Ejercicios tema 1

Creen un proyecto nuevo para cada subtema, y en cada uno una clase ejecutable para probar todo. (Recuerden que nuestras clases Ejecutable sólo tienen `main()`).

COmo pueden ver, hay pocos ejercicios de POO y más de arreglos y control de flujo. Me interesa que mejoren su pensamiento algorítmico, POO pueden mejorar sobre la marcha en estas cinco semanas siempre que dominen los conceptos.

Control de flujo

Creen una clase ProblemasFlujo. En ella definan los siguientes métodos 1. `resolverCuadratica(double a, double b, double c)` que resuelva una ecuación cuadrática de la forma $ax^2 + bx + c = 0$. Recuerden pensar en los tres casos del discriminante.

2. Implementen un algoritmo para calcular el n-ésimo número de Fibonacci (`n` es un argumento de su función) *sin utilizar todo un arreglo.*
3. *Conjetura de Collatz.* Implementen una función `collatz(int x)` que haga lo siguiente:

- si `x` es par, lo divide entre 2
- si no, lo multiplica por tres y le suma 1 y continúe con esto hasta que el número sea 1.

Arreglos

Creen una clase ManejadorArreglos. En ella definan los siguientes métodos: (recuerden que todos deben ser estáticos)

1. `swap(int a[], int n, int i, int j)` recibe un arreglo `a` de tamaño `n` e intercambia los elementos en las posiciones `i` y `j`. Ojo con los casos donde podría no funcionar.
2. `reverse(int a[], int n)` invierte el arreglo. Pista: piensen en el caso de `n` par o impar y usen `swap()`.
3. `sort(int a[], int n)` ordena el arreglo. Pista. usen `swap()`.
4. `max(int a[], int n)` regresa el mayor elemento de `a`.
5. `maxIndex(a[], int n)` regresa *el índice* del mayor elemento de `a`;
6. `cuantosMayores(int a[], int n, int x)` cuenta cuántos elementos de `a` son mayores que la constante `x`.

7. `binarySearch(int a[], int n, int x)` busca la posición de `x` en `a` y devuelve *su índice*. Implementen también el caso que puede ser problemático. (Pista: es similar al 2)

8. `sonParalelos(int a[], int b[], n)` recibe dos arreglos `a` y `b` de tamaño `n` y devuelve True si son paralelos (es decir, el j-ésimo elemento de b es un múltiplo entero del j-ésimo elemento de a, para cada j). Por ejemplo `[1, 2, 3]` y `[2, 4, 6]` son paralelos; pero `[1, 2, 3]` y `[1, 4, 9]` no lo son.

9. Para hacer una matriz en Java, la sintaxis es similar a la de un arreglo, pero necesitan dos pares de corchetes, uno para cada dimensión. Por ejemplo,

```
int[][] a;
a = new int[5][10];
```

hace una matriz de 5×10. El ejercicio es que inicialicen una matriz `a` de 4×3 con los números que gusten y guarden su **transpuesta**. Es decir, guarden una matriz `b` de 4×3 tal que cada i-ésima fila de `b` sea la i-ésima columna de `a`.

Orientado a objetos

1. Este ejercicio repasa los conceptos básicos de POO y encapsulamiento: Un **número complejo** es de la forma $a + bi$ donde $i = \sqrt{-1}$. Escribe una clase Complejo que tenga como atributos `double parteReal` (la a) y `double parteImaginaria` (la b). Investiga cómo se definen la suma y la multplicación de números complejos e implementa ambas en una clase `Calculadora`. La clase `Calculadora` tiene métodos estáticos y no necesita constructor. En `Complejo`, escribe los dos constructores triviales.

2. El siguiente ejercicio es sobre herencia y polimorfismo.

- Crea una clase abstracta Mascota con atributos `String nombre` e `int edad`. Incluye un método abstracto `saludar()` y un método `getNombre()`.
- Crea dos subclases de `Mascota`: `Perro` y `Gato`. No olvides los constructores.
- En cada método completa el método `saludar()`. En `Gato` debes imprimir "miau" y en perro "roof".
- En el ejecutable, crea un arreglo de `Animal` de tamaño $n = 5$. Manualmente introduce algunos gatos y algunos perros. Escribe un ciclo dentro del main que ejecute `saludar()` y `getNombre()` en cada animal del arreglo.
- Observa que aunque el arreglo anterior era de `Animal`, introdujimos en él perros y gatos. ¿Por qué fue posible?