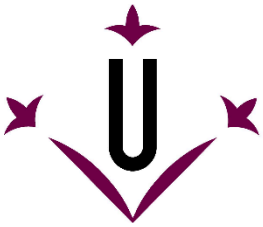
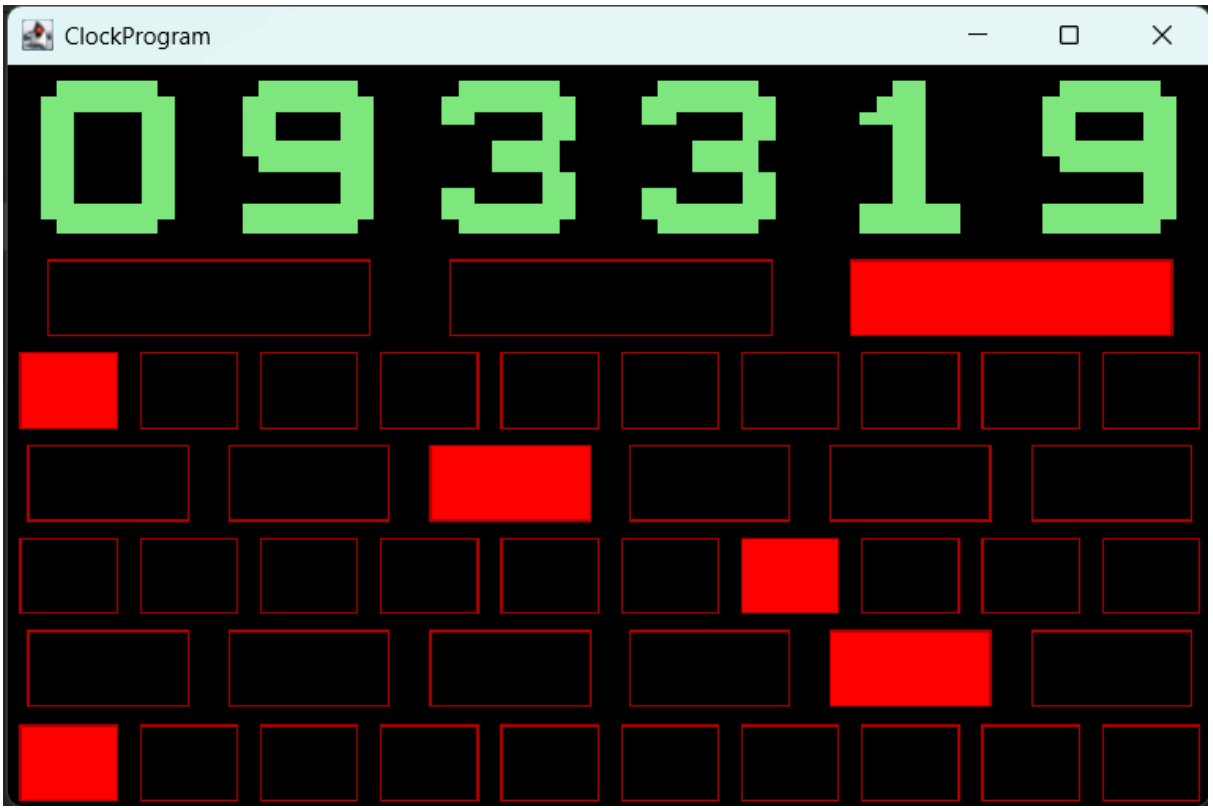


Universitat de Lleida
Escola Politècnica Superior

Segona pràctica
Programació II - Curs 2024-2025
Grau en Enginyeria Informàtica
Alumne: Biel Torruella Mayós

Rellotge digital amb leds:



Aquesta pràctica tracta de diferents classes, que s’ajuden entre elles per a poder arribar a l'objectiu final, crear un rellotge digital,
algunes de les classes ja venen completades, com serien:

- Step1 a Step7.
- ClockProgram
- Padding
- TimeTeller
- Led
- DecimalDigit
- Clock

També ens donen algunes classes a mig construir:

- Box

I finalment ens donen classes gairebé buides les quals hem de completar pel nostre compte.

- Bar
- Group
- DecimalNumber
- DecimalClock
- LedsClock

Cada Step ens serveix per comprovar part per part si estem programant correctament les classes, així que en aquest informe explicare com he realitzat les classes corresponents a cada Step.

Step1

(Box)

Atributs:

x i y, corresponent a la posició i width i heigth al tamany

```
private final double x;
private final double y;
private final double width;
private final double height;
```

Constructor

```
public Box (double x, double y, double width, double height) {
   this.x = x;
   this.y = y;
   this.width = width;
   this.height = height;
}
```

Mètodes d'accés

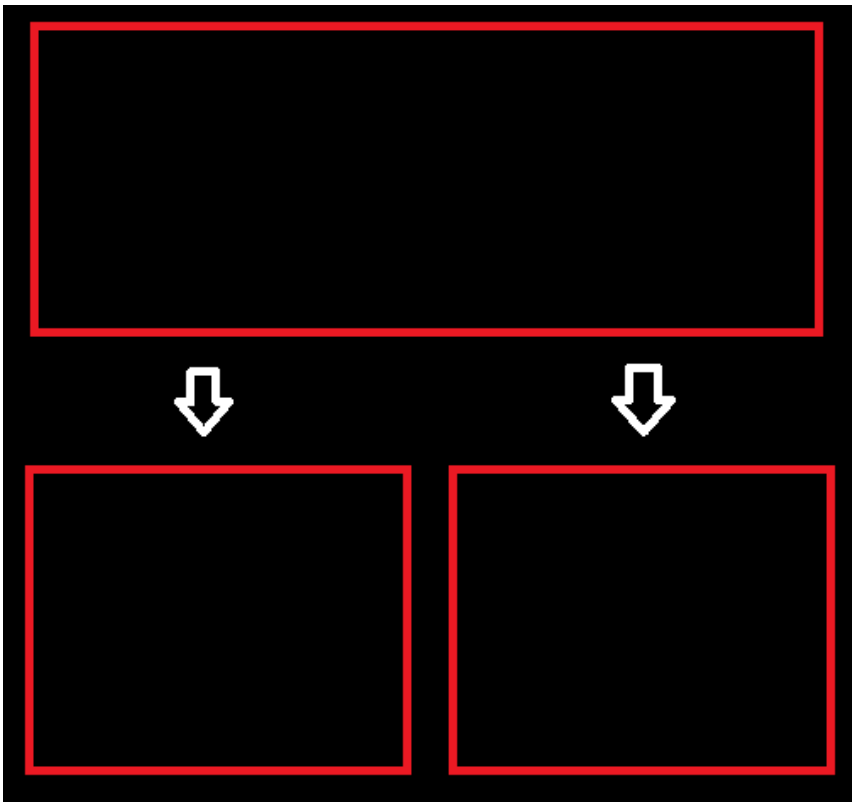
Aquí he creat els diferents getters per a poder accedir a aquesta informació desde una altra classe.

```
public double getX() {
   return x;
}

public double getY() {
   return y;
}

public double getWidth() {
   return width;
}

public double getHeight() {
   return height;
}
```

Mètodes fets per mi:

- **distributeHorizontally**: Aquesta funció s'encarrega de distribuir la box de forma horitzontal en diferents box de la mateixa mida.

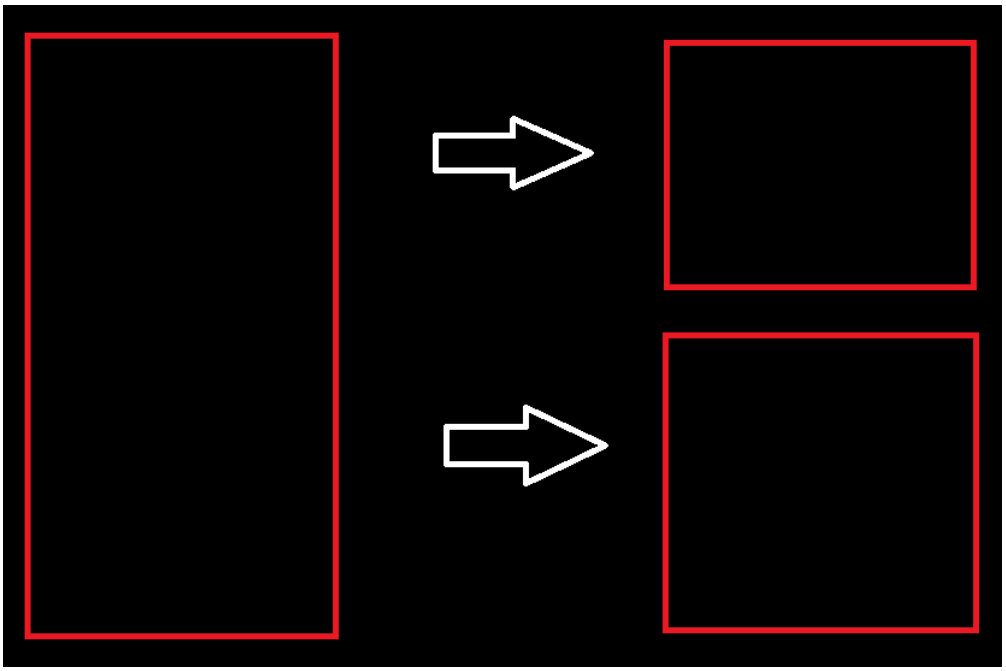
```
public Box[] distributeHorizontally (int numColumns) {
   //creo un double, equivalent a l'amplada de la caixa dividida
   double len=getWidth()/numColumns;
   Box[] b = new Box[numColumns];
   for (int i=0; i<numColumns; i++) {
       b[i] = new Box (this.x+i*len, this.y, len, this.height);
   }
   return b;
}
```



-

- **distributeVertically**: Aquesta funció s'encarrega de distribuir la box de forma vertical en diferents box de la mateixa mida.

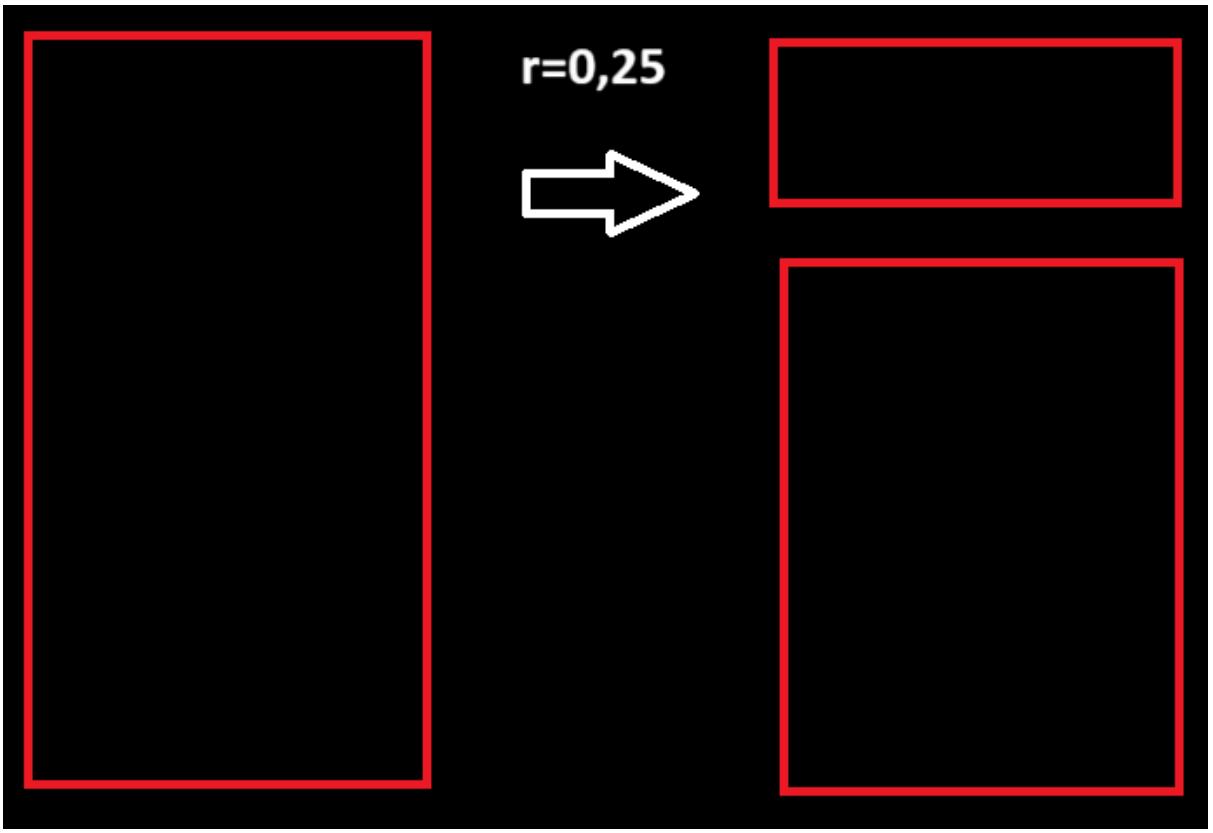
```
public Box[] distributeVertically (int numRows) {
   //creo un double, equivalent a l'alçada de la caixa dividida
   double len = getHeight()/numRows;
   Box[] b = new Box[numRows];
   for (int i=0; i<numRows; i++) {
       b[i] = new Box (this.x, this.y+i*len, this.width, len);
   }
   return b;
}
```



-

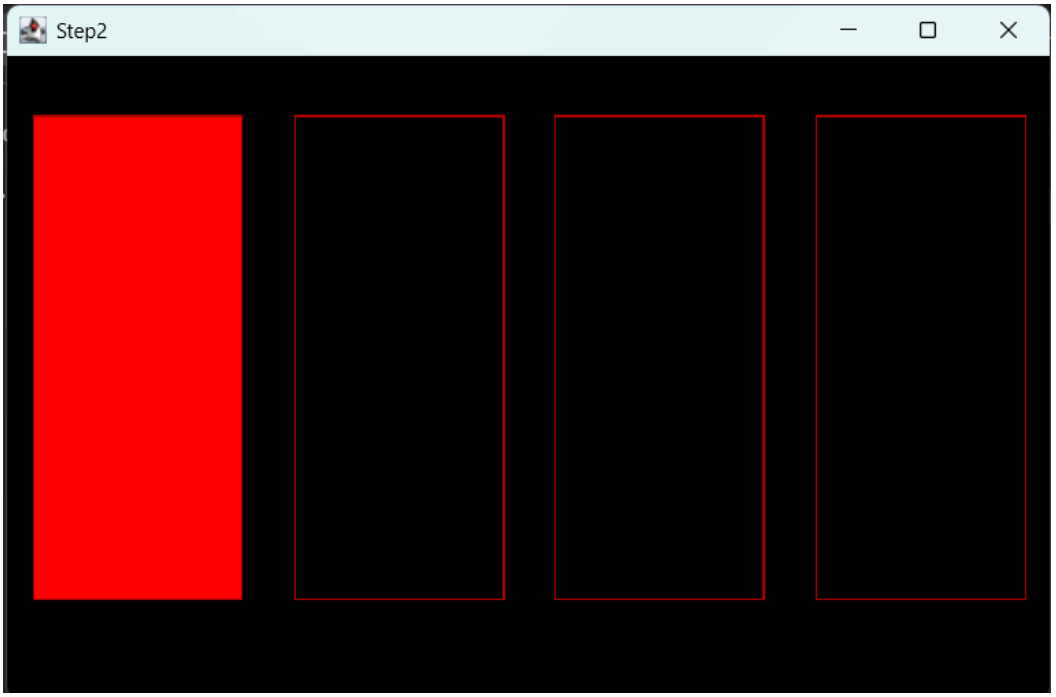
- **splitTopBottom**: Aquesta funció s'encarrega de dividir el box en dos, i l'altura en la que es fa aquesta divisió depèn del ratio.

```
public Box[] splitTopBottom (double ratio) {
   Box caixaSuperior = new Box(this.x, this.y,
this.width,this.height*ratio);
   Box caixaInferior = new Box(this.x,
this.y+caixaSuperior.height, this.width,this.height*(1-ratio));
   return new Box[]{ caixaSuperior, caixaInferior };
}
```



Step2

(Bar)



Atributs:
Dintre del bar hi ha un grup de leds.

```
private Led[] led;
```

Constructor

Guradem aquest grup de leds que anomenem bar en ordre de dreta a esquerra.

```
public Bar (int size, Box barBox) {
   Box[] boxes = barBox.distributeHorizontally(size);
   this.led = new Led[size];
   for (int i = 0; i < size; i++) {
       //Ho he guardat al [size-i-1] per a que vagi en direcció de dreta
a esquerra
       this.led[i] = new Led(boxes[size-i-1]);
   }
}
```

Metodes

- **addToGcanvas:** Aquesta funció s'encarrega de afegir al Canvas els leds de la barra

```
public void addToGCanvas (GCanvas gCanvas) {
   for (int i = 0; i < led.length; i++) {
       led[i].addToGCanvas(gCanvas);
   }
}
```

- **render:** Aquesta funció engega i apaga el led

```
public void render (int value) {
   for (int i = 0; i < led.length; i++) {
       //esta apagat i en cas de que i == value, s'engega
       led[i].off();
       if (i == value) {
           led[i].on();
       }
   }
}
```

Step3

(Group)

Atributs

Un group, és un grup de Bars, a més un Bar és un grup de leds

```
private Bar[] stick;
```

Constructor

Guardem en un array de Box anomenat boxes la caixa original dividida en subcaixes, distribuides verticalment.
Guardem els groups en aquestes subcaixes.

```
public Group (int[] sizes, Box groupBox) {
   Box[] boxes = groupBox.distributeVertically(sizes.length);
   stick = new Bar[sizes.length];
   for (int i = 0; i < sizes.length; i++) {
      stick[i] = new Bar(sizes[i], boxes[i]);
   }
}
```

Metodes

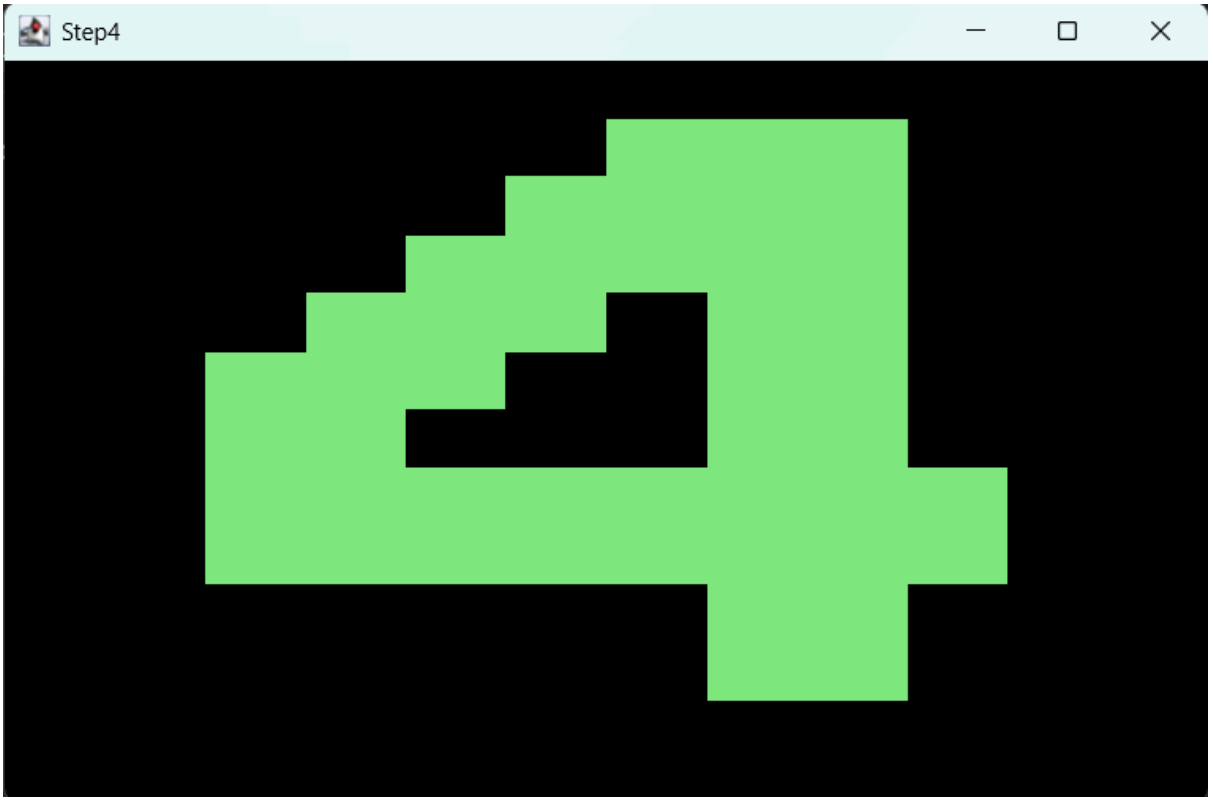
- **addToGcanvas:** Aquesta funció s'encarrega de afegir al Canvas els barrs

```
public void addToGCanvas (GCanvas gCanvas) {
   for (int i = 0; i < stick.length; i++) {
      stick[i].addToGCanvas(gCanvas);
   }
}
```

- **render:** Aquesta funció engega els leds de dintre dels diferent bars que hi ha a groups

```
public void render (int[] values) {
   for (int i = 0; i < stick.length; i++) {
      stick[i].render(values[values.length-i-1]);
   }
}
```

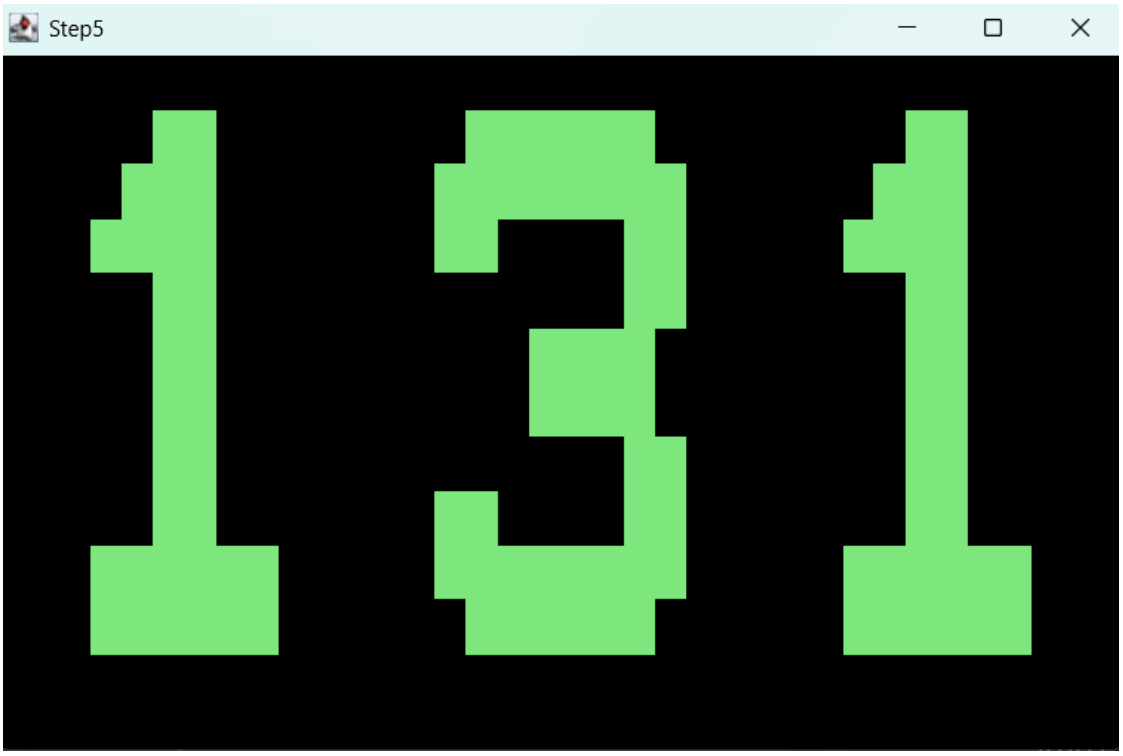
Step4



Aquesta part ja està completa, ja que utilitza la classe Box, que la hem completat anteriorment, i el DecimalDigit, que ja ens venía donat

Step5

(DecimalNumber)



Atributs

un DecimalNumber es un grup de DecimalDigits, que es guarda a un array.

```
private DecimalDigit[] digits;
```

Constructor

Guardem en un array de Box anomenat boxes la caixa original dividida en subcaixes, distribuides horitzontalment.
Guardem el array de DigitNumber en un DecimalNumber, guardant-ho en les subcaixes.

```
public DecimalNumber (int numDigits, Box dnBox) {
   Box[] boxes= dnBox.distributeHorizontally(numDigits);
   this.digits = new DecimalDigit[numDigits];
   for (int i = 0; i < numDigits; i++) {
      this.digits[i] = new DecimalDigit(boxes[i]);
   }
}
```

Metodes

- **addToGcanvas:** Aquesta funció s'encarrega de afegir al Canvas el digit decimal

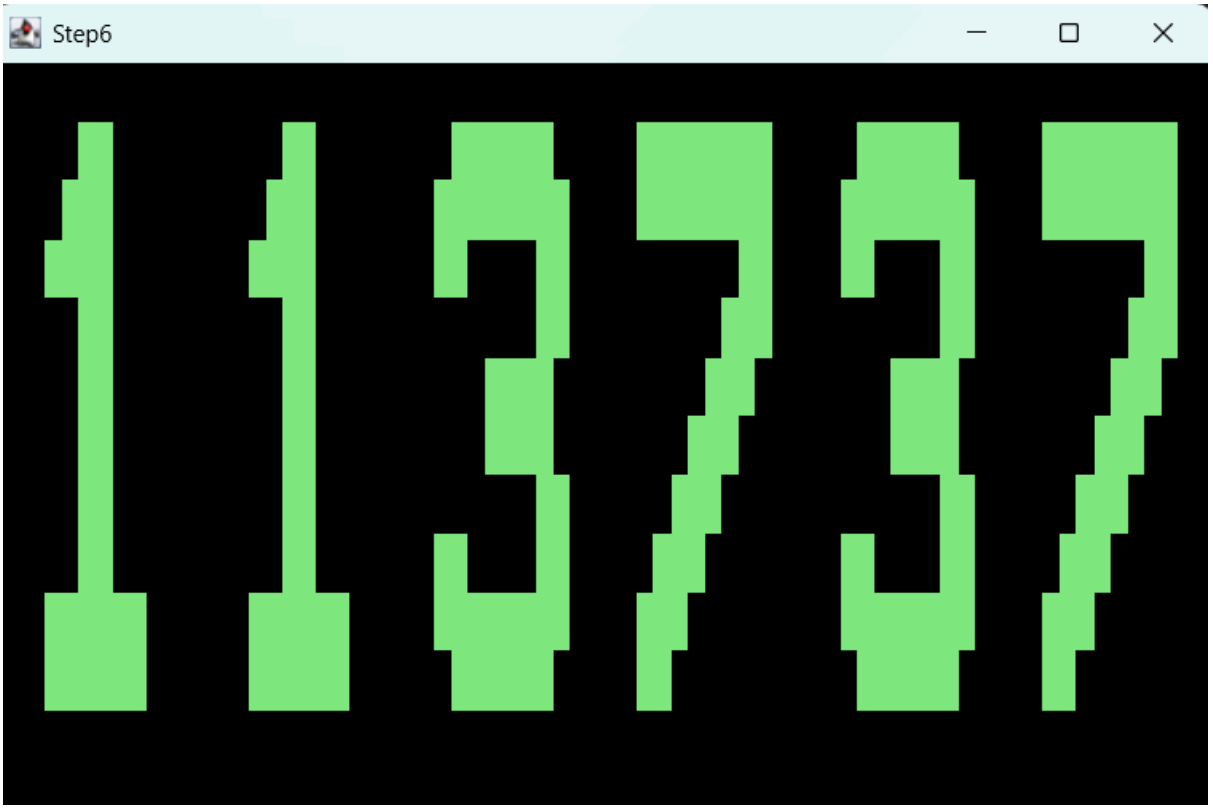
```
public void addToGCanvas (GCanvas gCanvas) {
   for (int i = 0; i < this.digits.length; i++) {
      this.digits[i].addToGCanvas(gCanvas);
   }
}
```

- **render:** Aquesta funció dibuixa al canvas cada digit decimal del number, per a fer aixo, primer converteix el number en forma int en un array, ex: 235 → {2, 3, 5}, ja que el render de digit decimal tracta amb int, i d'aquesta manera pot trebatllar amb cada digit un per un.

```
public void render (int number) {
   //Converteixo el int en array, per a que el render pugui tractar amb
cada dada unitat per unitat.
   // 264 = {2, 6, 4}
   int[] numbers = new int[this.digits.length];
   int carry = 0;
   numbers[digits.length-1] = number;
   for (int i = digits.length - 1; i >= 0; i--) {
      numbers[i]=numbers[i]+carry;
      carry=0;
      while (numbers[i] >= 10){
         carry++;
         numbers[i]-=10;
      }
   }
   for (int i = 0; i < numbers.length; i++) {
      this.digits[i].render(numbers[i]);
   }
}
```

Step6

(DecimalClock)



Atributs i Constants

NUM_DECIMALS és una constant que ens dona l'enunciat
NUM_DIGITS el suposem, ja que en un rellotge com a maxim s'utilitzen 2 digits per part

```
private static int NUM_DECIMALS = 3;
private static int NUM_DIGITS = 2;
DecimalNumber[] decnum= new DecimalNumber[NUM_DECIMALS];
```

Constructor

Guardem en un array de Box anomenat boxes la caixa original dividida en subcaixes, distribuides horitzontalment.
Guardem el array de DecimalNumber en un DecimalClock, guardant-ho en les subcaixes.

```
public DecimalClock (Box dcBox) {
   Box[] boxes = dcBox.distributeHorizontally(NUM_DECIMALS);
   for (int i = 0; i < NUM_DECIMALS; i++) {
      this.decnum[i] = new DecimalNumber(NUM_DIGITS, boxes[i]);
   }
}
```

Metodes

- **addToGCanvas:** Aquest metode dibuixa al gCanvas els diferents numeros decimals.

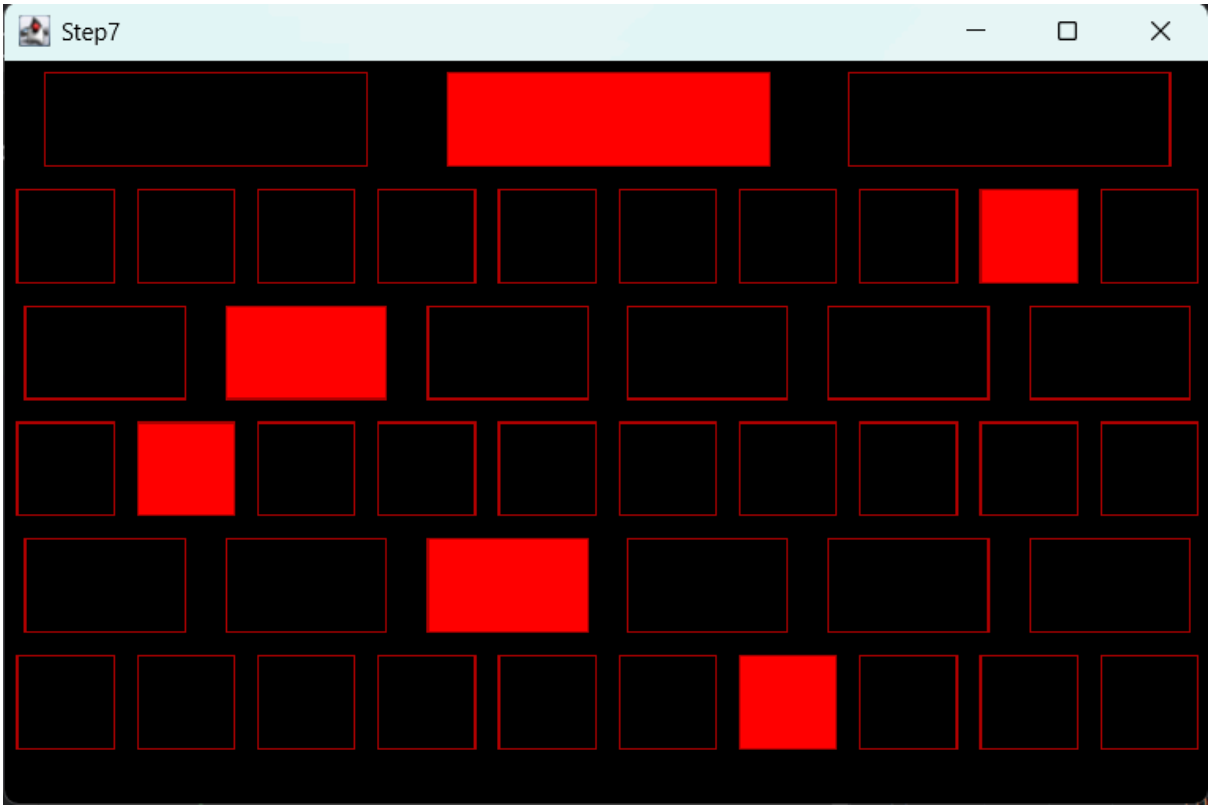
```
public void addToGCanvas (GCanvas gCanvas) {
   for (int i = 0; i < NUM_DECIMALS; i++) {
      this.decnum[i].addToGCanvas(gCanvas);
   }
}
```

- **render:** Aquest mètode encén els leds corresponents a les posicions de les dades: hores minuts i segons.

```
public void render (int hours, int minutes, int seconds) {
   //guardo al array les dades en el ordre que vull que es guardi al
decnum.
   int[] numbers = { hours, minutes, seconds };
   for (int i = 0; i < NUM_DECIMALS; i++) {
      this.decnum[i].render(numbers[i]);
   }
}
```

Step7

(LedsClock)



Atributs i Constants

```
//el valor de NUM_GROUPS m'el donen al enunciat
private static int NUM_GROUPS = 3;

//el valor de LED_SIZE el dono per suposat, ja que tractant de un
rellotje, seran dades de 2 unitats, ex: 09:19 (2 i 2).
private static int LED_SIZE = 2;

//el hours_size, es el maxim de valors que pot tenir un rellotje al
donar la hora, doncs el max seria 23 hores.
//3 = {0, 1, 2} || 10 = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
private static int[] hours_size = {3, 10};

//el minutes_size es igual a el seconds_size, per tant ho faig tot
directament amb el minutes_size.
//el qual tracta de el maxim de num de minuts que seria 59 minuts.
//6 = {0, 1, 2, 3, 4, 5} || 10 = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
private static int[] minutes_size = {6, 10};

Group[] ledGroups = new Group[NUM_GROUPS];
```

Constructor

Guardem en un array de Box anomenat boxes la caixa original dividida en subcaixes, distribuides verticalment.
Guardem el array de Groups en un LedsClock, guardant-ho en les subcaixes.

```
public LedsClock (Box lcBox) {
   Box[] boxes = lcBox.distributeVertically(NUM_GROUPS);
   for (int i = 0; i < NUM_GROUPS; i++) {
      //en cas de que i==0, vol dir que esta tractant de la hora, per
tant es treball amb el tamany hours_size
      if (i == 0){
         this.ledGroups[i] = new Group(hours_size, boxes[i]);
      } else {
         //en altre cas, es tracta amb el tamany de munutes_size
         this.ledGroups[i] = new Group(minutes_size, boxes[i]);
      }
   }
}
```

Metodes

- **addToGCanvas:** Aquest metode dibuixa al gCanvas els diferents grups de numeros decimals.

```
public void addToGCanvas (GCanvas gCanvas) {
   for (int i = 0; i < NUM_GROUPS; i++) {
      this.ledGroups[i].addToGCanvas(gCanvas);
   }
}
```

- **SplitDigit:**

```
//He creat una funció que converteixi un int a un int[], ex: 23 --> {3,
2}
public static int[] splitDigits(int number, int size) {
   int[] digits = new int[size];
   String numStr = String.format("%0" + size + "d", number); // afegeix
zeros davant si cal
   for (int i = 0; i < size; i++) {
      digits[size - 1 - i] = numStr.charAt(i) - '0';
   }
   return digits;
}
```

- **render:** Aquest mètode encén els leds corresponents a les posicions de les dades: hores minuts i segons, ocupant 2 caixes per cada dada.

```
public void render (int hours, int minutes, int seconds) {
   //guardo les dades de hora, minuts i segons en un array per poder
aplicar-ho en un for
   int[] d = {hours, minutes, seconds};
   for (int i = 0; i < NUM_GROUPS; i++) {
      //aplico el splitdigits a les dades dintre del array per a
convertirles en array i que es pugui aplicar al render
      this.ledGroups[i].render(splitDigits(d[i], LED_SIZE));
   }
}
```