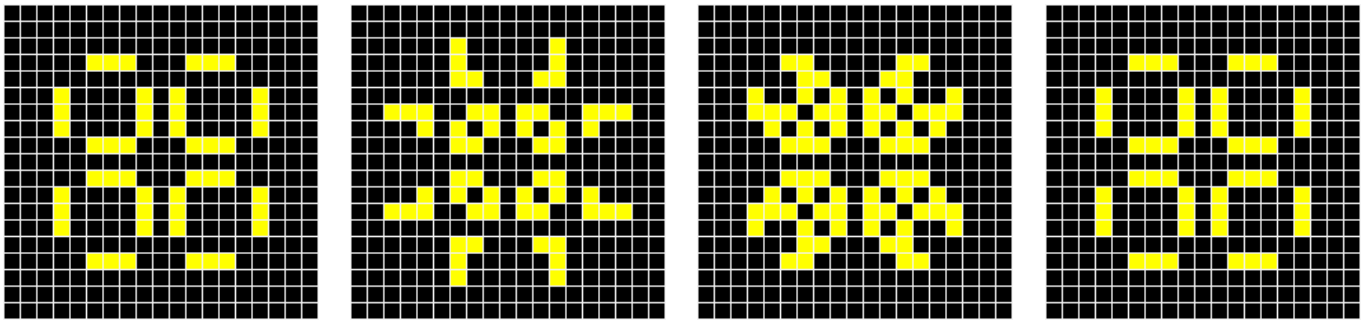
Domáca úloha č. 3

(Termín odovzdania úlohy: **do utorka 14. mája 2024, 09:50**, t. j. do začiatku trinástych cvičení.)

Vytvorte v JavaFX aplikáciu simulujúcu bunkový automat známy ako *Conway's game of Life*, ktorého beh sa často pripodobňuje k vývoju jednoduchého spoločenstva organizmov.[1] Ide o mriežku, ktorej políčka sa nazývajú bunkami. Každá bunka môže byť živá alebo mŕtva – to sa zvyčajne znázorňuje odlišnými farbami príslušných políčok. Za *suseda* bunky sa považuje *iná* bunka, ktorá s danou bunkou zdieľa spoločnú hranu alebo vrchol; každá bunka s výnimkou tých na okraji mriežky[2] má teda presne osem susedov.
Na začiatku je zvolených niekoľko živých buniek tzv. *počiatočnej generácie*. Následne sa v každom kroku behu bunkového automatu vypočíta nová generácia živých buniek, a to podľa nasledujúcich pravidiel:

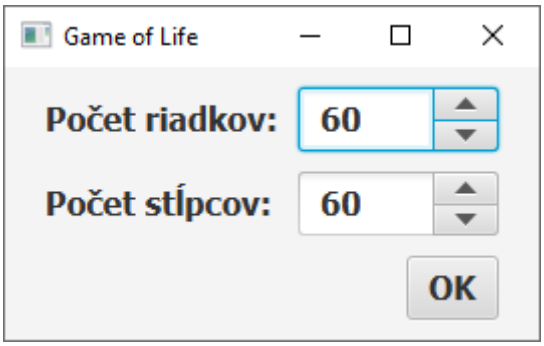
- Každá živá bunka zostane živou aj v nasledujúcej generácii práve vtedy, keď susedí s práve dvoma alebo troma živými bunkami.
- Každá mŕtva bunka sa v nasledujúcej generácii stane živou práve vtedy, keď susedí s práve troma živými bunkami.

Niekoľko po sebe idúcich krokov bunkového automatu *Game of Life* tak môže vyzerať napríklad takto:



Požiadavky na funkcionalitu aplikácie

Aplikácia by po spustení mala zobraziť ovládacie prvky umožňujúce používateľovi zadať požadovaný počet riadkov a stĺpcov mriežky, na ktorej sa bude *Game of Life* simulovať. Ako počet riadkov aj stĺpcov by pritom malo byť možné zadať prinajmenšom ľubovoľnú celočíselnú hodnotu v rozsahu od 10 po 60. Prípadné nekorektné vstupy by mali byť ošetrené „rozumným spôsobom" – aplikácia by teda nemala „spadnúť" ani „zamrznúť". Vhodným ovládacím prvkom na zadávanie počtu riadkov a stĺpcov je napríklad `Spinner<Integer>` alebo nám už známy `TextField`. Po spustení aplikácie by teda jej okno malo vyzerať *napríklad* nejako takto:



Po stlačení potvrdzovacieho tlačidla sa obsah okna prekreslí tak, aby obsahovalo predovšetkým nasledujúce prvky:

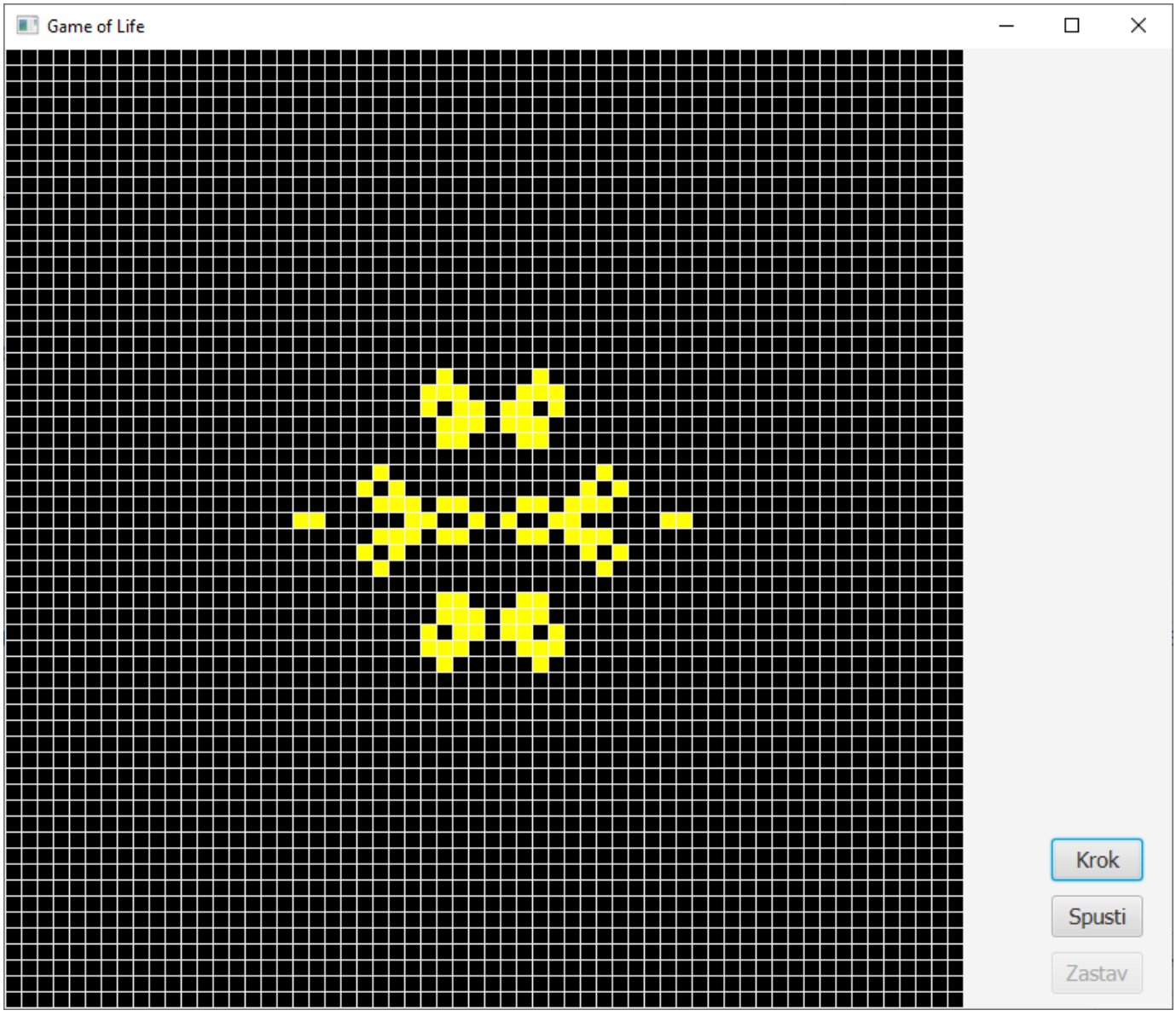
- Obdĺžnikovú mriežku o používateľom zadaných rozmeroch. Bezprostredne po jej vytvorení by mali byť všetky bunky mŕtve. Používateľ by následne mal mať možnosť klikaním na bunky meniť ich stav z mŕtvej na živú a naopak.

[1] Pred riešením samotnej úlohy môže byť prínosné vyhľadať si na internete ďalšie informácie o ňom – je ich k dispozícii neúrekom.
[2] Krajné bunky sa dajú chápať rôznymi spôsobmi (detaily možno nájsť nižšie).

- Tlačidlo, ktoré odsimuluje jeden krok behu bunkového automatu *Game of Life*. Vezme sa teda generácia buniek vykreslená na mriežke a vypočíta sa nasledujúca generácia podľa pravidiel opísaných vyššie. Obsah mriežky sa prekreslí tak, aby zodpovedal tejto novej generácii.
- Tlačidlo, ktoré spustí simuláciu behu bunkového automatu *Game of Life* tak, aby sa jeho kroky vykonávali v pravidelných taktoch (napríklad raz za pol sekundy). Na implementáciu tejto funkcionality použite inštanciu triedy `AnimationTimer`.
- Tlačidlo, ktoré spustenú simuláciu behu bunkového automatu zastaví (môže prípadne ísť aj o to isté tlačidlo ako na spustene simulácie; v každom prípade by ale používateľ nemal mať možnosť požadovať zastavenie ešte nespustenej simulácie, ani spustenie už bežiacej simulácie).

Aj po ukončení simulácie by malo byť možné ďalej modifikovať stavy jednotlivých buniek, prípadne spustiť ďalšiu simuláciu, atď.

Okno aplikácie po vytvorení mriežky a zmene stavov niekoľkých buniek tak môže vyzerať napríklad nasledovne:



Ošetrenie krajných políčok

Pôvodný variant bunkového automatu *Game of Life* predpokladá mriežku nekonečnú vo všetkých štyroch smeroch. V tejto úlohe ale uvažujeme mriežku konečných rozmerov. Správanie buniek na okraji vykreslenej mriežky možno ošetriť rôznymi spôsobmi (môžete si vybrať ktorýkoľvek z nich):

- Možno ich jednoducho považovať za bunky s menším počtom susedov, než pri bunkách vo vnútri mriežky.
- Alternatívne možno políčka na ľavom kraji mriežky považovať za susedov políčok na pravom kraji mriežky a podobne pre políčka na hornom a dolnom kraji mriežky. Všetky políčka tak naozaj budú mať práve osem susedov. Tento prístup možno chápať aj tak, že už ďalej nerobíme v rovine, ale na tore.
- O niečo pokročilejším riešením môže byť „potenciálne nekonečná“ mriežka. Zobrazovať sa v takom prípade bude iba nejaká fixná časť mriežky o používateľom zadaných rozmeroch, pričom tento pohľad je možné v prípade potreby posunúť napríklad pomocou ovládacieho prvku `ScrollBar`. V pamäti sa môže uchovávať napríklad najmenší obdĺžnik, v ktorom sa vyskytujú všetky živé alebo už raz zobrazené bunky.

Požiadavky na spôsob implementácie

- Pozície ovládacích prvkov na scéne nenastavujte manuálne, ale umiestnite ich do oblastí umožňujúcich rozloženie ovládacích prvkov spravovať automaticky.
- Na formátovanie ovládacích prvkov využite v čo možno najväčšej miere JavaFX CSS.
- Po potvrdení rozmerov mriežky používateľom by sa mala veľkosť okna prispôsobiť veľkosti mriežky a novozobrazených tlačidiel. Možno tu využiť metódu `sizeToScene()` inštancie triedy `Stage`, ktorá prispôsobí veľkosť okna jeho scéne. Táto metóda však pracuje správne iba v prípade, že sa pri vytváraní scény jej rozmery *nezadali* manuálne.
- Hodnotiť sa bude aj programátorský štýl a dodržiavanie konvencií jazyka Java.

Možné rozšírenia aplikácie (bonus)

V tejto úlohe je možné získať aj maximálne dva bonusové body za prípadné ďalšie podstatné vylepšenia aplikácie. Za takéto podstatné vylepšenie sa bude považovať napríklad aj „potenciálne nekonečná“ mriežka opísaná vyššie. Ďalšie námety na vylepšenia (nie všetky samé o sebe postačujú na zisk dvoch bonusových bodov a niektoré vyžadujú použitie ovládacích prvkov preberaných až na dvanástej prednáške):

- Možnosť zadávať vlastné pravidlá určujúce, za akých podmienok bunka prežije resp. ožije.
- Možnosť zvoliť si vlastné farby živých a mŕtvych buniek.
- Pridanie hlavnej ponuky (`MenuBar`) obsahujúcej okrem iného aj možnosť na vytvorenie novej mriežky s novými rozmermi.
- Pridanie tlačidla na zmazanie mriežky.
- Možnosť nastavovať rýchlosť simulácie behu bunkového automatu.
- Možnosť uložiť stav bunkového automatu do súboru, resp. načítať ho zo súboru (pri nejakej vhodne zvolenej reprezentácii).
- ...

Odovzdávanie na testovač

Na testovač odovzdávajte ZIP archív obsahujúci práve všetky súbory potrebné na skompilovanie vášho programu. V prípade, že bude tento archív obsahovať väčšie množstvo súborov, odovzdajte v ňom aj textový súbor `readme.txt` stručne opisujúci jeho štruktúru. Testovač nebude vami odovzdanú aplikáciu kompilovať, ani ju inak kontrolovať. Výsledok odovzdania je teda pri tejto úlohe úplne nepodstatný.