SPĒLE 16 – KĀ SAJAUKT?

25.nodarbība

MĒRĶIS – VISU SAJAUKT!

VARIANTS AR GADĪJUMA SKAITĻIEM

- Jāiet cauri visām spēles laukuma rūtiņām un katrā jāģenerē gadījuma skaitlis diapazonā no 0 līdz 15.
- To mēs protam ☺

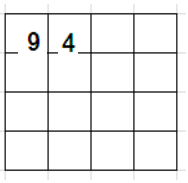
9	4	1	2
12	6	0	15
14	10	9	7
3	5	8	6

```
int[][] game=new int[4][4];
Random generator=new Random();
for(int row=0;row<4;row++){
    for(int column=0;column<4;column++){
        game[row][column]=generator.nextInt(16);
    }
}
```

- Taču var gadīties, ka 9 tiek uzģenerēts divreiz, bet 13 ne reizi.
- Tas mums neder ☹

KATRS SKAITLIS TIEŠI VIENREIZ

Vajadzētu kādu kontroles mehānismu..



- «Uzkrīt» 9. Atķeksējam un ieliekam laukumā!
- «Uzkrīt» 4. Atķeksējam un ieliekam laukumā!
- «Uzkrīt» 9. Tas jau ir atķeksēts. Ģenerējam citu gadījuma skaitli.
- Tik ilgi, kamēr katrā lauciņā cits skaitlis.

KĀ «ATĶEKSĒŠANU» REALIZĒT PROGRAMMĀ?

- Vispirms vajag kaut ko, kam var būt 2 stāvokļi – atķeksēts vai nē. Tam labi noder masīvs no loģiskajiem jeb boolean mainīgajiem.

```
boolean[] yesNo=new boolean[16];
```

- Masīvu vispirms piepilda ar vērtībām true – visi gadījuma skaitļi vēl ir pieejami.

```
for (int i=0;i<16;i++){
    yesNo[i]=true;
}
```

- Tad ģenerējam gadījuma skaitli x. Ja masīva x-tajā rūtiņā stāv true, tad šis skaitlis x vēl ir brīvs un izmantojams, bet turpmāk tāds vairs nebūs – nomainām true uz false.
- Lai procesu kādreiz varētu arī beigt, pie i pieskaita 1 katru reizi, kad izdevies atrast brīvu skaitli.
- Kad atrasti 16 brīvi, atšķirīgi skaitļi, darbu beidzam.

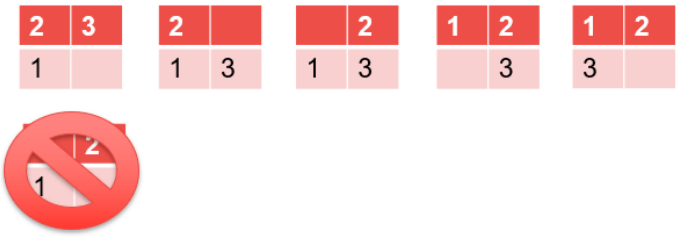
```
Random generator=new Random();
int i=0;
do{
    int x=generator.nextInt(16);
    if(yesNo[x]){
        yesNo[x]=false;
        i++;
    }
}while(i<16);
```

KATRS SKAITLIS TIEŠI VIENREIZ - KODS

```
int game[][]=new int[4][4];
boolean mem[] =new boolean[16];
for(int i=0;i<16;i++) mem[i]=true;
int x;
Random generator =new Random();
for(int row=0;row<4;row++){
    for(int column=0;column<4;column++){
        do{
        x=generator.nextInt(16);
        }while (!mem[x]);
    game[row][column]=x;
    mem[x]=false;
    }
}
```

KATRS SKAITLIS TIEŠI VIENREIZ - PROBLĒMAS

- Diezgan lēni, taču ar 16 skaitļiem to nejutīs.
- 0 jeb tukšais lauciņš var gadīties jebkurā vietā, taču pēc noteikumiem tam jābūt labajā apakšējā stūrī.
- Trakākais – var gadīties tāds skaitļu izkārtojums, kas noved pie neatrisināma uzdevuma.
- Neatrisināms uzdevums ir tad, ja tieši 2 blakus stāvošie skaitļi ir samainījušies vietām.
- Pamēģini, lai sajustu atšķirību!

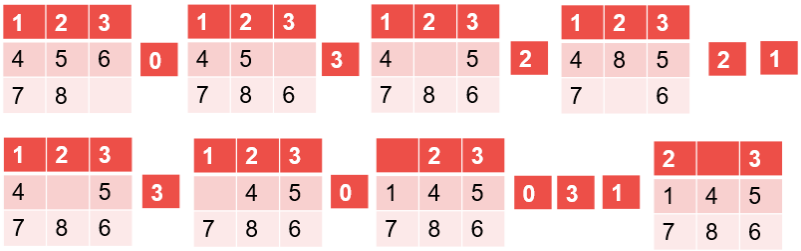


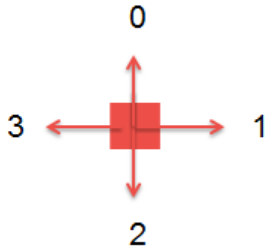
KATRS SKAITLIS TIEŠI VIENREIZ – CITS RISINĀJUMS

- Izjauksim jau saliktu spēli!
- Lai to izdarītu, veiksim noteiktu skaitu gadījuma gājienu.
- Gadījuma gājiens:
  - izvēlamies gadījuma skaitli no 0 lidz 3,

- ja skaitlim atbilstošais gājiens atļauts, to izdara.

Sajaukšanas piemērs 10 soļos:





Sajaukšana notika tikai 7 reizes.

KATRS SKAITLIS TIEŠI VIENREIZ - SĀKUMS

```
int game[][]=new int[4][4];

Random generator =new Random();

int n=0;

for(int column=0;column<4;column++){
    for(int row=0;row<4;row++){
       n++;
       game[row][column]=n;
    }
}
game[3][3]=0;
```

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	0

KATRS SKAITLIS TIEŠI VIENREIZ – SAJAUCAM

```
int x0=3, y0=3;    int dx=0, dy=0;
for (int i=0;i<10;i++){
   int d=generator.nextInt(4);
   switch (d) {
   case 0:
       dy=-1;  break;
   case 1:
       dx=1;   break;
   case 2:
       dy=1;   break;
   case 3:
       dx=-1;  break;
   }
```

```
int xn=x0+dx,   yn=y0+dy;
if(xn>=0 && xn<4 && yn>=0 && yn<4){
    game[x0][y0]=game[xn][yn];
    game[xn][yn]=0;
    x0=xn;  y0=yn;
}
```

ATRISINĀMS UZDEVUMS, BET NEATBILST NOTEIKUMIEM

2		3
1	4	5
7	8	6

- Tukšais lauciņš neatrodas pareizajā vietā.
- Varbūt varam to vienkārši saimainīt ar kauliņu 6?
- To nedrīkst darīt, jo tad atkal var sanākt neatrisināms uzdevums.
- Nāksies ciklu turpināt tik ilgi, kamēr tukšais kauliņš nebūs savā vietā.

KATRS SKAITLIS TIEŠI VIENREIZ UN NULLE VIETĀ

```
int x0=3, y0=3, i=0;
while(i<10|game[3][3]!=0){
        int d=generator.nextInt(4);
        int dx=0, dy=0;
        switch (d) {
        case 0:
            dy=-1;  break;
        case 1:
            dx=1;   break;
        case 2:
            dy=1;   break;
        case 3:
            dx=-1;  break;
        }
    int xn=x0+dx,   yn=y0+dy;
    if(xn>=0 && xn<4 && yn>=0 && yn<4){
    game[x0][y0]=game[xn][yn];
    game[xn][yn]=0;
    x0=xn;  y0=yn;
    i++;
    }
}
```