Universidad de Costa Rica
Escuela de Ciencias de la Computación e Informática
CI-0112 Programación 1, I-2019, 03-Julio-2019
Profesores: Jose Ciccio, Luis Gustavo Esquivel, Ricardo Gang, Jeisson Hidalgo

Examen Final

Observaciones

- Tiene tres horas para entregar su solución del examen.
- Debe dedicarse exclusivamente a la realización de la prueba y de manera individual.
- Es prohibido utilizar herramientas digitales para intercambio de documentos o comunicación.
- Guarde su celular. El uso del celular durante la prueba implica la anulación del examen.
- Puede utilizar cualquier material impreso (prácticas, tareas, libros, apuntes).
- Firme la hoja de asistencia.
- En la solución de los ejercicios, está prohibido utilizar contenedores de la biblioteca estándar.
- Las buenas prácticas y la funcionalidad del código serán evaluadas.

Descripción

1. [3%] Dibuje el árbol binario de búsqueda resultado de insertar los siguientes elementos: 10,6,3,5,2,3,2.

2. [3%] Sólo una de las siguientes declaraciones es verdadera, o sólo una es falsa. Escoja la que tiene un valor de verdad distinto a las demás.

a. Una declaración de un atributo siempre debe ir dentro del método main.
b. `System.out.println("hola")` es una declaración de una variable.
c. La declaración sin inicialización nunca puede tener un signo '=' con un valor a la derecha.
d. Una declaración con inicialización nunca puede tener un signo '=' con un valor a la derecha.

3. [10%] Considere el siguiente código:

```
public int x (int n [], int i) {
       int aux;
       if (i == 0)
           return n [i];
       else {
          aux = x (n, i - 1);
          if (n [i] > aux)
              return n [i];
          else
              return x(n, i - 1);
       }
}
```

Si envía por parámetro el arreglo: `n = [1,25,5,10]` e `i = 3`, qué valor retorna la función x?

4. [17%] Programe un método recursivo que convierta un número entero positivo en base diez a su equivalente en base 2 y lo retorne como un `String`.

Nota: Recuerde que para convertir un número entero decimal positivo a un número binario se debe dividir el número entre dos hasta que el cociente sea cero. Al final se toman los residuos de abajo hacia arriba.

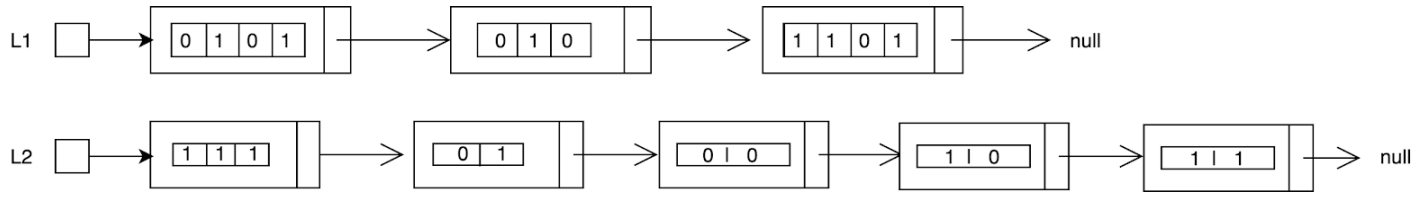
```
 50 |_2
-50   25 |_2
  0  -24   12 |_2
       1  -12   6 |_2
            0  -6   3 |_2
                0  -2   1 |_2
                    1  -0   0
                        1
```

$(50)_{10} = (110010)_2$

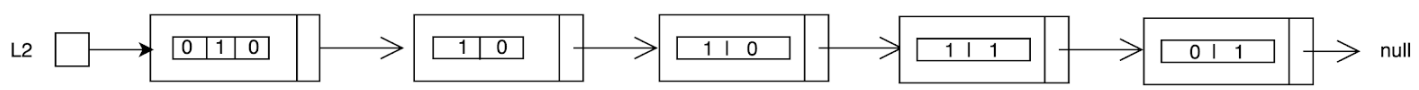
5. [17%] Usted posee dos listas, las cuales contienen nodos definidos de la siguiente forma:

```
public class Lista {
   Nodo primero = null;
   private class Nodo {
      private byte[] bits; //puede contener valores 0 o 1
      private Nodo siguiente;
      //métodos set y métodos de tipo get para cada atributo anterior
   }
}
```

El arreglo bits tiene un tamaño dado por su atributo `length`. La cantidad de elementos en estos arreglos no se puede modificar de ninguna forma. Cada lista tiene un número variable de nodos, los cuales a su vez pueden tener arreglos de bits de distintos tamaño. Sin embargo, la cantidad total de bits en una lista es igual a la cantidad total de bits en la otra, aunque su estructura sea distinta. Por ejemplo:



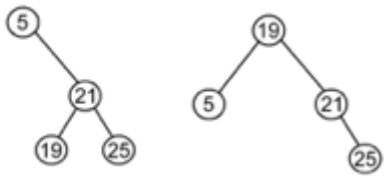
Programe un método en la clase `Lista` que copie los bits de una lista en otra. Recuerde que su método no puede modificar la estructura de ninguna de las dos listas. Sólo puede cambiar los valores en los arreglos de bits de la lista destino, para que sean iguales a los de la lista origen. Por ejemplo, al copiar la primera lista (`L1`) en la segunda lista (`L2`) se obtendría:



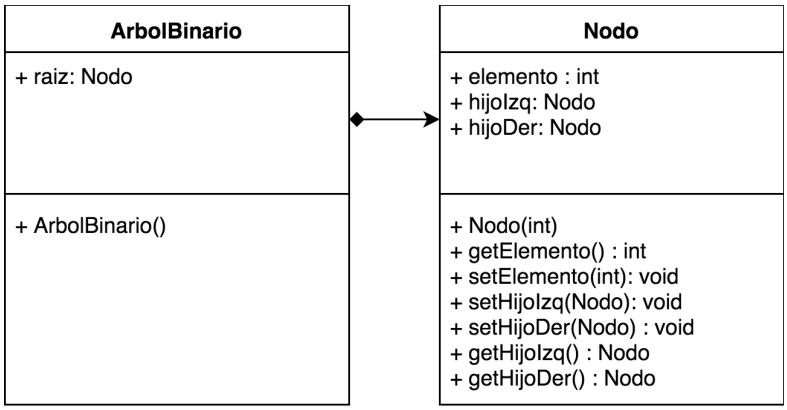
6. [25%] En una clase llamada `ArbolBinario` que implementa un árbol binario de búsqueda, programe el método:

```
boolean comparar(ArbolBinario otro)
```

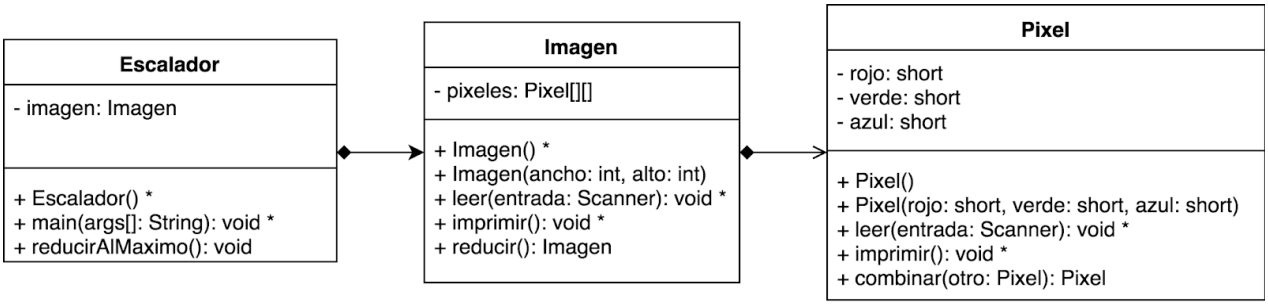
El método será capaz de determinar si **`otro`** tiene los mismos datos. Si el contenido de los árboles es igual, debe devolver verdadero aunque su balanceo sea diferente (ver la siguiente figura). El método `comparar` deberá retornar si ambas estructuras poseen exactamente los mismos datos.



Programe, además, todos los métodos que necesite dentro de la clase para que su solución sea funcional. Suponga que existe la clase Nodo con los siguientes atributos:



7. [25%] Implemente las clases, atributos, y métodos del diseño UML de la figura. Suponga que los métodos marcados con un asterisco (*) ya están implementados y que puede invocarlos en su solución.



Un `Pixel` en color real tiene tres componentes: rojo (R), verde (G), y azul (B). Cada componente puede contener un valor entre 0 y 255. Entre mayor este valor, más presencia del color se tiene. Por ejemplo: el negro es la ausencia de color (R=0 G=0 B=0), el turquesa se obtiene con la mitad de verde y azul (R=0 G=128 B=128), y el blanco con el máximo de todos los componentes (R=255 G=255 B=255).

Suponga que un píxel es capaz de cargar sus tres componentes desde un archivo usando un objeto `Scanner`, y de imprimirse en la salida estándar. Implemente los dos constructores `Pixel` según el diagrama. Además programe un método `combinar()` que permite a un píxel combinarse con otro píxel. Combinar dos píxeles de la forma `px1.combinar(px2)` produce un nuevo `Pixel` cuyos componentes son el promedio de los componentes respectivos de `px1` y `px2`. Por ejemplo, si se usa el símbolo • para representar la operación de combinar, entonces combinar turquesa (R=0 G=128 B=128) con morado (R=128 G=0 B=128) produce un azul oscuro:

```
(0 128 128) • (128 0 128) => (64 64 128)
```

Implemente una clase `Imagen` que tiene una matriz de píxeles en color real. Provea un constructor que permite crear una imagen de un tamaño dado con todos sus píxeles en negro. Implemente en la clase `Imagen` un método `reducir()` que al invocarse de la forma `img1.reducir()` produce otra `Imagen` con la mitad del alto y la mitad del ancho de `img1`. Los pixeles de la imagen producida es el resultado de combinar los pixeles en filas y columnas pares con los de las filas y columnas impares. Por ejemplo, si `pij` representa el píxel en la fila `i` y columna `j`, reducir una imagen de 3 filas y 4 columnas produciría otra imagen de 2 filas y 2 columnas cuyos valores serían:

```
p11 p12 p13 p14            (p11•p12•p21•p22) (p13•p14•p23•p24)
p21 p22 p23 p24     =>     (p31•p32)         (p33•p34)
p31 p32 p33 p34
```

En caso de que la imagen no se pueda escalar porque su alto o ancho es de un píxel, el método `reducir()` debe retornar una referencia `null`. Asuma que la clase Imagen tiene un método `leer()` que le permite cargarse desde un archivo a través de un objeto `Scanner`; y un método `imprimir()` que produce la salida como se ve en el caso de prueba.

Ejemplo de entrada:

```
3 5
(2 4 5) (10 20 30) (20 40 60) (80 90 100) (1 2 3)
(6 7 8) (40 50 60) (30 20 10) (50 60  40) (9 6 3)
(9 6 3) (30 10 20) (50 10 40) (60 20  80) (4 8 5)
```

Ejemplo de salida:

```
(2 4 5) (10 20 30) (20 40 60) (80 90 100) (1 2 3)
(6 7 8) (40 50 60) (30 20 10) (50 60  40) (9 6 3)
(9 6 3) (30 10 20) (50 10 40) (60 20  80) (4 8 5)

(14 20 25) (45 52 52) (5 4 3)
(19  8 11) (55 15 60) (4 8 5)

(33 23 36) (4 6 4)
```

Implemente una clase `Escalador`. Asuma que su método `main()` se instancia un `Escalador` y se invoca al método `reducirAlMaximo()`. Implemente el método `reducirAlMaximo()` para que lea la imagen de la entrada estándar, y repetitivamente imprima la imagen reducida hasta que ya no se pueda reducir más.