la altura y el último por defecto, para este último utiliza los métodos set para darle a los atributos unvalor.
- Para cada objeto, se deberá comprobar si está en su peso ideal, tiene sobrepeso o por debajo de su peso ideal con un mensaje.
- Indicar para cada objeto si es mayor de edad.
- Por último, mostrar la información de cada objeto.

6. (**20 minutos**) Haz una clase llamada Password que tenga los atributos longitud y contraseña . Por defecto, la longitud será de 8. Los constructores serán los siguiente:

- Un constructor por defecto que tendrá como contraseña "password".
- Un constructor con la longitud que nosotros le pasemos. Generará una contraseña aleatoria con esa longitud.

Los métodos de esta clase serán:

- esFuerte(): devuelve un booleano si es fuerte o no, para que sea fuerte debe tener más de 2 mayúsculas, más de 1 minúscula y más de 5 números.
- generarPassword(): genera la contraseña del objeto con la longitud que tenga.
- Método get para contraseña y longitud.
- Método set para longitud.

Ahora, crea una clase clase ejecutable:

- Cree un array de Passwords con el tamaño que tú le indiques por teclado.
- Cree un bucle que cree un objeto para cada posición del array. Indica por teclado la longitud de cada password.

- Crea otro array de booleanos donde se almacene si el password del array de Password es o no fuerte (usa el bucle anterior).
- Al final, muestra la contraseña y si es o no fuerte (usa el bucle anterior). Usa este simple formato:

contraseña1 valor_booleano1

contraseña2 valor_bololeano2

7. (**5-unlimited minutos**. Depende del control del código del ejemplo.) Usando las clases del código de ejemplo de los marcianos, construye una guerra donde combatan 5 naves de los marcianos y 10 naves de los terrícolas.
8. (**15 minutos**) Usando las clases del código de ejemplo del resolvedor de raíces, añade una nueva función que permita representar un polinomio de grado n. Crea esta función en tu propio paquete. Utilizando el resolvedor sin modificar ninguna línea de código, busca raíces de tu polinomio.
9. (**25 minutos**) Crea un nuevo método en la clase resolvedor tal que uno pueda especificar una función, y no sea necesario indicar el intervalo inicial para buscar la raíz. El propio método va a tratar de buscar un intervalo adecuado. Trata de idear una estrategia adecuada para encontrar ese intervalo.
10. (**20 minutos**) En una empresa todos los trabajadores tienen un sueldo base de 1000 €. Los jefes tienen un suplemento de 500 € por cada año que hayan sido jefe de la empresa, y los viajantes además del sueldo base cobran 300 € por viaje realizado. Crear una clase empleado de la cual deriven las clases jefe y viajante. Crear una plantilla de una empresa con dos jefes, cinco viajantes y 15 empleados, e imprimir por consola sus respectivos salarios. Para generar el número de viajes de los viajantes y la antigüedad de los jefes puedes generar números aleatorios entre 0 y 10. Emplea el polimorfismo de herencia.
11. (**30 minutos**) Crea la clase cuenta bancaria, que deberá tener como atributos un nombre de titular, una fecha de apertura, un número de cuenta y un saldo (puedes emplear un número real. La cuenta deberá tener un método para retirar dinero, otro para ingresar dinero y otro para transferir dinero a otra

cuenta. De una cuenta no se podrá retirar nunca dinero (ni transferir) si la cantidad de dinero a retirar o transferir es mayor que el saldo. Crea una cuenta a plazo fijo, en la cual cuando se retira dinero de algún modo antes deuna fecha de vencimiento (que será otro atributo de esta clase) además del dinero a retirar se penaliza con un 5% adicional. Crea además una cuenta Vip, que tendrá un atributo adicional que es el saldo negativo máximo que puede tener. En las cuentas Vip uno podrá tener saldo negativo siempre que no supere este valor. A continuación, construye un main que permita crearlos tres tipos de cuentas, y transferir dinero de unas a otras, ingresar dinero y retirar dinero. Almacena las cuentas en un array. Emplea polimorfismo de herencia.

12. (**20 minutos**) Crea una clase fecha que almacene el día, el mes y el año de una fecha. Proporciona funciones miembro para acceder a estos atributos (getDia(), getMes() y getAño() y para modificarlos (setDia(int dia), setMes(int mes) y setAño(int año)). Sobreescribe su método toString(). La clase debe asegurarse de que los valores introducidos para sus miembros, tanto a través de los constructores como de los métodos modificadores, se corresponden con una fecha válida (no es necesario tener en cuenta años bisiestos). Para ello lanzará una excepción en caso de que los datos no sean válidos. Crea la fecha 31/02/2015 y verifica que se lanza la excepción correspondiente. Verifica que esto también sucede al invocar el métodosetDia (35).
13. (**30 minutos**) Escribir un resolvedor que busque raíces de una función empleando el algoritmo de Newton-Raphson para encontrar una raíz de una función concreta. Este método parte de una estimación inicial de la raíz, x0, y va calculando aproximaciones sucesivas al valor de la misma utilizando la fórmula:

$$x_{i+1}=x_i-f(x_i)/f'(x_i)$$

Al ser un método iterativo, es necesario tener un criterio para que termine. Puede establecerse como criterio de terminación que la diferencia entre f(x) y 0 sea menor que un valor e pequeño. Además, por seguridad, se debe establecer un número máximo de iteraciones para que el algoritmo termine,

aunque no converja a una solución (de lo contrario, al ejecutarlo en elordenador éste podría quedarse bloqueado).