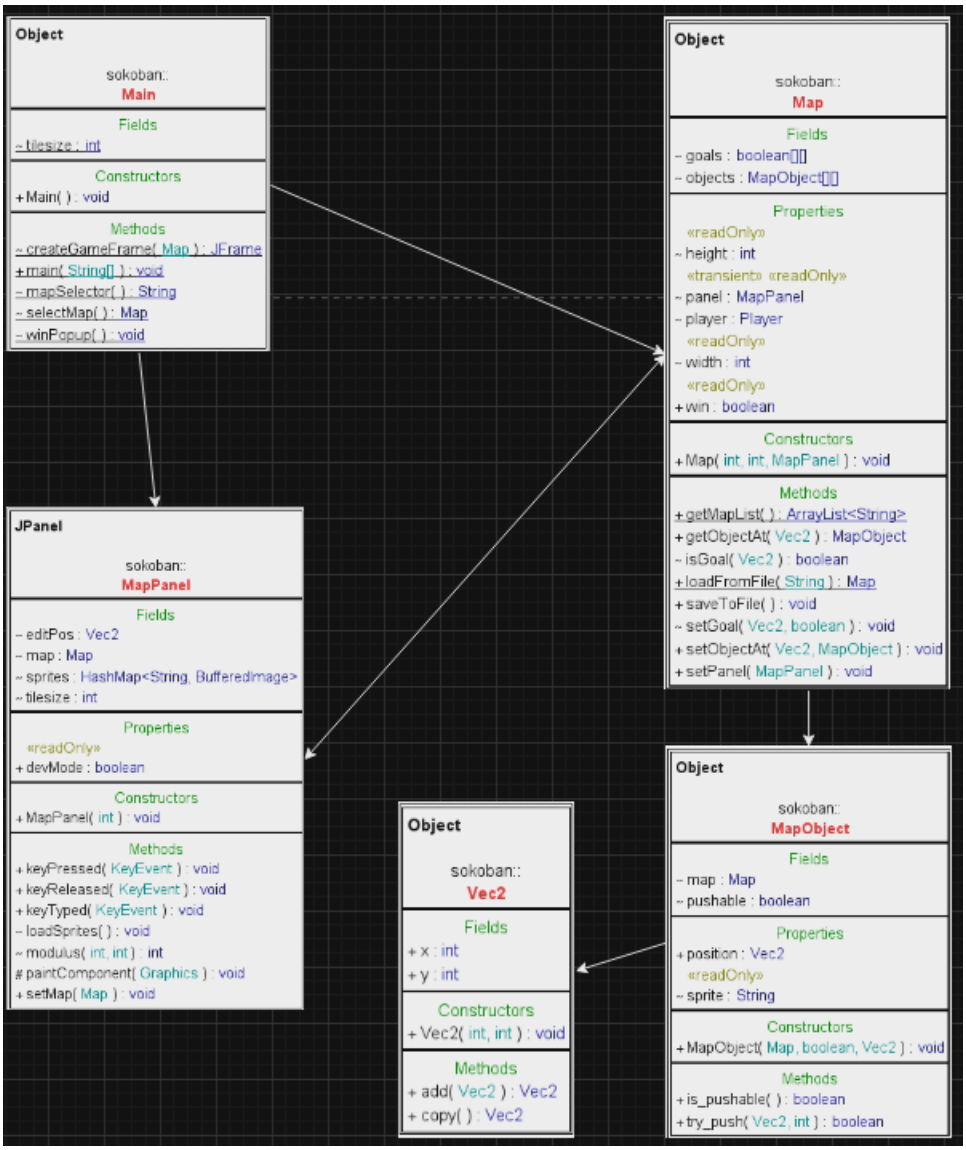
NHF 3 dokumentáció – “Sokoban”

Dúcz Ákos (GC1RTE)

1. AZ OSZTÁLYOK MŰKÖDÉSE



Az osztálydiagrammról a MapObject-et extendelő osztályok (tehát a többféle doboz és a player) le lettek hagyva, mivel lényegében ugyanezekkel a tulajdonságokkal és kapcsolatokkal rendelkeznek, de nem férnének el a képen.

1. Main osztály:

static String mapSelector() : Felhoz egy popup ablakot, ami a játékostól egy pálya választását kéri, és ennek a nevét visszaadja. Ez a játék főmenüje.

static Map selectMap() : Megkéri a játékost hogy válasszon egy pályát, majd ezt be is tölti és visszaadja. Ha a játékos új pályát akar létrehozni, akkor megkérdezi a méretét, és ekkora pályát hoz létre.

static int tilesize : a játék által kirajzolt csempék mérete.

static JFrame createGameFrame(Map map) : Egy megadott pályához létrehoz egy ablakot, amely ezt kirajzolja, és kezeli a bemeneket.

static void winPopup() : Kiírja a "nyertél" üzenetet egy popup ablakban.

public static void main(String[] args) : main metódus, elindítja a főmenüt, betölti a választott pályát, majd vár amíg a játék véget nem ér. ekkor kiírja a "nyertél" üzenetet, majd újból a főmenübe léptet.

2. Map osztály

MapObject[][] objects : a pályán található objektumok mezőnként

boolean[][] goals (get/set): minden mezőre megadja, hogy cél-e az adott mező

int width (get/set): a pálya szélessége

int height (get/set): a pálya magassága

transient MapPanel panel (get/set): a pályát kirajzoló MapPanel objektum

Player player (get/set): a játékos objektum a pályán

public boolean isWin() : megadja hogy vége-e a játéknak a dobozok és célok elhelyezkedése alapján. Ha egy cél sincs, akkor azt nem veszi nyerésnek.

public Map(int width, int height, MapPanel panel) : létrehoz egy pályát a megadott paraméterekkel

public MapObject getObjectAt(Vec2 pos) : visszaadja az adott pozícióban álló objektumot. (null ha nincs ott semmi). Ha nincs a pályán a megadott koordináta, akkor hibát dob.

public void setObjectAt(Vec2 pos, MapObject obj) : beállítja hogy az “obj” objektum a megadott pozíción legyen a pályán. (a pálya nézőpontjából, de ezt az objektumnak is tudnia kell)

public void saveToFile() : megkéri a felhasználót, hogy nevezze el a pályát, majd elmenti szerializáció segítségével a megfelelő fájlba.

static public Map loadFromFile(String name) : megadott fájlnév alapján a “maps” mappából tölt be egy pályát.

static public ArrayList<String> getMapList() : A "maps" mappában lévő fájlok listája, ezekből lehet választani a főmenüben. (és "új pálya" opció).

3. MapObject osztály (és leszármazottak)

(az “objektum” szó itt főként MapObject-re vonatkozik, de ez kontextusból egyértelmű)

boolean pushable (get): el lehet-e tolni ezt az objektumot más objektumok által?

Map map : az objektumot tartalmazó pálya

Vec2 position (get): a pozíciónk a pályán belül

Public MapObject(Map host_map, boolean can_push, Vec2 pos) : létrehoz egy MapObject-et a mgadott praméterekkel, és el is helyezi a pályán a megadott koordinátára.

public void setPosition(Vec2 newPos) : átállítja az objektum pozícióját a pályán, és törli a hivatkozást a régebbi pozícióról is.

public boolean try_push(Vec2 direction, int depth) : megpróbálja eltolni az objektumot a pályán a megadott irányban. (normális esetben ez egy x vagy y tengelyű egységvektor) Visszaadja, hogy sikeres-e az akció. Ez azzal is járhat, hogy közben egy másik objektumot kell eltolnia ennek, akkor a következő tolás sikerességétől is függ a jelenlegi. A depth változó a játékostól érkező eltolás esetén 0, egyébként eggyel nő minden továbbadott eltolás esetén. Ez a nehéz doboz implementációjánál hasznos.

String getSprite() : melyik sprite-ot rajzoljuk ki ehhez az objektumhoz?

A legtöbb MapObject-ből származtatott osztály csak minimális változtatásokat alkalmaz ezekben a függvényekben. Az említésre méltó esetek:

A HeavyBox try_push függvénye csak depth = 1 esetben engedi eltolni magát, és nem is próbál meg továbbtolni más MapObjecteket.

Az IceBox eltoláskor addig csúszik amíg akadályba nem ütközik. (extra slide függvény)

A Key try_push függvénye ellenőrzi hogy Door-nak tolódik-e neki, és ha igen, akkor törli mindkét objektumot a pályáról.

A Player osztály implementálja a KeyListener interfészt, és a **public void keyPressed(KeyEvent e)** függvény segítségével hallgat a játékos gombnyomásaira (és a megadott irányba tolja a játékost). Minden gombnyomás után notify-olja a main-ben lévő ciklust, hogy ellenőrizze a nyerési feltételeket, és újrarajzoltatja a pályát is.

4. MapPanel oszály

int tilesize : a kirajzolt csempék mérete

HashMap<String, BufferedImage> sprites : a betöltött sprite-ok név alapján, így elég egyszer betölteni őket a memóriába, nem kell minden MapObject-hez külön.

void loadSprites() : betölti a játék sprite-jait név alapján a sprites mappából.

Map map (set): a kirajzolandó pálya

public MapPanel(int tilesize) : MapPanel konstruktor

protected void paintComponent(Graphics g) : kirajzolja a pálya csempéit a betöltött sprite-ok és a Map getObjectAt függvénye segítségével

Vec2 editPos : a pályaszerkesztő kurzor pozíciója

boolean devMode (get): pályaszerkesztő módban vagyunk-e?

public void keyPressed(KeyEvent e) : kezeli a pályaszerkesztőhöz tartozó bemeneteket.

5. Vec2 osztály

public int x : x koordináta

public int y : y koordináta

public Vec2(int x, int y) : konstruktor

public Vec2 add(Vec2 v) : összead két vektort

public Vec2 copy() : lemásolja a vektort

2. A TESZTEK LEÍRÁSA

A tesztek JUNIT 5-öt használnak (standalone konzolos verzió)

A SokobanTest osztályban vannak implementálva.

Szükséges futtatásukhoz egy előre létrehozott 9x9-es “demo” pálya, mely az 5. sorában tartalmazza a játékban lévő összes féle dobozt, és egy célt. (5.sor: üres, fal, könnyű doboz, nehéz doboz, jég, ajtó, kulcs, cél, üres)

public void setUp() : betöti a “demo” pályát, és megtalálja benne a tesztelendő dobozokat

A tesztek:

public void map_getObjectTest() : megnézi hogy a getObjectAt függvény mefelelő értékeket ad-e vissza, illetve dob-e hibát ha rossz koordinátákat adunk meg.

public void map_createObjectTest() : a setObjectAt függvényt teszteli, és azt hogy a létrehozott MapObject-ek jó helyre kerülnek-e.

public void map_winTest() : a Map osztály isWin függvényét teszteli.

Az alábbi tesztek az egyes dobozfélék eltolásainak helyességét vizsgálják:

public void wall_pushTest()

public void box_pushTest()

public void hbox_pushTest()

public void icebox_pushTest()

public void key_and_door_test()

public void mapPanelTest() : létrehoz egy MapPanelt a meglévő pályához. Főként azt teszteli hogy sikeresen betöltenek-e a sprite-ok (különben exception-t dobna).

public void mapListTest() : a Map osztály getMapList függvényét teszteli.

3. FELHASZNÁLÓI KÉZIKÖNYV

A program megnyitásakor egy főmenü felajánl lehetőségeket az elmentett pályák közül, illetve egy “<create new map>” lehetőséget, mellyel új pályát hozhatunk létre. Amennyiben az utóbbit választjuk, megadhatunk egy pályaméretet (nem kötelező, alapértelmezett a 9x9), és így egy üres pályát kapunk. Egyébként betöltődik a választott pálya.

A játékos a WASD billentyűk lenyomásával mozgatjuk. Célunk, hogy minden piros pöttyel jelzett mezőre egy dobozt helyezzünk (de nem feltétlenül kell minden dobozt piros pöttyre helyezni).

Ha ez sikerül, akkor erről egy felugró ablak értesít, majd újból pályát választhatunk.

A shift billentyű lenyomásával beléphetünk a pályaszerkesztő módba. Ekkor egy lila kurzor jelenik meg, melyet a nyilakkal mozgathatunk. A 0 billentyűvel törölhetünk dobozokat, az 1...6 billentyűkkel lehelyezhetjük őket, a 7-es gombbal pedig egy célt kapcsolhatunk ki/be.

Pályaszerkesztő módban az “O” gomb megnyomásával menthetjük el a pályánkat. Kilépni ismét a shift-tel tudunk.

Balról jobbra: a játékos, fal, könnyű doboz, nehéz doboz, jég, ajtó, kulcs, cél, üres mező

Fal: nem mozdítható semmilyen módon

Könnyű doboz: tolható, akár több is egymás mögött

Nehéz doboz: csak egymagában tolható

Jég: tolható, de addig csúszik amíg akadályba nem ütközik. Érdemes kipróbálni mi történik ha egymás mögött két jeget tolunk.

Ajtó: nem mozdítható, de ha kulcsot tolunk neki eltűnik.

Kulcs: ha ajtónak toljuk, eltűnik.

Futtatáshoz/fordításhoz/teszteléshez rendelkezésre állnak batch scriptek.