```
// Active PDE program version.
// 8-bit Binary to Unsigned integer number
// Tilrettet til at være kvadrater i stedet for cirkler
// Tilrettelser lavet af Lucas og Kristoffer 2d2

Bit[] bits = new Bit[8];    // array with room for 8 on/off Bit instances
int decimal = 0;
int binary = 0;
PFont font;

void setup() {
  size(600, 300);
  noStroke();
  font = createFont("Arial", 48, true);    // Windows 10 have a wrong font path
  for (int i = 0; i < bits.length; i++) {
    bits[i] = new Bit(i);
  }  // for
}  // setup

void draw() {
  background(0);
  for (int i = 0; i < bits.length; i++) {
    bits[i].display();
    fill(255);
    int bitValue = 1 << (bits.length - i - 1);    // very fast calculation of 2^i
    text(bitValue, width/9 * bits[i].position - 10, 50);
  }  // for
  fill(255);
  textFont(font, 48);
  textAlign(RIGHT);
  text(nf(binary, 8), width/9*8, 180);
  text(decimal, width/9*8, 230);
  textAlign(LEFT);
  text("Binært:", width/9, 180);
  text("Decimalt:", width/9, 230);
  textFont(font, 18);
  fill(0, 255, 255);  // Cyan text
  text("Klik en bit for at tænde (og addere værdien 2^n) eller sluk for en bit.",
width/22, 25);
}  // draw

void keyReleased() {
  decimal = 0;
  binary = 0;
  for (int i = 0; i < bits.length; i++) {
    bits[i].updateKey();
    decimal += bits[i].value;
    binary += bits[i].digit;
  }  // for
}  // keyReleased

void mouseReleased() {
  decimal = 0;
  binary = 0;
  for (int i = 0; i < bits.length; i++) {
    bits[i].updateMouse();
    decimal += bits[i].value;
    binary += bits[i].digit;
  }  // for
}  // mouseReleased
```

```
60
61 class Bit {             // Bit object class
62   int position;
63   color colour = (55);  // Grey
64   int value = 0;
65   int digit = 0;
66
67   Bit(int pos) {
68     position = pos + 1;
69   }
70
71   void display() {
72     fill(colour);
73     rectMode(CENTER);                     // Sætter rectMode til "center" dette
   betyder at rektangler tegnet med "rect()" defineres ud fra deres centrum, dette
74     rect(width/9 * position, 80, 50, 50); // Tegner 8 rektangler med variabel  x
   position og en kostant y position på 80, alle disse rektangler har sidelængder på 80
75   }
76
77   void updateKey() {
78     if (key == position + 48) {
79       switch(colour) {
80         case (55):
81           colour = (255);
82           value = int (pow(2, 8 - position));
83           digit = int (pow(10, 8 - position));
84           break;
85         case (255):
86           colour = (55);
87           value = 0;
88           digit = 0;
89           break;
90       }  // switch
91     }  // if
92   }  // updateKey
93
94   void updateMouse() {
95     if (onSquare(width/9 * position, 80, 50)) {
96       switch(colour) {
97         case (55):
98           colour = (255);
99           value = int (pow(2, 8 - position));    // slow calculations
100          digit = int (pow(10, 8 - position));
101          break;
102        case (255):
103          colour = (55);
104          value = 0;
105          digit = 0;
106          break;
107      }  // switch
108    }  // onSquare
109  }  // updateMouse
110 }  // class
111
112 boolean onSquare(int x, int y, int squareWidth) {
113   float distX = abs(x - mouseX);                      // Tvinger musens x distancen
   til rektanglerne om til en positiv værdi
114   float distY = abs(y - mouseY);                      // Tvinger musens y distancen
   til rektanglerne om til en positiv værdi
```

```
115   float squareRadius = squareWidth / 2;               // Det halve af sidelængden
udregenes
116   if(distX <= squareRadius && distY <= squareRadius){ // checker ud fra samme
midtpunkter som rektanglerne har afstanden til musen, hvis begge disse afstande er
under eller lig halvdelen af rektanglernes bredde returneres "true" ellers retuneres
"false"
117     return true;
118   } else {
119     return false;
120   }  // if
121 }  // end onSquare
122
123 // end
124
```