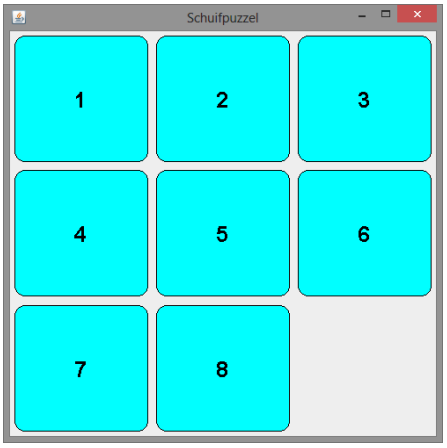
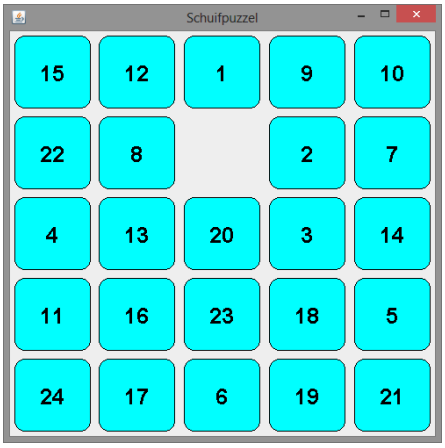
# Java Oefeningen – Reeks 5

Voor deze reeks gaan we een schuifpuzzel maken met een grafische interface. De gebruiker zal de blokjes kunnen verschuiven met zowel de muis als de pijltjestoetsen. We gaan ook een "slim" algoritme voorzien die de puzzel voor ons kan oplossen. Maar voor deze reeks een nieuw project "**Reeks5**" aan.




*Figuur 1*




*Figuur 2*

## Oef 1

We gaan het grid van blokjes voorstellen door een klasse **PuzzelNode**. Het spel wordt voorgesteld door een eendimensionale array van **int**s **data**. Verder heeft het grid een dimensie **size** (bovenstaande figuren hebben bv. dimensies van respectievelijk 3 en 5) en een attribuut **zeropos** die de positie van het lege vakje in het grid bijhoudt.

De array **data** stelt blokjes voor met ints groter dan 0, en het lege vakje met 0 zelf. Bv: het veld in de tweede figuur wordt voorgesteld als: $[15,12,1,9,10,22,8,0,2,7,\ldots]$. Maak een constructor die een **size** meekrijgt, en het PuzzelNode in zijn beginstaat initialiseert.

Integreer de klassen **Schuifpuzzel** (een **JFrame**) en **Puzzelpaneel** (een **JPanel**) in je code. Het **Puzzelpaneel** heeft als attributen een **final int gridsize** (zet deze voorlopig op 3) en een **PuzzelNode**.

Teken de blokjes in de methode **paintComponent()**. Gebruik hiervoor de methodes "fillRoundRect" en "drawRoundRect" van het object **Graphics**. Gebruik **centerString()** om de getallen gecentreerd te tekenen in hun respectievelijke vakjes. Probeer een grid te verkrijgen zoals aangegeven op de bovenstaande figuren.

# Oef 2

Maak een **enum Direction** die de vier richtingen voorstelt (***RIGHT, UP, LEFT, DOWN***). Maak ook een **static** attribuut die de array van alle **Direction**s bijhoudt via **Direction.values()**.

We gaan in **PuzzelNode** de mogelijkheid voorzien om met de muis het spel te besturen. Hiervoor hebben we de volgende methodes nodig:

- **boolean can_move(Direction dir):** geeft terug als returnwaarde of het mogelijk is om een blokje in de puzzel in de gegeven richting te verplaatsen. Maak gebruik van **zeropos**.
- **void move(Direction dir):** beweegt een blokje in de gegeven richting. Ga ervan uit dat al gecontroleerd werd indien dit mogelijk was (bv. met **can_move**). Vergeet niet om **zeropos** te updaten.
- **void click_grid(int pos):** beweeg het grid zodanig wanneer er op positie **pos** geklikt werd. Gebruik hiervoor de twee voorgaande methodes.

Gebruik de **MouseListener** implementatie in **Puzzelpaneel** (met het keyword **implements**) om te achterhalen op welk blokje er geklikt werd. Pas het veld aan met **PuzzelNode.click_grid()**, en herteken het spel. Je zou nu in staat moeten zijn om de puzzel te verschuiven.

# Oef 3

In dit deel gaan we besturing met toetsen implementeren. We gaan de **KeyListener** toevoegen aan **Puzzelpaneel**: gebruik hiervoor **implements** net zoals bij de **MouseListener**.

Om toesten te gebruiken, moet het paneel in focus zijn. Hiervoor wordt in **Puzzelpaneel** **"requestFocusInWindow()"** en **"setFocusable(true)"** opgeroepen. De KeyListener wordt ook toegevoegd aan het Puzzelpaneel met **addKeyListener(this).**

We gaan de functionaliteit voorzien in **keyPressed**. Zorg dat de speler het veld kan aanpassen met de pijltjestoetsen: roep de methodes van **PuzzelNode.can_move()** en **PuzzelNode.move()** afhankelijk van de ingedrukte toets.

We willen ook het veld automatisch kunnen laten reshuffelen. Maak hiervoor in **PuzzelNode:**

- **List<Direction> possibleMoves():** geeft een lijst terug met alle mogelijke richtingen in welke bewogen kan worden in de huidige configuratie.
- **void shuffle():** voert $20 \cdot size^3$ random moves uit m.b.v. **possibleMoves()**.

Zorg dat wanneer de speler op de toets **"R"** drukt, het veld "reshuffeled" wordt.