```
Peration == "MIRROR X":

Lror_mod.use_x = True

Lror_mod.use_y = False

Lror_mod.use_z = False

Operation == "MIRROR_Y"

Irror_mod.use_x = False

Operation == "MIRROR_Y"

Irror_mod.use_x = False

Irror_mod.use_z = False

Operation == "MIRROR_Z"

Irror_mod.use_x = False

Operation == "MIRROR_Z"

Irror_mod.use_x = False

Irror_mod.use_y = False

Irror_mod.use_y = True
```

Sement de la constant práce

irror_ob.select = 0
bpy.context.selected_ob
ata.objects[one.name].sel

int("please select exaction

OPERATOR CLASSEV astimil Pálfi

is not

xypes.Operator):
 X mirror to the select
 ject.mirror_mirror_x"
 ror X"

Zadání úlohy

- Zapsat program, který vypíše šachovnici
- Tvar obdélníkové matice při zadáním velikosti řádku a sloupce
- Skládá se z 1 a 0 (začíná od nuly)
- Zadá se velikost bloků
- Výsledkem bude výsledná šachovnice

Řešení úlohy

Řešení úlohy

- Načtení uživatelského vstupu
 - Hlídáno, jestli nezadal špatný vstup
- Matice je rozdělena na dvě části/metody

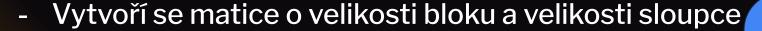
Důležité metody

Metoda pro vytvoření části matice

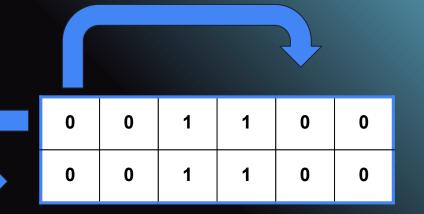
Metoda vytvoření matice k ní inverzní

Metoda pro pospojování první a druhé matice

První část

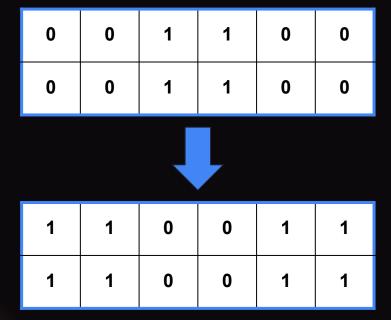


- Iteruje řádky, ve sloupci iteruje posunutím
 - Nejdříve zapisuje nuly do matice o velikosti bloku,
 POTÉ jedničky o stejné velikosti
 - Provede iteraci sloupce (dokud to umožnuje) a posune se o místo po poslední jedničce
 - Posune se na další řádek, dělá to tak dlouho, dokud ČÁST matice není hotová



Druhá část

- Vezme výsledek předchozí matice a vytvoří zní matici k ní inverzní



Finální část

- Vytvoření matice o velikosti našeho zadaného řádku a sloupce
- Poté se do šachovnice uloží opakovaně první a druhá část matice
- Pokud začne velikost šachovnice přetékat, vratí šachovnici

1	1	0	0	1	1
1	1	0	0	1	1
0	0	1	1	0	0
0	0	1	1	0	0
1	1	0	0	1	1
1	1	0	0	1	1

Nejzajímavější část kódu

Ukázka kódu

```
private static int[][] generateSecondPart(int[][] matrixPart1) {
    int[][] matrixPart2 = new int[matrixPart1.length][matrixPart1[0].length];
    for (int i = 0; i < matrixPart2.length; i++) {
        for (int j = 0; j < matrixPart2[i].length; j++) {
            matrixPart2[i][j] = (matrixPart1[i][j] == 0) ? 1 : 0;
        }
    }
    return matrixPart2;
}</pre>
```

Testy

Bězná hodnota

- Při vložení správných dat

```
Zadejte počet řádků.

Zadejte počet sloupců.

Zadejte velikost bloku

Výsledná matice

0 0 1 1 0 0

0 0 1 1 0 0

1 1 0 0 1 1

1 1 0 0 1 1

0 0 1 1 0 0

0 0 1 1 0 0
```

```
Zadejte počet řádků.

Zadejte počet sloupců.

Zadejte velikost bloku

Výsledná matice
0 0 0 1 1 1 0
0 0 0 1 1 1 0
1 1 1 0 0 0 1
1 1 1 0 0 0 1
1 1 1 0 0 0 1
0 0 0 1 1 1 0
```

Limitní stav

- Vstup, jenž jde přes omezení programu

Nevalidní vstup

- Ošetření vstupu, které nedávají pro samotný program smysl

```
Peration == "MIRROR X":

Irror_mod.use_x = True

Irror_mod.use_y = False

Irror_mod.use_z = False

Operation == "MIRROR Y"

Irror_mod.use_x = False

Irror_mod.use_x = False

Irror_mod.use_y = True

Irror_mod.use_z = False

Operation == "MIRROR_Z"

Irror_mod.use_x = False

Irror_mod.use_y = False

Irror_mod.use_y = True

Irror_mod.use_y = False

Irror_mod.use_z = True

Irror_mod.use_z = True
```

election at the end -add o select = 1 er ob elect = 1 ttxt cene objects.act "Selecter" + str(modification irror_ob.select = 0 bpy.context.selected_ob bata.objects[one.name].selected_ob ata.objects select exact wint("please select exact)

OPERATOR CLASSES ----

x mirror to the select
ject.mirror_mirror_x"
ror X"

te not

of selection at the end and of selection at the end at the end and of selection at the end at the end at the end and of selection at the end at