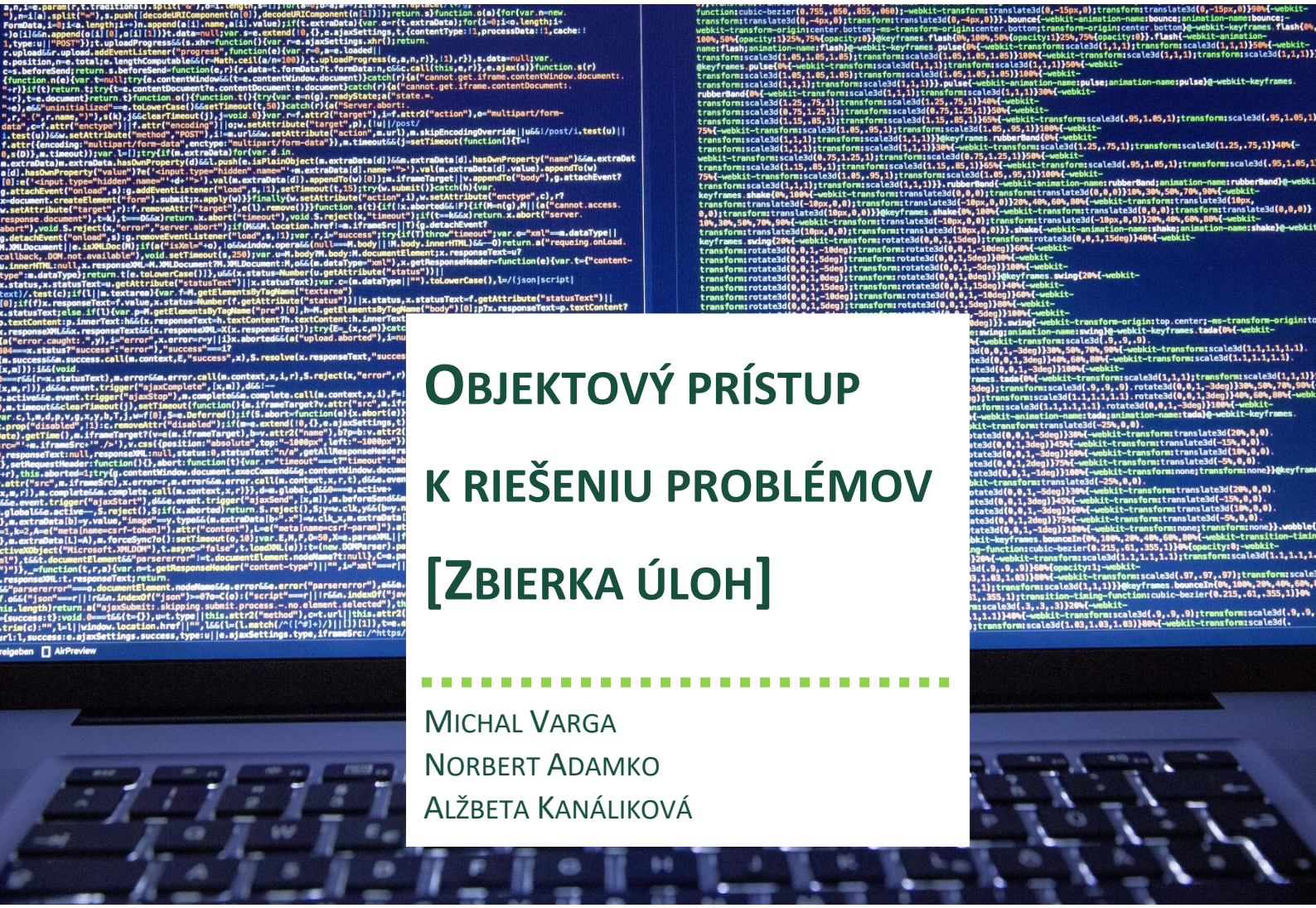


OBJEKTOVÝ PRÍSTUP K RIEŠENIU PROBLÉMOV [ZBIERKA ÚLOH]

MICHAL VARGA
NORBERT ADAMKO
ALŽBETA KANÁLIKOVÁ

EURÓPSKA ÚNIA
Európsky sociálny fond
Európsky fond regionálneho rozvoja

OPERAČNÝ PROGRAM ĽUDSKÉ ZDROJE

MINISTERSTVO ŠKOLSTVA, VEDY, VÝSKUMU A ŠPORTU SLOVENSKEJ REPUBLIKY

it akadémia

Tento projekt sa realizuje vďaka podpore z Európskeho sociálneho fondu a Európskeho fondu regionálneho rozvoja v rámci Operačného programu Ľudské zdroje

www.minedu.sk www.employment.gov.sk/sk/esf/ www.itakademia.sk

Objektový prístup k riešeniu problémov

Spracované v rámci národného projektu IT Akadémia – vzdelávanie pre 21. storočie

Bratislava, 2020

Objektový prístup k riešeniu problémov [pre študentov]

Spracované s finančnou podporou národného projektu IT Akadémia – vzdelávanie pre 21. storočie

Autori: Michal Varga, Norbert Adamko, Alžbeta Kanáliková

Recenzenti: Jana Jurinová, Peter Tomcsányi, Štefan Bocko

Neprešlo jazykovou úpravou.

Vydavateľ: Centrum vedecko-technických informácií SR, Bratislava
Rok vydania: 2020
Vydanie: 1.
ISBN: 978-80-89965-66-3 (verzia pre učiteľov)
EAN: 9788089965663 (verzia pre učiteľov)

Tento projekt sa realizuje vďaka podpore z Európskeho sociálneho fondu v rámci Operačného programu Ľudské zdroje.

OBSAH

1 Úvod do prostredia Greenfoot 5

2 Algoritmus, ovládanie aplikácie, tvorba metód 8

3 Vetvenie a ovládanie hráča 11

4 Premenné, výrazy a pokročilé ovládanie hráča 15

5 Spolupráca objektov tried 18

6 Dedičnosť a cyklus for 20

7 Zoznam a cyklus foreach 26

8 Cyklus while a súkromné metódy 29

9 Polymorfizmus 34

10 Náhodné čísla 38

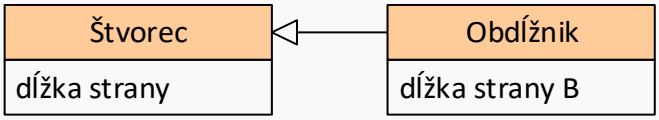
1 ÚVOD DO PROSTREDIA GREENFOOT

ÚLOHA 1.1

Identifikujte vo svojom okolí objekty a vypíšte ich vlastnosti a činnosti, ktoré vedia oni sami vykonať. Dokážete identifikovať objekty, ktoré nemajú žiadne vlastnosti? Dokážete identifikovať objekty, ktoré nedokážu nič vykonať? Dokážete identifikovať nehmotné objekty (také, ktorých sa nevieme fyzicky dotknúť)?

ÚLOHA 1.2

Na nasledujúcom obrázku je hierarchia tried Štvorec a Obdĺžnik. Je takáto hierarchia dobrá?



ÚLOHA 1.3

Vytvorte hierarchiu tried a) dopravných prostriedkov, b) zvierat a c) súčiastok počítača. Uveďte vlastnosti, ktoré dané triedy definujú. Ktoré triedy sú v návrhu abstraktné?

ÚLOHA 1.4

V grafickom editore vytvorte dlaždicu, ktorá bude graficky reprezentovať bunku sveta. Zvoľte štvorcovú reprezentáciu, ideálne 60x60 pixelov. Obrázok importujte alebo uložte v projektovom priečinku do priečinka **`images`**. Nastavte obrázok ako obrázok sveta. Všimnite si, že trieda **`MyWorld`** získala v diagrame tried malú ikonku reprezentujúcu jej grafickú podobu.

ÚLOHA 1.5

Upravte konštruktor triedy **`MyWorld`** tak, aby sa vytvoril svet o veľkosti 25x15 buniek, pričom každá bunka bude veľká 60 pixelov. Ako by bolo potrebné upraviť obrázok, aby svet vyzeral ako šachovnica (teda aby sa striedali rôzne farebné políčka)?

ÚLOHA 1.6

Chyťte vytvorenú inštanciu triedy **`Hrac`** pomocou myši a presuňte ju na inú pozíciu vo svete. Sledujte živý náhľad vnútorného stavu – čo pozorujete? Vytvorte ďalšiu inštanciu triedy **`Hrac`** a zobrazte aj jej vnútorný stav. Opäť presuňte pomocou myši jednu z dvoch inštancií – ktorý vnútorný stav sa zmenil?

ÚLOHA 1.7

Vyvolajte nad rôznymi inštanciami triedy **`Hrac`** metódy, ktoré ponúka nástroj Greenfoot. Sledujte, ako sa mení vnútorný stav inštancie.

ÚLOHY NA PRECVIČOVANIE

ÚLOHA 1.A

Popíšte, aké atribúty majú triedy Strom, Nábytok, Vyučovacia hodina.

ÚLOHA 1.B

Vytvorte hierarchiu tried reprezentujúcu hudobné nástroje. Ktoré triedy budú abstraktné?

ÚLOHA 1.C

Vytvorte svet, ktorého rozmery budú 10x10 buniek a každá bude veľká 50 pixelov.

ÚLOHA 1.D

Zmeňte grafickú reprezentáciu sveta tak, aby vytvoril šachovnicu.

ÚLOHA 1.E

Zmeňte grafickú reprezentáciu objektov triedy **`Hrac`**.

ÚLOHA 1.F

Aké metódy a s akými parametrami musíte zavolať, aby inštancia triedy hráč postupne navštívila všetky 4 rohy sveta?

ÚLOHA 1.G

Aké metódy ponúka trieda **`World`** a jej potomok **`MyWorld`**?

2 ALGORITMUS, OVLÁDANIE APLIKÁCIE, TVORBA METÓD

ÚLOHA 2.1

Zapíšte postup, ako je možné pripraviť kávu, dopraviť sa do školy, uvariť obed.

ÚLOHA 2.2

Zostavte všeobecný algoritmus pre prípravu horúceho nápoja. Zamyslite sa, aké vstupy bude takýto algoritmus potrebovať, aby bol hromadný?

ÚLOHA 2.3

Preskúmajte metódy inštancie triedy **`Hrac`**. Čo pozorujete?

ÚLOHA 2.4

Doplňte do tela metódy **`urobDlhyKrok()`** taký príkaz, aby sa inštancia triedy **`Hrac`** pohla o dve bunky v aktuálnom smere. Následne vytvorte viac inštancií triedy **`Hrac`** a túto metódu vyvolajte na jednotlivých inštanciách. Je správanie očakávané?

ÚLOHA 2.5

Doplňte dokumentačný komentár pre metódu **`urobDlhyKrok()`**.

ÚLOHA 2.6

Upravte dokumentačný komentár triedy **`Hrac`**. Vypíšte verziu triedy a jej autora.

ÚLOHA 2.7

Preskúmajte dokumentačné okno.

ÚLOHA 2.8

Upravte telo metódy **`act()`** triedy **`Hrac`** tak, aby sa zavolala metóda **`urobDlhyKrok()`**.

ÚLOHA 2.9

Vyskúšajte tlačidlá na ovládanie aplikácie. Vytvorte si viac inštancií triedy **`Hrac`**. Stlačte tlačidlo **`Act`** – čo sa stane? Stlačte tlačidlo **`Run`** – čo sa stane? Po prvom posune tlačidla **`Run`** stlačíme tlačidlo **`Pause`**, čo sa stane? Aký vplyv má posuvník **`Speed`** na volanie metódy **`act()`** po stlačení tlačidla **`Run`**?

ÚLOHA 2.10

Pridajte do triedy **`Hrac`** metódu, pomocou ktorej bude inštancia triedy **`Hrac`** chodiť do zvoleného štvorca. Zdokumentujte svoju metódu. Na pohyb a otáčanie využite vhodné metódy z predka **`Actor`**. Upravte metódu **`act()`** tak, aby inštancia triedy **`Hrac`** chodila po jej vyvolaní do štvorca. Potom svoje riešenie overte spustením aplikácie.

ÚLOHY NA PRECVIČOVANIE

ÚLOHA 2.A

Popíšte algoritmus, ktorým by ste navigovali cudzinca z vašej školy na vlakovú stanicu, do reštaurácie, ku vám domov.

ÚLOHA 2.B

Preskúmajte dokumentáciu triedy **`Actor`**.

ÚLOHA 2.C

Napíšte a zdokumentuje metódu **`chodDoSestuholnika()`**, pomocou ktorej prejde inštancia triedy **`Hrac`** po dráhe v tvare šesťuholníka. Na pohyb využite metódu **`urobDlhyKrok()`**.

ÚLOHA 2.D

Vytvorte metódy **`dopravaDole()`**, **`dopravaHore()`**, **`dolavaDole()`** a **`dolavaHore()`**, ktoré posunú inštanciu triedy **`Hrac`** v danom smere. Využite metódu **`setRotation()`**, preskúmajte jej činnosť v dokumentácii.

3 VETVENIE A OVLÁDANIE HRÁČA

ÚLOHA 3.1

Upravte kód metódy **`act()`** v triede **`Hrac`** tak, aby sa hráč hýbal iba vtedy, keď je stlačený kláves P (ako pohyb). Zachovajte kód zodpovedný za otočenie hráča, keď dôjde na kraj sveta, ale zamyslite sa na jeho umiestnení. Kedy sa môže vykonať otočenie hráča?

ÚLOHA 3.2

Vytvorte inštanciu triedy **`Hrac`** a umiestnite ju do stredu hracej plochy. Otvorte okno s vnútorným stavom inštancie a umiestnite ho tak, aby bolo viditeľné počas behu aplikácie. Potom spustite aplikáciu a pozorujte, ako sa menia hodnoty atribútov **`x`** a **`y`** v triede **`Hrac`**. Ako sa menia tieto hodnoty pri pohybe nahor, nadol, doľava a doprava?

ÚLOHA 3.3

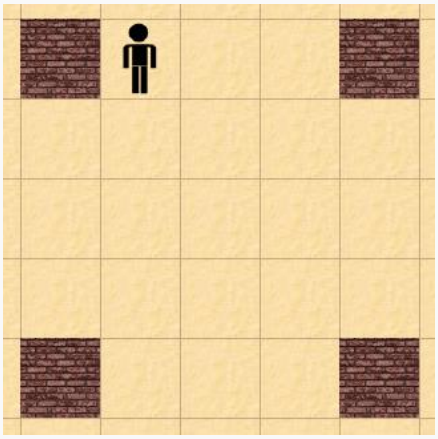
Doplňte do tela metódy **`act()`** kód na správne natočenie hráča po dosiahnutí dolného a ľavého okraja sveta.

ÚLOHA 3.4

Vytvorte dve nové triedy, potomkov triedy **`Actor`**. Prvá bude trieda **`Mur`** a druhá trieda bude **`Stena`**. Pripravte si v grafickom editore vhodné obrázky s veľkosťou 60x60 pixelov. Tieto obrázky potom priraďte novovytvoreným triedam.

ÚLOHA 3.5

Vytvorte štyri inštancie triedy **`Mur`** a jednu inštanciu triedy **`Hrac`** tak, ako je to zobrazené na obrázku nižšie. Odhadnite, ako sa bude hráč pohybovať? Spustite aplikáciu. Zhoduje sa Vaša predpoveď s tým, čo pozorujete?



Obrázok 3.1: Rozostavenie múrov a hráča

ÚLOHA 3.6

Pridajte do metódy **`act()`** triedy **`Hrac`** kód, ktorý zabezpečí, aby sa hráč otočil o 90° proti smeru hodinových ručičiek, keď vojde na políčko, kde sa nachádza inštancia triedy **`Stena`**.

ÚLOHA 3.7

Predpovedzte, ako sa bude pohybovať inštancia triedy **`Hrac`**. Zhoduje sa výsledok s predpoveďou?

ÚLOHA 3.8

Postupne umiestnite jednu inštanciu triedy **`Hrac`** do rohov sveta. Predpovedzte, ako sa bude táto inštancia pohybovať po spustení aplikácie. Zhoduje sa výsledok s predpoveďou?

ÚLOHA 3.9

Doplňte kaskádu podmienok tak, aby sa kontroloval dotyk s inštanciou triedy **`Mur`** a triedy **`Stena`** iba vtedy, ak sa hráč nenachádza na okrajoch sveta. Najskôr kontrolujte inštanciu triedy **`Mur`**.

ÚLOHA 3.10

Vytvorte metódu pre automatický pohyb inštancie triedy **`Hrac`**. Presuňte do nej všetok kód z metódy **`act()`**. Identifikátor metódy môže byť napríklad **`pohybujSaAutomaticky`**.

ÚLOHA 3.11

Vytvorte v triede **`Hrac`** metódu **`pohybujSaSipkami()`**. Naprogramujte túto metódu tak, aby sa hráč hýbal iba vtedy, keď je stlačená šípka. Pohybovať sa bude v smere stlačenej šípky. Dbajte na efektívnosť kódu. V metóde **`act()`** vyvolajte túto metódu.

ÚLOHA 3.12

Pripravte si štyri obrázky pre hráča pre pohyb smerom hore, dole, doľava a doprava. **Rozmery obrázkov nesmú presiahnuť rozmery bunky**, v našom prípade 60x60 pixelov. Pripravené obrázky vložte do projektového adresára **`images`**.

vpravo.png vlavo.png hore.png dole.png

Obrázok 3.2: Obrázky hráča smerujúceho do rôznych smerov [4]

ÚLOHA 3.13

Vytvorte metódu **`aktualizujObrazok()`**, ktorá zmení obrázok hráča podľa jeho aktuálneho otočenia. Doplňte volanie tejto metódy do tela metódy **`act()`**.

ÚLOHA 3.14

Spustite aplikáciu. Vidíme, že obrázky sa nastavujú, ale otáčajú sa podľa toho, ako je otočený hráč. Vyriešte tento problém.

ÚLOHA 3.15

Vyskúšajte vytvoriť viacerých hráčov a ovládajte ich pomocou klávesnice. Čo pozorujete?

ÚLOHY NA PRECVIČOVANIE

ÚLOHA 3.A

Vytvorte metódu, ktorá po spustení tlačidlom nechá hráča kráčať po jednej bunke až k okraju sveta. Po jeho dosiahnutí sa zmení obrázok hráča na iný obrázok.

ÚLOHA 3.B

Vytvorte metódu **`skusZmenitObrazok()`**, tak aby sa po stlačení klávesu "z" zobrazila žena, po stlačení klávesu "m" zobrazil muž, a po stlačení klávesu "d" zobrazilo dieťa. Nezabudnite si uložiť potrebné obrázky do priečinka **`images`**.

ÚLOHA 3.C

Nechajte hráča otáčať o určitý uhol po stlačení klávesu "p" smerom doprava a po stlačení klávesu "l" smerom doľava. Kód doplňte na vhodné miesto.

ÚLOHA 3.D

Upravte kód v metóde **`pohybujSaAutomaticky()`** tak, že ak hráč dosiahne okraj sveta v strede (teda súradnice [0; 7], [24; 7], [12, 0] alebo [12, 15]) bude pomocou metódy **`setLocation()`** premiestnený do stredu sveta (na súradnice 12, 7).

4 PREMENNÉ, VÝRAZY A POKROČILÉ OVLÁDANIE HRÁČA

ÚLOHA 4.1

Aký je rozdiel medzi nasledovnými algoritmami?

```
1.   int a;
     boolean c;
     ...
     if(a > 0){c = true;}
     if(a < 0){c = false;}

2.   int a;
     boolean c;
     ...
     if(a > 0){c = true;}
     else {c = false;}
```

ÚLOHA 4.2

Napíšte prostredníctvom logických a relačných operátorov výraz vyjadrujúci, že premenná **`int a`** má hodnotu patriacu do nasledovných intervalov: <-10,10), (5,142), (-11,-3) OR (1,25>.

ÚLOHA 4.3

Zvoľte si hodnoty premenných x a y a dosaďte ich do výrazov. Aké budú hodnoty premenných b1 a b2 v jednotlivých prípadoch (1 a 2)?

```
1.   int x, y;
     ...
     boolean b1 = x > 0 && y == 1;
     boolean b2 = x <= 0 || y <= 0;

2.   int x, y;
     ...
     boolean b1 = (x > 0) && (y == 1);
     boolean b2 = (x <= 0) || (y <= 0);
```

ÚLOHA 4.4

Otestujte si vytvorený konštruktor a upravenú metódu pre ovládanie pohybu hráča vložením dvoch inštancií triedy **`Hrac`** do sveta. Pre každú inštanciu v dialógu nastavte iné klávesy pre ovládanie jej pohybu. Vyskúšajte, či je možné vložených hráčov ovládať nezávisle.

ÚLOHA 4.5

Premenujte triedu **`MyWorld`** na triedu **`Arena`**.

ÚLOHA 4.6

Pridajte do sveta ďalšieho hráča, napríklad pomocou referenčného atribútu **`hrac2`**, ktorý bude mať ovládanie pomocou kláves w - hore, s - dole, d - vpravo a a - vľavo. Hráča vložte na súradnice [24,14]. Po pridaní vyskúšajte ovládanie.

ÚLOHA 4.7

Čo by sa stalo, keby sme po vytvorení oboch hráčov vykonali priradenie **`this.hrac1 = this.hrac2;`**? Bol by to problém?

ÚLOHA 4.8

Rozšírte triedu **`Hrac`** o ďalší atribút typu **`int`** reprezentujúci veľkosť kroku hráča.

ÚLOHA 4.9

Upravte metódu **`pohybujSaKlavesmi()`** tak, aby rešpektovala zvýšenú veľkosť kroku. Vytvorte novú inštanciu triedy **`Hrac`** a otestujte funkčnosť programu.

ÚLOHA 4.10

Vašou úlohou je zabezpečiť, aby sa hráč pohyboval vždy po jednej bunke, no s rôznou rýchlosťou. Každý hráč môže mať inú rýchlosť. Ako pomôcku uvádzame, že rôznu rýchlosť pohybu je možné naprogramovať napríklad tak, že hráča neposuniete pri každom zistení stlačenia klávesu (zisťovanie robíme v metóde act(), takže ak držíte kláves stlačený, jeho stlačenie zistíte pri každom jej vykonaní), ale až pri každom N-tom. Čím bude N väčšie, tým bude rýchlosť hráča nižšia. Ako zadáte N? Kde si ho uložíte? Ako budete vedieť, koľko stlačení klávesu už ubehlo? (Pomôcka: je potrebné aj počítadlo).

ÚLOHY NA PRECVIČOVANIE

ÚLOHA 4.A

Doplňte ďalšie vlastnosti hráča podľa vlastného uváženia. Čím by sa hráči mohli v hre odlišovať? Aké ďalšie vlastnosti či činnosti bude hráč potrebovať?

5 SPOLUPRÁCA OBJEKTOV TRIED

ÚLOHA 5.1

Analogicky doplňte kód pre smery hore a dole (budete teda meniť hodnotu lokálnej premennej **`y`**).

ÚLOHA 5.2

Upravte metódu zabezpečujúcu pohyb hráča tak, aby ste pred zmenou jeho polohy overili možnosť vstupu do cieľovej bunky.

ÚLOHA 5.3

Upravte metódu **`mozeVstupit()`** tak, aby hráč reagoval aj na múry a nemohol cez ne prejsť.

ÚLOHA 5.4

Vytvorte novú triedu **`Bomba`**, navrhnite jej atribút, ktorý bude reprezentovať silu výbuchu. Vytvorte parametrický konštruktor a inicializujte atribúty objektu.

ÚLOHA 5.5

Doplňte do triedy **`Hrac`** bezparametrickú metódu **`mozePolozitBombu()`** s návratovou hodnotou typu **`boolean`**, ktorá vráti príznak, či je možné položiť bombu na políčko, kde aktuálne stojí hráč. Bombu je možné položiť vtedy, keď je stlačený príslušný kláves a keď sa na políčku nenachádza iná bomba.

ÚLOHA 5.6

Rozšírte hru tak, aby bol výbuch bomby sprevádzaný zvukovým efektom. Zvuk si môžete nahrať alebo stiahnuť z internetu. Nájdite príkaz na prehratie zvuku v dokumentácii triedy **`Greenfoot`**.

ÚLOHY NA PRECVIČOVANIE

ÚLOHA 5.A

Skúste si nahrať vlastný zvuk a použite ho napríklad pri položení bomby. Zvuky môžete nahrávať aj v prostredí Greenfoot. (**`Nástroje | Show sound recorder)`**.

ÚLOHA 5.B

Skúste porozmýšľať ako by bomba zničila aj samotného hráča. Čo je potrebné zmeniť, dopracovať.

6 DEDIČNOSŤ A CYKLUS FOR

ÚLOHA 6.1

Vytvorte triedu **`Prekazka`**. Aká trieda je predkom triedy **`Prekazka`**? Upravte hlavičky tried **`Stena`** a **`Mur`** tak, aby boli potomkami triedy **`Prekazka`**.

ÚLOHA 6.2

Po pridaní predka **`Prekazka`** je možné jednoduchšie otestovať, či môže hráč vstúpiť na danú bunku. Svet vo svojej metóde **`getObjectsAt()`** požaduje ako tretí parameter triedu, ktorú má na danej bunke hľadať. Keďže sú aj steny aj múry prekážkami, je možné sa k nim správať jednotne. Upravte metódu **`mozeVstupit()`** v triede **`Hrac`** tak, aby ste využili iba jediný zoznam prekážok.

ÚLOHA 6.3

Upravte triedu **`Arena`** tak, aby mal jej konštruktor dva parametre reprezentujúce šírku a výšku. Upravte volanie predkovho konštruktora v triede **`Arena`** tak, aby prebral tieto parametre. Odstráňte z tohto konštruktora kód zodpovedný za tvorbu a umiestnenie hráčov do arény, budú to vykonávať potomkovia. Takisto môžete odstrániť deklaráciu atribútov typu **`Hrac`** z triedy **`Arena`**.

ÚLOHA 6.4

Upravte konštruktor triedy **`TestovaciaArena`** tak, aby vytvoril prázdnu arénu s veľkosťou 7x7 buniek. Na overenie vytvorte inštanciu triedy **`TestovaciaArena`**.

ÚLOHA 6.5

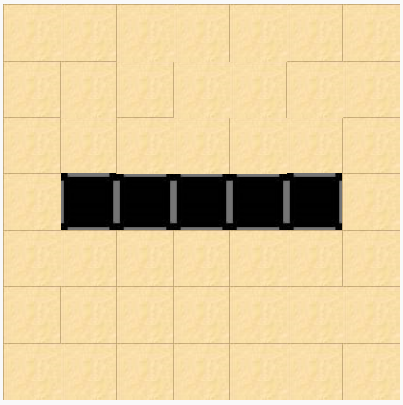
Pridajte do triedy **`TestovaciaArena`** metódu **`ukazRozmery()`**, ktorá vypíše na obrazovku rozmery arény.

ÚLOHA 6.6

Pridajte do triedy **`Hrac`** metódu **`ukazRozmery()`**, ktorá zobrazí rozmery arény, ak sa nachádza v testovacej aréne. Ak áno, využite metódu **`ukazRozmery()`** triedy **`TestovaciaArena`**.

ÚLOHA 6.7

Upravte konštruktor triedy **`TestovaciaArena`** tak, aby vytvoril arénu s rozmermi 7x7 buniek so stenami, ktoré budú rozložené takto:



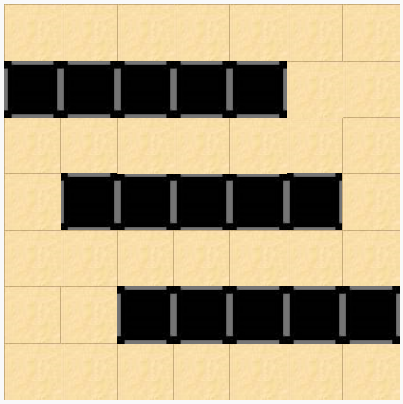
Pripomeňme si, že na vloženie inštancie triedy **`Actor`** do sveta (teda do potomkov triedy **`World`** a v našom prípade **`Arena`**) môžeme využiť metódu **`addObject()`** triedy **`World`**, ktorá má tri parametre:

- Aktor, ktorý má byť vložený
- x-ová súradnica bunky, na ktorú má byť vložený (teda index stĺpca číslovaný od 0)
- y-ová súradnica bunky, na ktorú má byť vložený (teda index riadku číslovaný od 0)

Pozícia [0;0] sa vo svete nachádza vľavo hore.

ÚLOHA 6.8

Upravte konštruktor triedy **`TestovaciaArena`** tak, aby boli bunky rozložené tak, ako je to zobrazené na obrázku.



ÚLOHA 6.9

Zamyslime sa, koľko informácií potrebujeme na to, aby sme vedeli vytvoriť akýkoľvek rad stien idúcich za sebou?

ÚLOHA 6.10

Deklarujte v predkovi **`Arena`** metódu **`vytvorRiadokStien()`**, ktorá bude mať tri parametre:

- Riadok (horný riadok má index 0), na ktorom sa majú začať vytvárať steny.
- Stĺpec (ľavý stĺpec má index 0), od ktorého sa majú začať vytvárať steny.
- Počet vyjadrujúci, koľko stien za sebou má byť vytvorených.

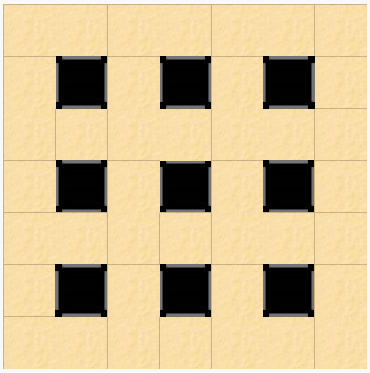
Metóda nemá návratovú hodnotu (použijeme teda kľúčové slovo **`void`**).

ÚLOHA 6.11

Upravte kód v konštruktore potomka triedy **`Arena`** tak, aby využíval metódu **`vytvorRiadokStien()`**.

ÚLOHA 6.12

Upravte konštruktor triedy **`TestovaciaArena`** tak, aby vytvoril arénu zobrazenú na obrázku nižšie. Upravte na to metódu **`vytvorRiadokStien()`** tak, aby mala štvrtý parameter definujúci medzery medzi stenami.



ÚLOHA 6.13

Zamyslime sa, koľko a akých informácií potrebujeme na to, aby sme vedeli vytvoriť akýkoľvek rad stien v obdĺžniku, ktorého počiatočný bod bude možné zadať a ktorému bude možné nastaviť medzery medzi stenami v riadkoch aj v stĺpcoch?

ÚLOHA 6.14

Deklarujte v predkovi **`Arena`** metódu **`vytvorObdlznikStien()`**, ktorá bude mať nasledujúce parametre:

- Riadok (horný riadok má index 0), od ktorého má začať vytvárať steny.
- Stĺpec (ľavý stĺpec má index 0), od ktorého má začať vytvárať steny.
- Počet riadkov, koľko má byť vytvorených.
- Počet stien, koľko má byť za sebou v riadku vytvorených.
- Veľkosť medzery medzi riadkami.
- Veľkosť medzery medzi stenami v riadku.

Metóda nemá návratovú hodnotu.

ÚLOHA 6.15

Upravte konštruktor potomka triedy **`Arena`** tak, aby využil metódu **`vytvorObdlznikStien()`** a rozložil arénu tak, ako je zobrazená v úlohe 6.12.

ÚLOHA 6.16

Vytvorte vo svojej aréne niekoľko múrov a pridajte dvoch hráčov. Otestujte funkčnosť hry, teda či hráči nevojdú ani do steny a ani do múru.

ÚLOHY NA PRECVIČOVANIE

ÚLOHA 6.A

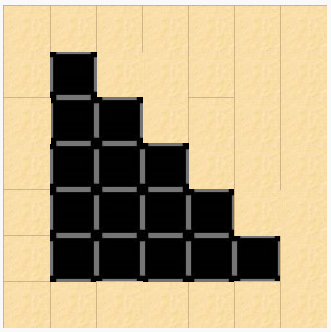
Pridajte do triedy **`Arena`** metódu, ktorá bude mať celočíselné parametre **`naKtoromStlpci`**, **`odKtorehoRiadku`**, **`kolko`** a **`medzery`**. Metóda nech vytvorí stĺpec stien, podľa zadaných parametrov. Medzera určuje, koľko polí je medzi stenami voľných.

ÚLOHA 6.B

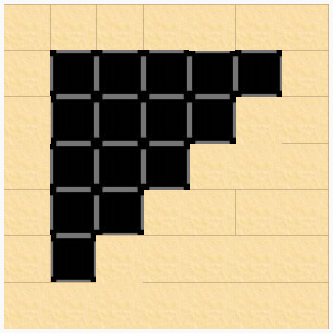
Pridajte do triedy **`Arena`** metódy, ktoré budú schopné vygenerovať a) riadok a b) obdĺžnik zložený z múrov. Parametre metód sú zhodné ako parametre metód generujúcich steny.

ÚLOHA 6.C

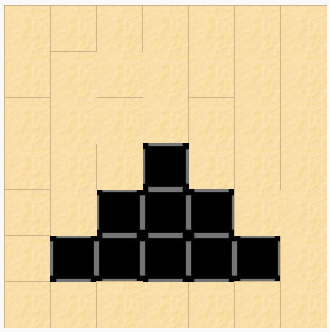
Pridajte do trieda **`Arena`** metódy, ktoré vygenerujú nasledujúce trojuholníky stien (medzi riadkami a stĺpcami bude možné zadať medzery). Parametre metód vhodne navrhnite.



a



b



c

ÚLOHA 6.D

Vytvorte triedy **`PravidelnaArena`** a **`MurovanaArena`** ako potomkov triedy **`Arena`**, pričom rozloženie stien, múrov a hráčov bude také, ako na obrázku 6.2.

7 ZOZNAM A CYKLUS FOREACH

ÚLOHA 7.1

Vytvorte v triede **`Arena`** bezparametrickú metódu **`koniecHry()`**, ktorá zistí, či nastal koniec hry (ostal iba jeden alebo žiadny hráč) a v návratovej hodnote typu **`boolean`** oznámi, či sa tak stalo. Zatiaľ predpokladajme, že koniec hry nenastane nikdy.

ÚLOHA 7.2

Pridajte do triedy **`Arena`** metódu **`act()`**. V nej skontrolujete, či nastal koniec hry (pomocou metódy **`koniecHry()`**) a ak áno, zastavte hru. Pre zastavenie činnosti prostredia Greenfoot využite príkaz **`Greenfoot.stop();`** .

ÚLOHA 7.3

Pridajte do triedy **`Arena`** atribút **`zoznamHracov`** typu **`LinkedList<Hrac>`**. Nezabudnite, že musíte dopísať import balíčka s triedou **`LinkedList`**. Inicializujte inštanciu atribút **`zoznamHracov`** v konštruktore triedy **`Arena`**.

ÚLOHA 7.4

Pridajte do triedy **`Arena`** metódu **`zaregistrujHraca()`**, ktorá preberie jediný parameter typu **`Hrac`** a pomocou metódy **`add()`** ho vloží na koniec zoznamu **`zoznamHracov`**.
Upravte potomkov triedy **`Arena`** tak, aby hráča po vložení hráča do sveta na správne miesto aj zaregistrovali u predka.

ÚLOHA 7.5

Pridajte do triedy **`Arena`** metódu **`odregistrujAOdstranHraca()`**, ktorá má jediný parameter typu **`Hrac`**. Metóda odstráni hráča zo zoznamu hráčov a následne ho odstráni zo sveta.

ÚLOHA 7.6

Implementujte telo metódy **`koniecHry()`** tak, aby metóda vrátila **`true`** vtedy, keď sa v hre nachádza jeden alebo žiadny hráč. Využite vhodné metódy zoznamu.

ÚLOHA 7.7

Upravte kód v metóde **`act()`** v triede **`Bomba`** tak, aby predtým, ako sa bomba odstráni zo sveta, získala všetkých hráčov, ktorí sú od nej vzdialení najviac o hodnotu jej sily. Metóda **`getObjectsInRange()`** vracia zoznam typu **`List`**. Jeho generický parameter je zhodný s triedou, ktorá je poslaná ako druhý parameter.

ÚLOHA 7.8

Pomocou cyklu **`for`** prejdite všetkých hráčov v zozname zasiahnutých hráčov. Každého hráča odregistrujte v triede **`Arena`**.

ÚLOHA 7.9

Vytvorte v triede **`Hrac`** metódu **`zasah()`**, ktorá bude vyvolaná hráčovi bombou po tom, ako ho bomba zasiahne. Hráč sa v tejto metóde odregistruje zo sveta. Upravte kód v metóde **`act()`** triedy **`Bomba`** tak, aby odrážala novú funkčnosť.

ÚLOHA 7.10

Vytvorte v triede **`Bomba`** metódu **`zrusVlastnika()`**, ktorá nastaví jej atribút **`vlastnik`** na **`null`**.

ÚLOHA 7.11

Vytvorte v triede **`Hrac`** atribút **`zoznamAktivnychBomb`** typu **`LinkedList<Bomba>`**. Inicializujte ho v správnom konštruktore. Upravte telá metód podľa nasledujúcich pravidiel:

- V metóde **`act()`** zaregistrujte novo vytvorenú bombu do zoznamu **`zoznamAktivnychBomb`.**
- V metóde **`vybuchlaBomba()`** odstráňte zo zoznamu **`zoznamAktivnychBomb`** bombu, ktorá prišla ako parameter (tá vybuchla).
- V metóde **`zasah()`** pomocou cyklu **`foreach`** zrušte vlastníka všetkým bombám zo zoznamu **`zoznamAktivnychBomb`**.

ÚLOHY NA PRECVIČOVANIE

ÚLOHA 7.A

Upravte výbuch bomby tak, aby zo sveta odstránila aj všetky inštancie triedy **`Mur`** v jej dosahu.

ÚLOHA 7.B

Pridajte hráčovi životy. Počiatočný počet životov môže byť pevne stanovený alebo ho môžete zadať v parametri konštruktora. Hráč je odregistrovaný zo sveta až po tom, ako bol zasiahnutý určený počet krát. Upravte metódu **`zasah()`** v triede **`Hrac`**.

ÚLOHA 7.C

Rozšírte hráča o možnosť okamžitej jednorazovej detonácie bômb. Definujte klávesu, ktorou odpálite všetky aktívne hráčove bomby. Umožnite hráčovi využiť túto schopnosť iba raz.

ÚLOHA 7.D

Upravte hru tak, aby aréna oživila vypadnutého hráča (pri hre viac ako dvoch hráčov), ak hra trvá posledným dvom hráčom pridlho. Hráča oživte na pozícií podľa svojho výberu.

8 CYKLUS WHILE A SÚKROMNÉ METÓDY

ÚLOHA 8.1

Vytvorte triedu **`Ohen`**. Zvoľte vhodnú grafickú reprezentáciu. Konštruktor tejto triedy má jeden parameter, ktorý určuje, ako dlho oheň na svojom mieste horí. Zabezpečte, aby oheň po prejdení daného času zmizol zo sveta.

ÚLOHA 8.2

Upravte existujúci kód výbuchu bomby tak, aby na mieste bomby zanechal inštanciu triedy **`Ohen`**. Otestujte svoje riešenie.

ÚLOHA 8.3

Pomocou cyklu **`for`** rozšírte výbuch bomby (vytvorte inštancie triedy **`Ohen`**) v smere doprava od bomby. Výbuch rozšírte na toľko buniek, koľko udáva atribút **`sila`** bomby.

ÚLOHA 8.4

Upravte výbuch bomby tak, aby generoval ohne do všetkých strán. Pomôžte si zmenou súradníc zobrazenou na predchádzajúcom obrázku.

ÚLOHA 8.5

Prepíšte všetky cykly **`for`** pre šírenie ohňa s využitím cyklu **`while`**. Druhú časť podmienky (je možné do nasledujúcej bunky položiť oheň) zatiaľ vynechajte.

ÚLOHA 8.6

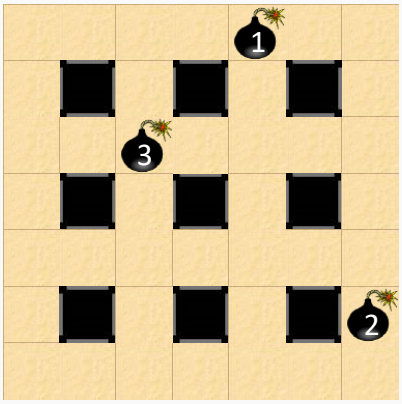
Využite súkromnú metódu **`rozsirOhen()`** na rozšírenie ohňa po výbuchu bomby.

ÚLOHA 8.7

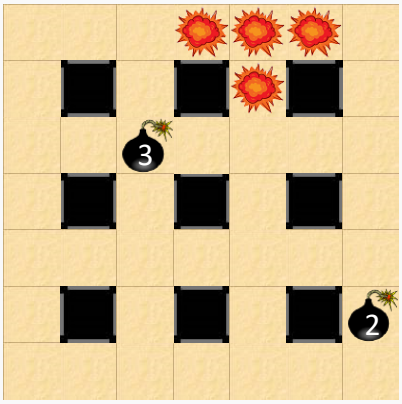
Vytvorte v triede **`Bomba`** súkromnú metódu **`mozeBunkaVybuchut()`**, ktorá v parametroch preberie súradnice riadku a stĺpca a vráti **`true`**, ak na danej bunke môže výbuch nastať podľa prvých dvoch podmienok vyššie. Ak to nie je možné, metóda vráti **`false`**.

ÚLOHA 8.8

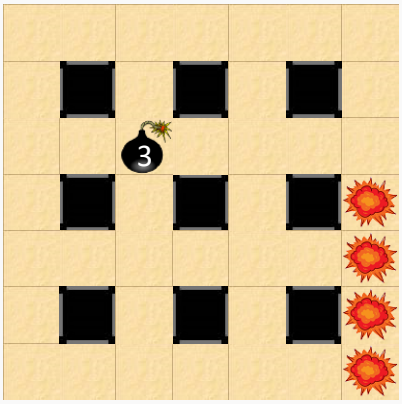
S využitím metódy **`mozeBunkaVybuchnut()`** môžete teraz upraviť podmienku v cykle **`while`** v metóde **`rozsirOhen`** v triede **`Bomba`**. Upravte podmienku v cykle tak, aby sa rešpektoval výsledok kontroly z metódy **`mozeBombaVybuchnut()`**. Otestujte funkčnosť riešenia s bombami s rôznou silou medzi stenami. Testy rôznych výbuchov sú zobrazené na nasledujúcich obrázkoch:



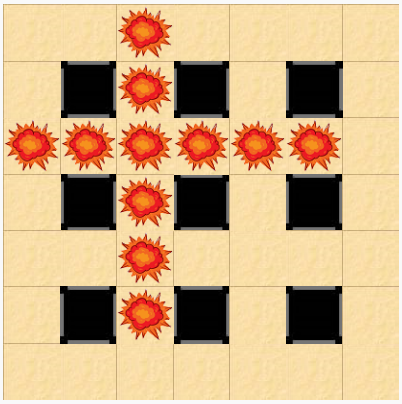
a



b



c



d

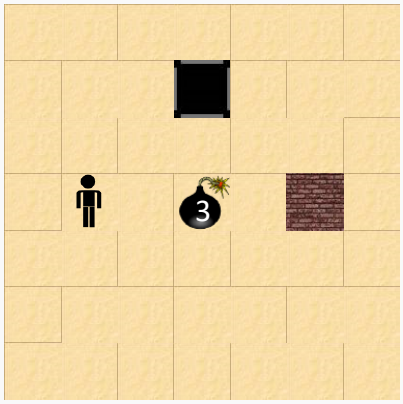
V časti (a) vidíme pôvodné rozloženie, v časti (b) vybuchla bomba so silou 1, v časti (c) vybuchla bomba so silou 2 a v časti (d) bomba so silou 3.

ÚLOHA 8.9

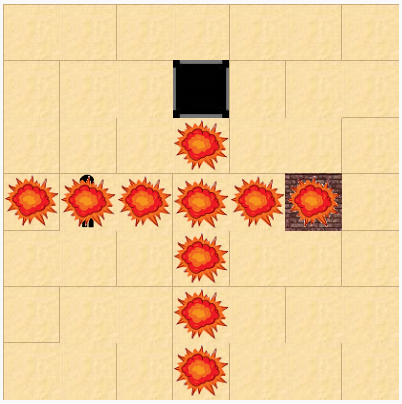
Pridajte do triedy **`Bomba`** súkromnú metódu **`mozeVybuchPokracovat()`**, ktorá v parametroch preberie súradnice riadku a stĺpca a vráti **`true`**, ak bunka na daných súradniciach nezastavila výbuch. Ak bunka výbuch zastavila, tak metóda vráti **`false`**. Výbuch nemôže pokračovať, ak narazil na múr.

ÚLOHA 8.10

S využitím metódy **`mozeVybuchPokracovat()`** upravte metódu **`rozsirOhen()`** v triede **`Bomba`**. Ak výbuch môže ďalej z danej bunky pokračovať, zvýšime hodnotu premennej **`i`** o 1 a prepočítame súradnice nového riadku a stĺpca výbuchu. Inak umelo zvýšime hodnotu premennej **`i`** na hodnotu vyššiu ako je sila bomby, čím sa zastaví cyklus. Otestujte svoje riešenie na situácií z nasledovného obrázku:



a



b

ÚLOHA 8.11

Upravte správanie inštancie triedy **`Bomba`** tak, aby nevolala metódu **`zasah()`** hráčov v jej okolí. Namiesto toho bude hráč sám kontrolovať, či sa neprekrýva s ohňom. Upravte správanie hráča v jeho metóde **`act()`** tak, aby najskôr zistil, či sa neprekrýva s inštanciou triedy **`Ohen`**. Ak áno, sám vyvolá vlastnú metódu **`zasah()`**. Takto zabezpečíme, aby bol hráč zasiahnutý aj ohňom, ktorý horí po výbuchu bomby.

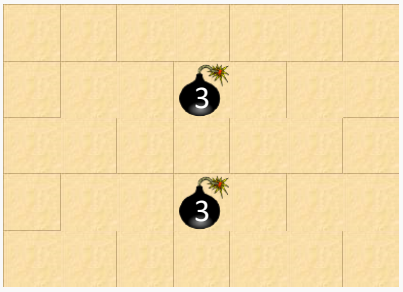
ÚLOHA 8.12

Upravte metódu **`act()`** v triede **`Bomba`** tak, aby bomba vybuchla aj keď je na rovnakej bunke s ohňom. Riešenie overte tak, že spravíte reťazovú reakciu niekoľkých bômb.

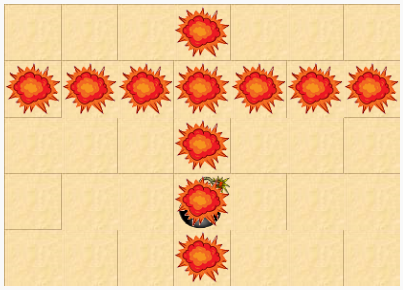
ÚLOHY NA PRECVIČOVANIE

ÚLOHA 8.A

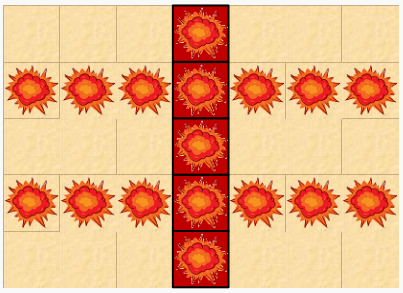
Upravte metódu **`rozsirOhen()`** tak, aby negenerovala ohne v bunkách, v ktorých sa už oheň nachádza (napr. v dôsledku reťazovej reakcie). Šírenie výbuchu sa týmto ale nezastaví. Nasledovný obrázok ukazuje príklad prekrytia viacerých inštancií triedy Ohen. Časť (a) ukazuje postavenie dvoch bômb. V časti (b) vybuchla horná bomba, čo spôsobilo umiestnenie ohňov a zároveň reťazovú reakciu dolnej bomby. Tá opäť umiestni ohne (časť (c)), ale niektoré z nich by mali byť vkladané do buniek, kde sa už ohne nachádzajú v dôsledku výbuchu prvej bomby (červené bunky). Do týchto buniek teda druhá bomba ohne negeneruje.



a



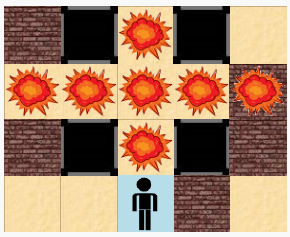
b



c

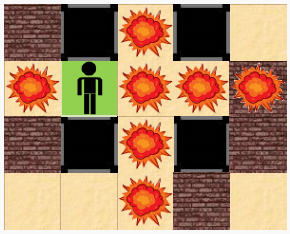
ÚLOHA 8.B

Vytvorte v aréne nový druh aktora – silové pole. Toto pole chráni hráča pred účinkami ohňa. Bunka so silovým poľom nemôže vybuchnúť ani sa cez takúto bunku nemôže šíriť oheň. Silové pole môže zaniknúť iba tak, že sa priamo naň umiestni bomba.

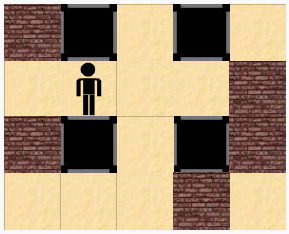


ÚLOHA 8.C

Vytvorte v aréne nový druh aktora – silový štít. Toto pole chráni hráča pred účinkami ohňa, ale iba jednorázovo. Bunka so silovým štítom nemôže vybuchnúť, ale ak bol štít zasiahnutý, odstráni sa. Oheň sa cez silový štít môže šíriť ďalej. Obrázok zobrazuje silový štít pred výbuchom (a) a po výbuchu (b) bomby so silou tri.



a



b

9 POLYMORFIZMUS

ÚLOHA 9.1

Začnime pridaním míny. Mína vybuchne práve vtedy, keď na ňu hráč stúpi. Mína tiež vždy vybuchne, keď ju zasiahne oheň (napr. z bomby, ktorá vybuchla blízko). Na svojom mieste (a iba tam) zanechá oheň. Pridajte hráčovi možnosť klásť míny (podobne, ako bomby) po stlačení klávesy (napr. **`control`** alebo **`shift`**). Podobne, ako v prípade bômb, má hráč aj obmedzený počet mín (teda ak položí všetky míny, tak ďalšiu mínu môže položiť až vtedy, ak niektorá z predtým položených mín vybuchne). Počiatočný počet mín získa hráč z parametra konštruktora. Pre evidovanie mín a reakciu na ich výbuch v triede **`Hrac`** postupujte rovnako, ako pri bombách (vytvorte zoznam mín, pridajte metódy **`vybuchlaMina()`**, **`mozePolozitMinu()`**, atď.).

ÚLOHA 9.2

Vytvorte spoločného predka pre triedy **`Bomba`** a **`Mina`** – triedu **`Vybusnina`**. Ktoré atribúty a metódy je vhodné presunúť do predka a ktoré by mali ostať v potomkoch? Upravte podľa vášho návrhu triedy.

ÚLOHA 9.3

Upravte viditeľnosť atribútu **`vlastnik`** v predkovi **`Vybusnina`** na **`protected`**.

ÚLOHA 9.4

Vytvorte bezparametrickú metódu **`vypisKtoSi()`** v triede **`Vybusnina`**, ktorá nemá návratovú hodnotu. Metóda vypíše na obrazovku text VYBUSNINA tam, kde sa aktuálne výbušnina nachádza. Vytvorte inštanciu triedy **`Mina`** a vyvolajte metódu **`vypisKtoSi()`**. Čo sa stane? Vytvorte inštanciu triedy **`Bomba`** a vyvolajte metódu **`vypisKtoSi()`**. Čo sa stane v tomto prípade?

ÚLOHA 9.5

Vytvorte v triede **Mina** metódu **vypisKtoSi()** s rovnakou hlavičkou ako v triede **Vybusnina** (teda metóda bude mať rovnaký názov, rovnaké parametre a rovnaký typ návratovej hodnoty). Metóda vypíše na obrazovku text MINA. Opäť vytvorte inštanciu triedy **Mina** a triedy **Bomba**. Odhadnite, čo sa stane, keď vyvoláte metódu test v inštancii triedy **Mina** a v inštancii triedy **Bomba**. Potom naozaj vyvolajte metódy. Zhoduje sa predpoveď s výsledkom?

ÚLOHA 9.6

Prekryte metódu **vypisKtoSi()** v triede **Bomba** tak, aby vypísala na obrazovku text BOMBA. Overte funkčnosť svojho riešenia.

ÚLOHA 9.7

Vytvorte metódy **maVybuchnut()** a **vybuch()** v triede **Vybusnina** a metódu **vybuchlaVybusnina()** v triede **Hrac** podľa popisu vyššie. Telá metód zatiaľ neimplementujte. Ak je potrebná návratová hodnota, vráťte **false**.

ÚLOHA 9.8

Napíšte telo metódy **act()** v triede **Vybusnina**.

ÚLOHA 9.9

Prekryte metódy **maVybuchnut()** a **vybuch()** v triede **Bomba**. Využite vhodný kód z jej metódy **act()**. Všimnite si, že je možné jednoducho napísať telá metód, nakoľko nie je potrebné uvažovať nad podmienkami (to spravil predok).

ÚLOHA 9.10

Definujte metódy **`maVybuchnut()`** a **`vybuch()`** v triede **`Mina`**. Využite vhodný kód z jej metódy **`act()`**. Prečo je nakoniec potrebné odstrániť metódu **`act()`**?

ÚLOHA 9.11

Upravte telo metódy **`maVybuchnut()`** v triede **`Vybusnina`** tak, aby metóda vrátila **`true`**, ak sa dotýka inštancie triedy **`Ohen`**. Upravte prekryté metódy **`maVybuchnut()`** v triede **`Bomba`** a triede **`Mina`** tak, aby využívali funkčnosť predkovej metódy.

ÚLOHA 9.12

Odstráňte z triedy **`Hrac`** atribúty **`zoznamAktivnychBomb`** a **`zoznamAktivnychMin`**. Pridajte do triedy **`Hrac`** jediný atribút **`zoznamAktivnychVybusnin`** typu **`LinkedList<Vybusnina>`**. Inicializujte ho v konštruktore a odstráňte z konštruktora inicializáciu pôvodných atribútov.

ÚLOHA 9.13

Implementujte telo metódy **`vybuchlaVybusnina()`**. Pomocou operátora **`instanceof`** zistite, či je výbušnina **`Bomba`** alebo je výbušnina **`Mina`**. Na základe jej skutočného typu zvýšte počítadlo dostupných bômb alebo počítadlo dostupných mín. Nezabudnite výbušninu odstrániť zo zoznamu aktívnych výbušnín. Nakoniec odstráňte nepotrebne metódy **`vybuchlaBomba()`** a **`vybuchlaMina()`**.

ÚLOHY NA PRECVIČOVANIE

ÚLOHA 9.A ČASOVANÉ VÝBUŠNINY

V súčasnosti má bomba atribút reprezentujúci časovač výbuchu. Upravte hierarchiu tried **`Vybusnina`** tak, aby ste vytvorili potomka **`CasovanaVybusnina`**, ktorý bude automaticky odpočítavať čas, po ktorom vybuchne.

ÚLOHA 9.B

Porovnajte metódy **`vybuch()`** v triedach **`Bomba`** a **`Mina`**. Identifikujte rovnaký príkaz. Upravte metódu **`vybuch()`** predka **`Vybusnina`** tak, aby túto spoločnú funkčnosť poskytoval on. Potom upravte metódy potomkov. Ktorá metóda sa stane zbytočná?

ÚLOHA 9.C

Pridajte možnosť, aby hráči mohli klásť dynamit. Všetky inštancie triedy **`Dynamit`** vybuchnú naraz – keď to určí hráč stlačením klávesu. Kláves, ktorým sa ovláda výbuch dynamitov, je parametrom konštruktora triedy **`Hrac`**. Ak hráč zomrie, všetky jeho dynamity okamžite vybuchnú.

10 NÁHODNÉ ČÍSLA

ÚLOHA 10.1

Zamyslite sa, čo je to náhoda, ako vieme získať nejaký náhodný výsledok pokusu, aké náhodné javy pozorujeme vo svete okolo nás.

ÚLOHA 10.2

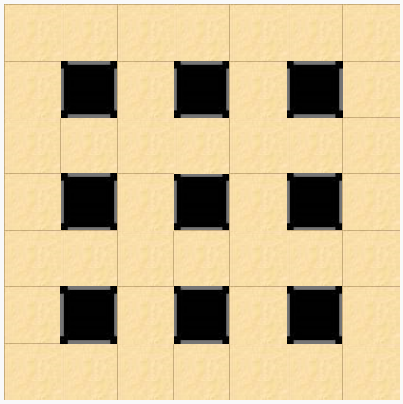
Vezmime si klasickú hraciu kocku so šiestimi stranami. Vedeli by sme pomocou nej vygenerovať náhodnú pozíciu na šachovnici s rozmermi 6x6 políčok? A čo šachovnica s rozmermi 3x3 políčka? Ako by sa zmenil spôsob generovania, ak by sme použili mincu? Navrhnite takéto algoritmy generovania polohy.

ÚLOHA 10.3

Pomocou algoritmu z predošlej úlohy a s pomocou kocky generujte náhodné pozície na šachovnici. Svoje výsledky si do šachovnice zaznačte. Dá sa pozorovať nejaká pravidelnosť výsledkov?

ÚLOHA 10.4

Pripravte si arénu. Vytvorte potomka triedy **`Arena`**, ktorý nazvete napr. **`NahodnaArena`**. V konštruktore nastavte vhodnú veľkosť sveta. Odporúčame pravidelné rozloženie stien s jedným voľným poľom medzi stenami tak, ako je ukázané na nasledujúcom obrázku:



ÚLOHA 10.5

Pridajte do triedy **NahodnaArena** referenčný atribút triedy **Random**. Nezabudnite, že trieda **Random** sa nachádza v balíčku **java.util.Random**. Kocku inicializujte v konštruktore.

ÚLOHA 10.6

Pridajte do triedy **NahodnaArena** súkromnú metódu **jeBunkaVolna()**, ktorá preberie dva parametre – stĺpec a riadok. Metóda vráti **true**, ak je bunka vo svete voľná (neobsahuje žiadneho aktora), inak vráti **false**.

ÚLOHA 10.7

Upravte konštruktor triedy **NahodnaArena** tak, aby náhodne vygeneroval múry do tretiny všetkých buniek v aréne.

ÚLOHA 10.8

Presuňte metódy **vytvorNahodnyMur()**, **jeBunkaVolna()** a atribút (vrátane jeho inicializácie v konštruktore) **kocka** do predka **Arena**. Nezabudnite presunúť aj riadky **import**.

ÚLOHA 10.9

Pridajte do metódy **vytvorNahodnyMur()** parameter typu **Actor** – toto bude aktor, ktorého budeme vkladať na náhodné súradnice. Upravte názov metódy (napr. **vlozNaNahodneSuradnice()**) a upravte volanie metódy z triedy **NahodnaArena**.

ÚLOHA 10.10

Vytvorte triedu **Bonus** ako potomka triedy **Actor**. Vytvorte dvoch potomkov triedy **Bonus** – triedu **BonusOhen** a triedu **BonusBomba**. Nastavte triedam vhodné obrázky.

ÚLOHA 10.11

Upravte kód v metóde **`act()`** triedy **`Mur`**. S pravdepodobnosťou 10 % vygenerujte na jej mieste po zničení bonus oheň, s pravdepodobnosťou 10 % vygenerujte na jej mieste bonus bomba. V 80 % prípadoch sa po zničení nevygeneruje nič.

ÚLOHA 10.12

Pripravte predka **`Bonus`**. Vytvorte v ňom chránenú virtuálnu metódu bez návratovej hodnoty **`aplikujSa()`**, ktorá preberie jediný parameter typu **`Hrac`**. Predok túto metódu môže nechať prázdnu. Následne definujte činnosť v metóde **`act()`** tak, aby najskôr zistila, či na bonus stúpil hráč (metóda **`(Hrac)this.getOneIntersectingObject(Hrac.class)`**) a ak áno, tak sa naňho aplikuje (vyvolaním virtuálnej metódy) a nakoniec sa bonus odstráni zo sveta.

ÚLOHA 10.13

Pridajte do triedy **`Hrac`** verejnú bezparametrickú metódu **`zvysPocetBomb()`**, ktorá nemá návratovú hodnotu, a ktorá zvýši počet bômb, ktoré môže hráč položiť o jedna. Potom prekryte metódu **`aplikujSa()`** v triede **`BonusBomba`**. Aplikácia bomby znamená navýšenie počtu bômb hráčovi o jedna (vyvolanie metódy **`zvysPocetBomb()`**).

ÚLOHA 10.14

Pridajte do triedy **`Hrac`** verejnú bezparametrickú metódu **`zvysSiluBomb()`**, ktorá nemá návratovú hodnotu, a ktorá zvýši hodnotu atribútu **`silaBomb`** o jedna. Potom prekryte metódu **`aplikujSa()`** v triede **`BonusOhen`**, a zvýšte v nej silu bômb hráčovi o jedna.

ÚLOHY NA PRECVIČOVANIE

ÚLOHA 10.A

Vytvorte v triede **`Arena`** metódu **`vytvorNahodneSteny()`**, ktorá v parametri preberie počet stien a takýto počet stien vygeneruje a vloží na náhodné voľné miesta do sveta.

ÚLOHA 10.B

Upravte konštruktor triedy **`Ohen`** tak, aby si pri vzniku inštancia zvolila náhodný obrázok ohňa. Všetky ohne majú rovnakú pravdepodobnosť výskytu. Pre zmenu obrázku použite metódu **`setImage()`** triedy **`Actor`**, ktorá v parametri preberá reťazec s názvom súboru obrázka (obrázky musia byť umiestnené v priečinku **`images`** v koreňovom priečinku projektu).

ÚLOHA 10.C

Pridajte bonusy, ktoré ovplyvnia hráča negatívne, teda znížia mu počet bômb a znížia mu dosah bômb.

ÚLOHA 10.D

Pridajte bonus teleport. Nech sa tento bonus vytvorí s pravdepodobnosťou 5 %. Teleport presunie hráča na náhodné pole v aréne.