Sokoban – Dokumentáció

Use case-ek leírása:

Pálya választás	A játékos a legördülő menüben kijelölheti pályát, amin játszani akar.
Segítség használata	A „Segítség” gombra kattintva a játékos kap egy rövid leírást a játékról illetve a gombokról.
Játék elkezdése	A „Start” gomb megnyomásával indul a játék a kiválasztott pályán.
Soko irányítása	A játékos a nyilakkal vagy kurzor billentyűkkel képes a Sokot irányítani.
Pálya újra kezdése	Az „R” gomb megnyomásával a pálya alap helyzetbe kerül.
Doboz lökése	Ha a játékos egy pozícióba kerülne egy dobozzal, akkor a lenyomott billentyű irányába el löki azt. Viszont egy irányba egyszerre csak egy doboz képes lökni.
Kilépés	Az alkalmazás bezárása.

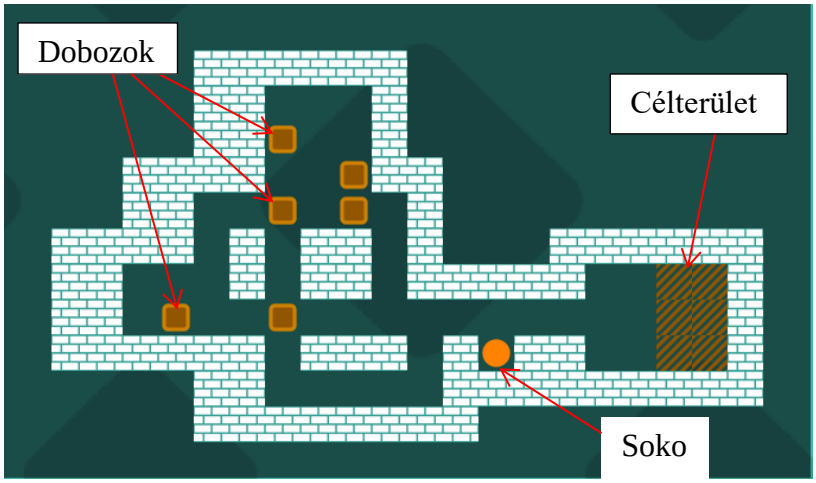
Használati útmutató:

Játék leírás:

A Sokoban (magyarul raktárosnak is hívják) egy olyan logikai játék melyben a játékosnak ládákat kell mozgatnia a megadott helyekre. A játék gyakorlatilag egy „helyiség” játszódik, melyben X darab ládát kell X darab pozícióba elhelyezni a pálya teljesítéséhez. Egy adott pályán a dobozokat a megadott pontokba kell tologatni. A játék nehézségét az jelenti, hogy dobozokat nem lehet húzni és egyszerre csak egy dobozt tud a játékos mozgatni. Illetve a pálya kialakításától is függ, mert ha egy dobozt rossz helyre tol, kezdheti újra a pályát. Jó pálya kialakításnál minden doboz csak egy helyre kerülhet ahhoz, hogy teljesítsük a pályát. (A mellékelt kép esetén a kis emberkének a dobozokat a piros pontokra kell tolnia.) Ebben a konkrét esetben 2 darab pálya lesz elérhető. Az egyes pályák sikeres teljesítését az ablakon megjelenő „Teljesítve” felirat fogja jelezni.

Irányítás:

A programban a legtöbb helyen egér segítségével lehet navigálni. Ez igaz a menüre és a segítségre is. A pályát is ennek segítségével lehet kiválasztani.
A játékot a start megnyomásával lehet elkezdeni. A tényleges játékban a „Soko”-t a nyilakkal vagy más néven a kurzor billentyűkkel lehetséges mozgatni. Itt működik még az „R” billentyű, mely segítségével a pálya alaphelyzetbe állítható.
Illetve a teljes alkalmazásban az „F” billentyűvel lehet ki/be kapcsolni a teljes képernyős módot.



Osztályok leírása:

Map:

Leírás: A pályákat beolvassa és egy Stringként tárolja.

Metódusai:

void initMap()	Beolvassa a megfelelő pályát és eltárolja.(map1.txt és map2.txt)
void Map(String s)	A Map konstruktora, beállítja a tárolt String értékét s-re és meghívja a beolvasáshoz készült függvényt.
String getLevel()	Visszaadja a tárolt pályát Stringként.
void setLevel(String s)	Beállítja a tárolt Stringet s-re. Erre azért van szükség mert először a menüben lévő kiválasztás eredményét tároljuk, hogy tudjuk melyik pályát kell beolvasni.

Element:

Leírás: A pálya elemeket általánosítja, eltárolja pozíciójukat és a képeiket.

Metódusai:

Element(int x, int y)	Az Element konstruktora, beállítja az elem pozícióját.
Image getImage()	Visszaadja a tárolt image értékét.
int getX()	Visszaadja az Element x pozícióját.
void setX(int numX)	Beállítja az Element x pozícióját numX-re.
int getY()	Visszaadja az Element y pozícióját.
void setY(int numY)	Beállítja az Element x pozícióját numY-ra.
void setImage(Image i)	Beállítja az Element képét i-re.
boolean isLeftColl(Element elem)	Megnézi, hogy az adott Element balról ütközik-e az elem Elementel.
boolean isRightColl(Element elem)	Megnézi, hogy az adott Element jobbról ütközik-e

	az elem Elementel.
boolean isTopColl(Element E)	Megnézi, hogy az adott Element fentről ütközik-e az elem Elementel.
boolean isBottomColl(Element e)	Megnézi, hogy az adott Element lentről ütközik-e az elem Elementel.

Box:

Leírása: Az Element leszármazottja a dobozokat reprezentálja.

Metódusai:

Box(int x, int y)	A Box konstruktora, ami ősének a konstruktorát hívja meg. Azért van rá szükség, mert a többitől eltérő képe van.
void initBox()	Beolvassa és beállítja a Box képét.
void move(int X, int Y)	Megváltoztatja a Box pozícióját X-re illetve Y-ra.

Area:

Leírás: Az Element leszármazottja a cél darabokat reprezentálja.

Metódusai:

Area(int x, int y)	Az Area konstruktora, ami ősének a konstruktorát hívja meg. Azért van rá szükség, mert a többitől eltérő képe van.
void initArea()	Beolvassa és beállítja az Area képét.

Wall:

Leírás: Az Element leszármazottja a fal darabokat reprezentálja.

Metódusai:

Wall(int x, int y)	A Wall konstruktora, ami ősének a konstruktorát hívja meg. Azért van rá szükség, mert a többitől eltérő képe van.
void initWall()	Beolvassa és beállítja a Wall képét.

Player:

Leírás: Az Element leszármazottja a Sokot azaz a játékost reprezentálja.

Metódusai:

Player(int x, int y)	A Player konstruktora, ami ősének a konstruktorát hívja meg. Azért van rá szükség, mert a többitől eltérő képe van.
void initPla()	Beolvassa és beállítja a Player képét.
void move(int X, int Y)	Megváltoztatja a Player pozícióját X-re illetve Y-ra.

HelpJPanel:

Leírás: A JPanel leszármazottja mely a program Segítség részét kezeli. Tárolja a feliratait, gombját, hátterét.

Metódusai:

void initHelp()	Beolvassa és eltárolja a háttérképet, a gomb képét illetve a feliratot. Beállítja a gomb (hogy a gomb és a képe egymáson legyen jól igazítva) és a felirat tulajdonságait. Hozzáadja panelhez a gombot és szöveget a megadott elrendezésben.
void HelpTextRead()	Beolvassa a feliratot egy String tömbbe a help.txt-ből.
HelpJPanel()	A konstruktor, meghívja az initHelp() „beállító” függvényt.
void paintComponents(Graphics g)	Meghívja a JPanel paintComponents-ét hiszen csak a háttér kirajzolásához van erre szükség.

MenuJPanel:

Leírás: A JPanel leszármazottja mely a program Menü részét kezeli. Tárolja a feliratait, gombjait, hátterét illetve a pálya választó legördülő menüt.

Metódusai:

void initMenu()	Beolvassa és eltárolja a háttérképet és a gombok képét. Beállítja a gombok (hogy a gomb és a képe egymáson legyen jól igazítva) és a felirat tulajdonságait. Hozzáadja panelhez a gombokat, szöveget és legördülő menüt a megadott elrendezésben.
MenuJPanel()	A konstruktor, meghívja az initMenu() „beállító” függvényt.
void paintComponents(Graphics g)	Meghívja a JPanel paintComponents-ét hiszen csak a háttér kirajzolásához van erre szükség

StartJPanel:

Leírás: A JPanel leszármazottja mely a program Játék részét kezeli. Tárolja az egyes pálya elemeket, a vissza gombot, a hátteret, a Játékost és az aktuális pályát. Létrehozza a pályát a beolvasott szöveges fájl segítségével a pályaelemekből. Biztosítja a mozgást és a restartot, kezeli az ütközéseket.

Metódusai:

void setLevel(Map m)	Beállítja a pályát m-re.
sethasMap()	Beállítja false-ra azt az értéket ami jelzi, hogy van-e pálya. Azért van rá szükség, hogy frissüljön a pálya, de mégis ha vissza megyünk a menübe a másik pályát

	is be tudja tölteni.
StartJPanel()	A konstruktor, az initWorld() „beállító” függvényt. Beállítja a billentyűkiosztást a mozgáshoz és a restart-hoz.
UpKey osztály	Az AbstrackAction leszármazottja, melyben az ősosztály actionPerfomed függvényét definiáljuk felül, ami a felfelé való mozgáshoz kell. Az ütközés esetén visszatér, egyébként mozgatja felfele a játékost és újrarajzolja a képet.
DownKey osztály	Az AbstrackAction leszármazottja, melyben az ősosztály actionPerfomed függvényét definiáljuk felül, ami a lefelé való mozgáshoz kell. Az ütközés esetén visszatér, egyébként mozgatja lefele a játékost és újrarajzolja a képet.
RightKey osztály	Az AbstrackAction leszármazottja, melyben az ősosztály actionPerfomed függvényét definiáljuk felül, ami a jobbra való mozgáshoz kell. Az ütközés esetén visszatér, egyébként mozgatja jobbra a játékost és újrarajzolja a képet.
LeftKey osztály	Az AbstrackAction leszármazottja, melyben az ősosztály actionPerfomed függvényét definiáljuk felül, ami a balra való mozgáshoz kell. Az ütközés esetén visszatér, egyébként mozgatja balra a játékost és újrarajzolja a képet.
RestartKey osztály	Az AbstrackAction leszármazottja, melyben az ősosztály actionPerfomed függvényét definiáljuk felül, ami a pálya újrakezdéséhez kell. Meghívja az erre megírt függvényt és újrarajzolja a képet.
void initWorld()	Beolvassa és eltárolja a háttérképet és a gomb képét. Beállítja a gomb (hogy a gomb és a képe egymáson legyen jól igazítva) tulajdonságait. Hozzáadja panelhez a gombot a megadott elrendezésben.
void initMap()	Inicializálja a falakat, a dobozokat és a célterületeket. A Map level Stringjében az egyes karaktereket megfelelteti a pályaelemeknek és hozzáadja a tárolójukhoz. Egyébként üres helyet hagy.
void buildMap(Graphics g)	Ellenőrzi, hogy van-e a már pálya, ha nincs, akkor beolvassa a megfelelőt.

	Létrehozza a pályát (world) , ehhez hozzáadja az egyes elemeket (falakat, célterületeket, dobozokat és a játékost). Ezután egy for ciklus segítségével a végig megy az a pálya elemeken és megrajzolja az adott darabot a képe alapján a megadott helyre. Illetve a pálya teljesítése esetén a teljesítve feliratot is megjeleníti.
void paintComponent(Graphics g)	Meghívja a JPanel paintComponents-ét hiszen csak a háttér és a pályaelemek (buildMap(g)) kirajzolásához van erre szükség.
boolean checkWallColl(Element elem, int type)	Az adott Element (elem) fallal való ütközését adja mind a négy irány ellenőrizve. Ha fenn áll az ütközés igazzal tér vissza ha nem akkor hamissal.
boolean checkBoxColl(int type)	Külön kezeli a négyféle ütközést (felülről, alulról, jobbról, balról). Az egyes esetekben végig megy a dobozokon egy ciklussal, mindegyiknél ellenőrzi, hogy igaz-e az adott ütközés, ha nem akkor hamissal visszatér. Egyébként egy másik ciklus indul szintén a dobozokon (az első ciklusban lévő kivételével) egy másik dobozzal való ütközés miatt (hiszen ilyenkor nem lehet elmozdítani az első ciklusban lévő dobozt). Ha van ütközés egy másik dobozzal visszatér igazzal. Ha fallal ütközik, akkor is igazzal tér vissza. Ha előző kettő feltétel egyik-e se igaz mozgatja az első dobozt a megfelelő irányba illetve ellenőrzi, hogy vége van-e a játéknak.
void isCompleted()	A dobozok és a célterületek pozícióját veti össze. Ha dobozok száma megegyezik a jó helyen lévő dobozok számával, akkor az isCompleted értékét igazra állítja és újrarajzolja a képet.
void restartLevel()	Törli a pálya elemeket, kivéve a játékost. A hasMap értékét hamisra állítja illetve, ha az isCompleted igaz, akkor hamisra állítja.

Sokoban:

Leírás: A gombokhoz készült ActionListenereket és a fullscreenhez készült key bindingot tartalmazza. Magát az alkalmazás futtatja.

Metódusai:

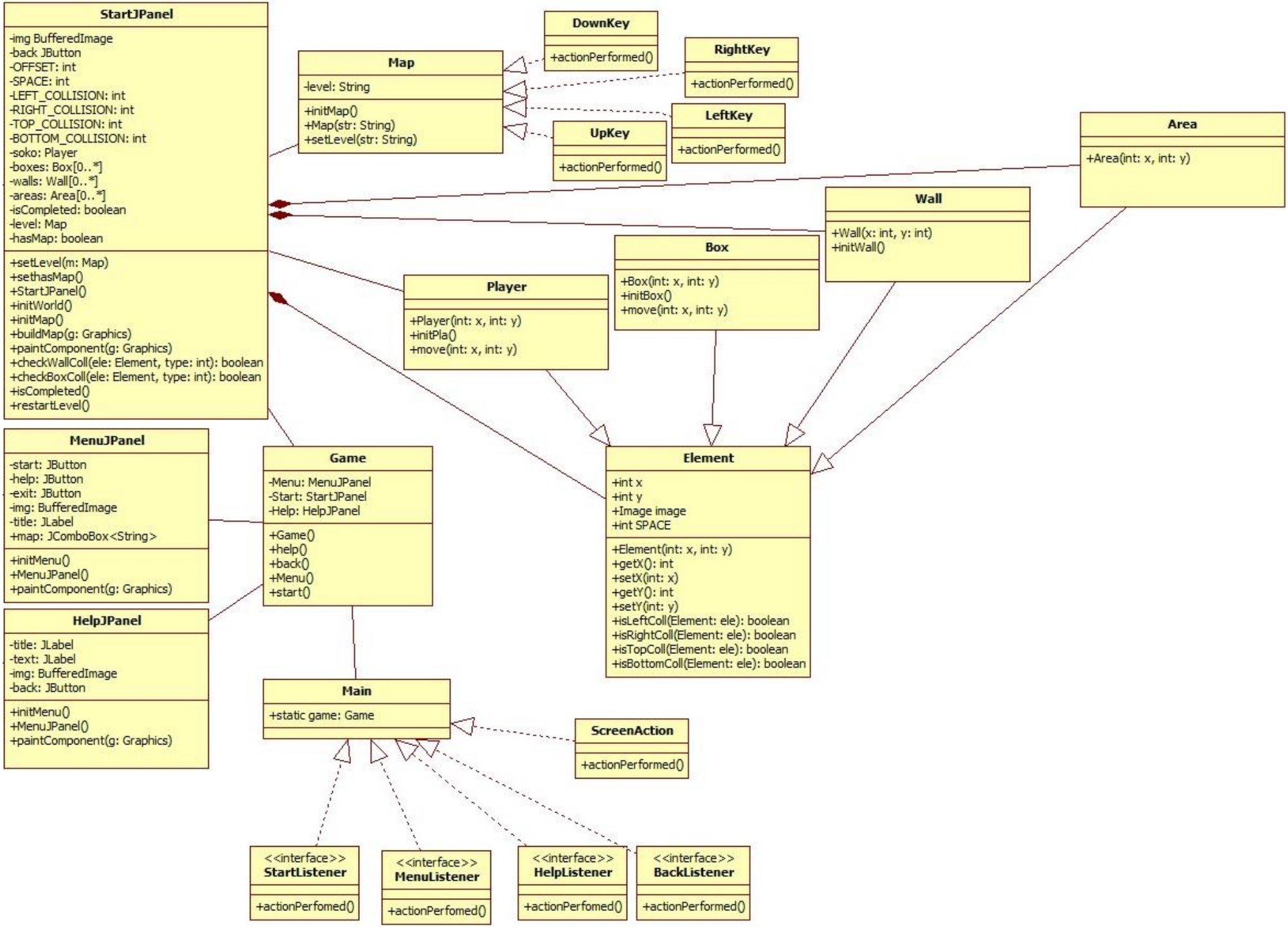
HelpListener osztály	A segítség panel elérését biztosító ActionListener implementáció.
MenuListener osztály	A játékban a menübe való visszalépést biztosító ActionListener implementáció.
BackListener osztály	A segítségben a menübe való visszalépést biztosító ActionListener implementáció.
StartListener osztály	A játék elkezdését biztosító ActionListener implementáció.
CloseListener osztály	A kilépést biztosító ActionListener implementáció.
ScreenAction osztály	A teljes képernyő ki/be kapcsolását biztosító AbstractAction-ból származó osztály.
void main(String[] args)	Maga az alkalmazás.

Game:

Leírás: A JFrame leszármazottja, tárolja az egyes Paneleket, kezeli az átmenetek a panelek között.

Metódusai:

Game()	A Game konstruktora, inicializálja a paneleket, ezekre beállítja az teljes képernyő billentyűt (F). Hozzáadja a framehez a kezdő panelt.
void help()	A menu panel helyett a help panelt adja hozzá a framehez.
void back()	A help panel helyett a menu panelt adja hozzá a framehez.
void Menu()	A start panel helyett a menu panelt adja hozzá a framehez.
void start()	A menü panel helyett a start/ panelt adja hozzá a framehez.


StartJPanel
-img BufferedImage
-back JButton
-OFFSET: int
-SPACE: int
-LEFT_COLLISION: int
-RIGHT_COLLISION: int
-TOP_COLLISION: int
-BOTTOM_COLLISION: int
-soko: Player
-boxes: Box[0..*]
-walls: Wall[0..*]
-areas: Area[0..*]
-isCompleted: boolean
-level: Map
-hasMap: boolean
+setLevel(m: Map)
+sethasMap()
+StartJPanel()
+initWorld()
+initMap()
+buildMap(g: Graphics)
+paintComponent(g: Graphics)
+checkWallColl(ele: Element, type: int): boolean
+checkBoxColl(ele: Element, type: int): boolean
+isCompleted()
+restartLevel()
DownKey
+actionPerformed()
RightKey
+actionPerformed()
Map
-level: String
+initMap()
+Map(str: String)
+setLevel(str: String)
LeftKey
+actionPerformed()
UpKey
+actionPerformed()
Area
+Area(int: x, int: y)
Wall
+Wall(x: int, y: int)
+initWall()
Box
+Box(int: x, int: y)
+initBox()
+move(int: x, int: y)
Player
+Player(int: x, int: y)
+initPla()
+move(int: x, int: y)
MenuJPanel
-start: JButton
-help: JButton
-exit: JButton
-img: BufferedImage
-title: JLabel
+map: JComboBox<String>
+initMenu()
+MenuJPanel()
+paintComponent(g: Graphics)
Game
-Menu: MenuJPanel
-Start: StartJPanel
-Help: HelpJPanel
+Game()
+help()
+back()
+Menu()
+start()
Element
+int x
+int y
+Image image
+int SPACE
+Element(int: x, int: y)
+getX(): int
+setX(int: x)
+getY(): int
+setY(int: y)
+isLeftColl(Element: ele): boolean
+isRightColl(Element: ele): boolean
+isTopColl(Element: ele): boolean
+isBottomColl(Element: ele): boolean
HelpJPanel
-title: JLabel
-text: JLabel
-img: BufferedImage
-back: JButton
+initMenu()
+MenuJPanel()
+paintComponent(g: Graphics)
Main
+static game: Game
ScreenAction
+actionPerformed()
<<interface>>
StartListener
+actionPerfomed()
<<interface>>
MenuListener
+actionPerfomed()
<<interface>>
HelpListener
+actionPerfomed()
<<interface>>
BackListener
+actionPerformed()