Cvičenia č. 3, úloha č. 5

Cieľom tejto úlohy je napísať niekoľko tried reprezentujúcich robotov pohybujúcich sa v bludisku. Pod bludiskom rozumieme dvojrozmerné pole obdĺžnikového tvaru pozostávajúce z políčok troch typov: *prázdnych políčok*, *stien* a *východov z bludiska*. Robot je na začiatku simulácie umiestnený na niektorom z prázdnych políčok a v priebehu simulácie môže vykonávať jednu z nasledujúcich piatich *pohybových akcií*:

- `"UP"`, `"RIGHT"`, `"DOWN"`, resp. `"LEFT"` – t. j. pohyb robota nahor, doprava, nadol, resp. doľava.
- `"SKIP"` – robot v danom kroku simulácie zostáva na mieste.

Na každom políčku bludiska[1] je navyše zaznamenané jedno celé číslo, ktoré môže robot využívať pri rozhodovaní o nasledujúcej akcii a ktoré tiež môže prepísať pomocou akcie `"PUT x"` pre nejaké celé číslo `x`. Pri tejto akcii, ktorú nepovažujeme za pohybovú, si teda robot na svojom políčku zaznamená celé číslo `x`. Na začiatku simulácie je na všetkých políčkach zaznamenané číslo 0.
V archíve priloženom k tejto úlohe na testovači je už okrem iného implementovaná trieda `Robots` načítavajúca vstup, spúšťajúca simuláciu a vypisujúca výstup, rozhranie `Robot` určené na implementáciu triedami reprezentujúcimi robotov, trieda `ProgrammedRobot` implementujúca toto rozhranie, ako aj trieda `Simulation` starajúca sa o samotnú simuláciu pohybu robota v bludisku. Vašou úlohou bude doprogramovať tri ďalšie triedy implementujúce rozhranie `Robot` reprezentujúce rôzne sa chovajúcich robotov.

Obsah archívu na testovači

Priložený archív pozostáva z balíka `robots` obsahujúceho:

- Triedu `Robots` s metódou `main` starajúcou sa predovšetkým o vstup a výstup. Vstupom sú informácie o bludisku, robotovi a počte krokov simulácie; formát vstupu je detailne opísaný nižšie. Po načítaní vstupu metóda `main` spustí simuláciu a priebežne vypisuje na výstup vykonávané pohybové akcie robota.
- Abstraktnú triedu `Square` reprezentujúcu políčko bludiska a jej tri konkrétne podtriedy `EmptySquare`, `Wall` a `Exit` zodpovedajúce prázdnym políčkam, stenám a východom z bludiska. Trieda `Square` deklaruje abstraktné metódy `isAccessible` a `isFinal`, ktoré možno použiť na zistenie, či je príslušné políčko prístupné (t. j. nejde o stenu) resp. či je políčko východom z bludiska. Parametrom konštruktora všetkých týchto tried je celočíselný záznam, ktorý má byť na políčku uložený. K tomu možno neskôr pristupovať metódou `getRecord`. Tento záznam je nemodifikovateľný, čo znamená, že jeho prepísanie robotom sa realizuje pomocou vytvorenia novej inštancie triedy `Square`.
- Rozhranie `Robot`, ktoré musia implementovať všetky triedy reprezentujúce robotov. Toto rozhranie deklaruje jedinú metódu `String nextAction(View view)`. Jej vstupom je inštancia `view` triedy `View`, ktorá reprezentuje „výhľad“ robota z jeho políčka a je iba obalom pre políčko, na ktorom robot momentálne stojí a pre štyri susedné políčka. Metóda `nextAction` pre každého robota na základe jeho „výhľadu“ určí akciu, ktorú robot v nasledujúcom kroku vykoná. Táto akcia je reprezentovaná ako reťazec, ktorým môže byť `"UP"`, `"RIGHT"`, `"DOWN"`, `"LEFT"`, alebo `"SKIP"` pre jednotlivé pohybové akcie, prípadne `"PUT x"` pre nejaké celé číslo `x`.
- Triedu `ProgrammedRobot` implementujúcu rozhranie `Robot` a reprezentujúcu jednoduchého robota, ktorý už v rámci svojho konštruktora dostane pevne danú konečnú postupnosť akcií a túto následne slepo vykonáva. Akonáhle tento robot vykoná všetky akcie zo svojho programu, vykonáva už len akciu `"SKIP"`.
- Triedu `Simulation` starajúcu sa o samotnú simuláciu pohybu robota v bludisku. V rámci konštruktora táto trieda dostane všetky potrebné informácie o bludisku a robotovi. Jej kľúčovou metódou je metóda `nextMove`, ktorá obvykle zodpovedá vykonaniu jednej pohybovej akcie robota a správa sa nasledujúcim spôsobom:
  a) V prípade, že už robot objavil východ z bludiska, vráti metóda reťazec `"FINISHED"`, pričom nevykoná žiadnu ďalšiu činnosť. To znamená, že každá simulácia sa objavením východu *de facto* skončí.

[1] Ako uvidíme neskôr, význam to má iba pri prázdnych políčkach.

b) V opačnom prípade metóda vykonáva akcie robota pomocou jeho metódy `nextAction` do tej doby, kým sa robot pokúsi vykonať *pohybovú akciu*.[2] Táto akcia sa môže alebo nemusí dať vykonať podľa toho, či políčko, na ktoré sa robot hodlá presunúť, je alebo nie je prístupné. V prípade úspechu sa na výstupe metódy `nextMove` vráti vykonaná pohybová akcia. V opačnom prípade sa nevykonateľná pohybová akcia simuláciou ignoruje a výstupom metódy `nextMove` je reťazec `"FAIL"`. Takýto výstup signalizuje len neúspešné vykonanie jednej konkrétnej pohybovej akcie – v simulácii možno pokračovať aj v takom prípade.

Tieto hotové súčasti balíka nemodifikujte. Vašou úlohou bude do balíka `robots` doprogramovať ďalšie triedy. Lepšie pochopenie hotových tried získate zo samotného kódu a z komentárov v ňom.

Formát vstupu a výstupu

Prvou časťou vstupu spracúvaného hotovou triedou `Robots` je špecifikácia bludiska spoločne s východzou pozíciou robota. Vstup sa teda začína dvojicou celých čísel `m` a `n` zodpovedajúcich rozmerom bludiska, za ktorými nasleduje `m` riadkov o `n` znakoch, kde znak `.` zodpovedá prázdnemu políčku, `#` zodpovedá stene, `*` zodpovedá východu z bludiska a `R` zodpovedá počiatočnej pozícii robota. Predpokladáme pritom, že znak `R` sa v tejto „mape bludiska“ nachádza práve raz. Pre jednoduchosť tiež predpokladáme, že `m` ani `n` nie sú menšie ako 3 a že nultý aj posledný riadok aj stĺpec pozostáva výhradne zo znakov `#`. Nie je teda potrebné uvažovať okrajové prípady, pri ktorých by sa robot nachádzal až na samom kraji bludiska.
Za „mapou bludiska“ na vstupe nasleduje názov triedy reprezentujúcej robota, ktorého pohyb v bludisku sa má simulovať. V prípade, že ide o robota triedy `ProgrammedRobot` alebo jeho podtriedy[3], nasleduje riadok obsahujúci dĺžku `commandCount` programu tohto robota a za ním `commandCount` riadkov obsahujúcich jednotlivé akcie tento program tvoriace.
Na poslednom riadku vstupu je vždy celé číslo udávajúce počet krokov vykonávanej simulácie – t. j. počet volaní metódy `nextMove` inštancie triedy `Simulation`.
Výstup tvoria postupne všetky výstupy volaní metódy `nextMove` inštancie triedy `Simulation` (každý na samostatnom riadku).
Príklady vstupov a výstupov možno nájsť nižšie.

Zadanie samotnej úlohy

Doprogramujte do balíka `robots` nasledujúce tri triedy implementujúce rozhranie `Robot`:

- Podtriedu `CowardProgrammedRobot` triedy `ProgrammedRobot` reprezentujúcu programovaného robota, ktorý bude „zbabelo ignorovať“ tie akcie svojho programu, ktoré by viedli k pokusu o vstup do steny. Robot teda bude pracovať rovnako ako robot triedy `ProgrammedRobot`, ale bude preskakovať uvedené akcie odsúdené na neúspech. (Preskočenie akcie tu znamená pokračovanie ďalším príkazom programu, nie nahradenie akciou `"SKIP"`.)

  Trieda `CowardProgrammedRobot` má poskytovať konštruktor s rovnakým argumentom ako u triedy `ProgrammedRobot` a samozrejme vhodným spôsobom prekrývať metódu `nextAction`.

  V rámci triedy `CowardProgrammedRobot` neimplementujte nanovo celú triedu `ProgrammedRobot` – naopak sa snažte maximálne využiť dedenie.

- Triedu `DirectionRotatingRobot` reprezentujúcu robota, ktorý si udržiava premennú reprezentujúcu jeden zo štyroch možných smerov jeho pohybu. Týmto smerom potom robot ide, kým je to možné – t. j. kým posun týmto smerom neznamená pokus o vstup do steny. Pokiaľ už posun daným smerom nie je možný, určí sa nový smer pohybu – a to ako prvý prípustný smer nasledujúci za doterajším v zozname `["UP", "RIGHT", "DOWN", "LEFT"]` chápanom cyklicky, t. j. za `"LEFT"` nasleduje opäť `"UP"`.

  Bezprostredne po vytvorení robota má byť jeho smer nastavený na `"UP"`. V prípade, že žiaden zo štyroch možných smerov pohybu nie je prípustný – čiže robot je „zamurovaný v stene“ – má jeho metóda `nextAction` vracať akciu `"SKIP"`.

  Robot triedy `DirectionRotatingRobot` by sa mal dať vytvoriť volaním konštruktora bez parametrov.

[2] V rámci jedného volania metódy `nextMove` triedy `Simulation` teda robot môže vykonať niekoľko (aj nula) akcií `"PUT x"`, za ktorými nasleduje pokus robota o vykonanie niektorej z akcií `"UP"`, `"RIGHT"`, `"DOWN"`, `"LEFT"`, alebo `"SKIP"`.

[3] Napísať jednu takúto podtriedu je súčasťou úlohy.

- Triedu `DepthFirstSearchRobot`, ktorý prehľadáva bludisko do hĺbky (pričom susedov každého políčka spracúva v poradí `["UP", "RIGHT", "DOWN", "LEFT"]`). Namiesto rekurzie alebo zásobníka, ktorým ste toto prehľadávanie implementovali minulý semester, si však robot bude musieť vystačiť so svojimi záznamami v jednotlivých políčkach bludiska, na základe ktorých bude určovať svoj smer pohybu.

  Pre *pohybové*[4] akcie robota by teda malo platiť nasledujúce:

  a) Robot sa vždy najprv pokúsi vykonať prvú spomedzi akcií `["UP", "RIGHT", "DOWN", "LEFT"]`, ktorou sa dostane na prístupné *a súčasne* ešte nenavštívené políčko.

  b) Ak žiadna takáto akcia neexistuje a robot sa nenachádza na svojom východzom políčku, vykoná pohybovú akciu, ktorá ho vráti späť na políčko, z ktorého svoju momentálnu pozíciu (po prvý raz) objavil.

  c) Ak robot nemôže vykonať akciu ani podľa a), ani podľa b), vykoná akciu `"SKIP"`.

  Robot triedy `DepthFirstSearchRobot` by sa mal dať vytvoriť volaním konštruktora bez parametrov.

Odovzdávanie na testovač

Na testovač odovzdávajte ZIP archív obsahujúci priečinok `robots` a v ňom všetky triedy balíka `robots` (vrátane tých, ktoré už boli hotové). Všetky triedy by mali byť uložené v samostatnom súbore.

Príklad vstupu č. 1:

```
5 6
######
#*...#
#....#
#...R#
######
CowardProgrammedRobot
9
UP
UP
UP
UP
SKIP
LEFT
LEFT
LEFT
LEFT
10
```

Príklad výstupu č. 1:

```
UP
UP
SKIP
LEFT
LEFT
LEFT
FINISHED
FINISHED
FINISHED
FINISHED
```

Príklad vstupu č. 2:

```
5 6
######
#**..#
#.#..#
#..R.#
######
DirectionRotatingRobot
11
```

Príklad výstupu č. 2:

```
UP
UP
RIGHT
DOWN
DOWN
LEFT
LEFT
LEFT
UP
UP
FINISHED
```

[4] Okrem pohybových akcií však tento robot bude musieť vhodným spôsobom vykonávať aj akcie typu `"PUT x"`. Keďže sa tieto akcie nevypisujú na výstup, môžete ich implementovať rôznymi spôsobmi.

Príklad vstupu č. 3:

```
9 8
########
#*.....#
####.###
#....#.#
##.###R#
#..#.#.#
#.##.#.#
#......#
########
DepthFirstSearchRobot
32
```

Príklad výstupu č. 3:

```
UP
DOWN
DOWN
DOWN
DOWN
LEFT
LEFT
UP
UP
DOWN
DOWN
LEFT
LEFT
LEFT
UP
UP
RIGHT
UP
UP
RIGHT
RIGHT
UP
UP
RIGHT
RIGHT
LEFT
LEFT
LEFT
LEFT
LEFT
FINISHED
FINISHED
```