Gyűjtögető

Programozás alapjai 3 házi feladat

Tarcza Lídia

A program célja

Egy gyűjtögető játék elkészítése. A játék kezdetekor a játékos három élettel kezd, valamint a pontszáma nulla. Értelemszerűen a játék során veszthet az életéből, de akár meg is növelheti azt. A játékos a pálya alján tud mozogni jobbra és balra, eközben az égből különböző tárgyak hullanak, amelyeket elkaphat. A játék előrehaladtával egyszerre egyre több tárgy jelenik meg, és gyorsaságuk is növekedik. A tárgyakat két nagyobb csoportra oszthatjuk: az egyikkel pontokat lehet szerezni, a másikkal életet lehet veszteni/nyerni.

A pontszerző jellegű tárgyak különböző csoportokba sorólhatóak, és minden csoport más-más értékkel rendelkezik. Minél nagyobb értékű egy tárgy annál kisebb gyakorisággal jelenik meg. Ha a játékos elkap egy ilyen tárgyat, akkor az értéke hozzáadódik a játékos összpontszámához. A játék célja, minél nagyobb pontszámot elérése, mielőtt a játék véget ér.

A tárgyak másik csoportjával életet lehet veszteni vagy nyerni, amennyiben a játékos elkap egy ilyen tárgyat a fajtájától függően változik meg az életének az értéke. Az ilyen típusú tárgyak a játék elején kevésszer jelennek meg, de minél tovább tart a játék annál nagyobb gyakorisággal kezdenek el hullani az égből. Ha a játékos elveszti az összes életét, a játék véget ér.

A program megvalósítása

Osztályok leírása

Position

Minden játékbeli elemnek, azaz a játékosnak és a gyűjthető tárgyaknak is van saját koordinátája. Az aktuális koordinátához tartozó x és y értékét tárolja a Position osztály. Az osztály két tagváltozójában tároljuk az x és az y értékét.

Az osztály kétparaméteres konstruktorában meg kell adni az aktuális x és y értékét. Az osztály a gettereken és a settereken kívül más metódust nem valósít meg.

GameObject

Absztrakt osztály. Ebből az osztály származtatjuk a játékos osztályát és a gyűjthető tárgyak osztályait. Mivel mindegyiküknek van pozíciója, ezért van egy Position típusú tagváltozója, ami az aktuális pozíciót tárolja, illetve egy Color típusú tagváltozója, ahol az adott tárgy alap színét lehet eltárolni.

Az osztály a gettereken és a settereken kívül az alábbi metódusokat írja elő:

- public abstract void draw(Graphics g): Absztrakt metódus. Az adott tárgy megrajzolását végzi el. A paraméterként kapott g a Graphics osztály egy példánya, ami a rajzolást végzi.
- public abstract void changePosition(int num): Szintén absztrakt osztály. A pozíció megváltoztatásához. A játékos esetén az x koordináta értékét változtatjuk, minden más elem esetében az y koordináta értékét. A praméterként kapott szám értékével kell megváltoztatni a pozíciót.

Player

A játékost leíró osztály. Szerializálható, és a GameObject osztályból származtatott. Minden játékosnak van neve, élete és pontszáma. Az életet egy String típusú tagváltozóban, az életet egy int típusú tagváltozóban, a pontszámot pedig egy long típusú tagváltozóban tárolja.

Az osztály konstruktora beállítja a játékos nevét üres sztringre, az életét háromra, a pontszámot nullára. A játékos mindig a (250; 712) pozícióról indul az alap színének az RGB kódja: 16, 0, 110. A gettereken és a settereken kívül még megvalósítja az alábbi metódusokat:

- public void changePosition(int x): A játékos pozíciójának x értékét változtatja meg a paraméterként kapott számmal. Tehát a pozíció x értéke növekedhet is és csökkenhet is.
- public void draw(Graphics g): Kirajzolja a játékos bábúját. A játékos egy téglalap, aminek a bal felső koordinátja a játékos koordinátája. Szélessége 15, a magassága pedig 40.

Collectible

Absztrakt osztály, amely a GameObject osztályból származtatott. Belőle származtathatóak a gyűjthető tárgyak. Nincs újabb tagváltozója. Az alábbi metódusokat határozza meg:

- public abstract void collideWith(Player p): Absztrakt függvény, megmondja mi történjen, ha az adott tárgyat összegyűjti a játékos.
- public void changePosition(int y): Megváltoztatja a tárgy y koordinátáját, a paraméterként kapott szám értékével.
- public Boolean touchPlayer(Player player): Ellenőrzi, hogy a paraméterként kapott játékos és a tárgy fedik-e egymást, azaz a játékos elkapta-e az adott tárgyat. Ellenőrzi, hogy a tárgy y koordinátája, a játékos legfelső és legalsó y koordinátája közt van-e. Ha igen, akkor ellenőrzi, hogy az x koordinátáik valamelyike megegyezik-e. Ha egyezést talál, akkor a visszatérési érték true. Ha nem, akkor false.

Fruit

Absztrakt osztály, amely a Collectible osztályból származtatott. Belőle származtathatóak azok az osztályok, amelyek a játékos pontszámát változtatják meg. Ezek alapján tagváltozóban tároljuk azt az értékét, amennyivel a játékos pontszáma változik, ha elkapja az adott gyümölcsöt.

A gettereken és a settereken kívűl a következő metódust valósítja meg:

- public void collideWith(Player player): Ha a paraméterként kapott játékos összegyűjtötte, akkor a pontszáma a gyümölcs értékével nő.

Apple

Az almát, mint gyűjthető gyümölcsöt megvalósító osztály, a Fruit osztályból származtatott. Ha a játékos összegyűjt egy almát a pontszáma eggyel nő. További tagváltozója nincs.

Az egyparaméteres konstruktora beállítja az alma pozícióját a paraméterként kapott pozícióba, az értékét egyre állítja, az alap színének az RGB kódja: 132,4,0.

Az alábbi metódust valósítja még meg:

- public void draw(Graphics g): Rajzol, egy 45 átmérőjű kört, amelynek koordinátái megegyeznek az alma koordinátájával. Majd rajzol egy ellipszist, aminek x értéke kilenccel több, mint az almáé az y értéke, pedig huszonhárommal több. A szélessége 30, a magassága 20. A színének RGB kódja: 165, 5, 0. Illetve rajzol egy levelet, ami egy ív, ennek az x koordinátája tízzel több, az y koordinátája tizenöttel kevesebb, mint az almáé. A szélessége 20, a magassága 40, a kezdeti ív 10 és utána 30. A RGB kódja: 15, 65, 2.

Orange

A narancsot, mint gyűjthető gyümölcsöt megvalósító osztály, a Fruit osztályból származtatott. Ha a játékos összegyűjt egy narancsot a pontszáma kettővel nő. További Tagváltozója nincs.

Az egyparaméteres konstruktora beállítja a narancs pozícióját a paraméterként kapott pozícióra, az értékét kettőre állítja, az alap színének RGB kódja: 222, 135, 21.

Az alábbi metódust valósítja még meg:

- public void draw(Graphics g): Rajzol, egy 45 átmérőjű kört, amelynek koordinátái megegyeznek a narancs koordinátáival. Majd rajzol egy ellipszist, aminek x értéke kilenccel több, mint a narancsé, az y értéke pedig huszonhárommal több. A szélessége 30, a magassága 20. A színének RGB kódja: 240, 148, 27.

BlueBerry

Az áfonyát, mint gyűjthető gyümölcsöt megvalósító osztály, a Fruit osztályból származtatott. Ha a játékos összegyűjt egy áfonyát a pontszáma hárommal nő. További tagváltozója nincs.

Az egyparaméteres konstruktora beállítja a narancs pozícióját a paraméterként kapott pozícióra, az értékét háromra, az alap színének RGB kódja: 153, 51, 230.

Az alábbi metódust valósítja még meg:

- public void draw(Graphics g): Rajzol, egy 45 átmérőjű kört, amelynek koordinátái megegyeznek az áfonya koordinátáival. Majd rajzol egy ellipszist, aminek x értéke kilenccel több, mint az áfonyáé, az y értéke pedig huszonhárommal több. A szélessége 30, a magassága 20. A színének RGB kódja: 168, 61, 250.

LifeManipulator

A Collectible osztályból származtatott absztrakt osztály, amelyből azokat az osztályokat lehet származtatni, amelyek azokat a gyűjthető tárgyakat valósítják meg, amik megváltoztatják a játékos életének értékét. Ezek alapján egy tagváltozóban tároljuk az adott tárgy értékét, amennyivel megváltoztatja a játékos életét.

A settereken és a gettereken kívűl az alábbi metódust valósítja meg:

- public void collideWith(Player p): Ha a játékos hozzáér az élete az adott tárgy értékével nő.

Bomb

A bombát, mint gyűjthető tárgyat valósítja meg, a LifeManipulator osztályból származtatott. Ha a játékos összegyűjt egy bombát az élete eggyel csökken. Nincs további tagváltozója.

Az egyparaméteres konstruktora beállítja a pozícióját a paraméterként kapott pozícióra, az értékét mínusz egyre állítja, az alap színének RGB kódja: 7, 66, 73.

Az alábbi metódust valósítja még meg:

- public void draw(Graphics g): Rajzol, egy 45 átmérőjű kört, amelynek koordinátái megegyeznek a bomba koordinátáival. Majd rajzol egy télalapot, aminek az x koordinátája hússzal több, y koordinátája öttel kevesebb mint a bombáé, a szélesége 5 a magassága pedig 15. Végül rajzol egy 8 átmérőjű piros kört, aminek az x koordinátája tizennyolccal több, y koordinátája kilenccel kevesebb, mint a bombáé.

Heart

A szívet, mint gyűjthető tárgyat valósítja meg és a LifeManipulator osztályból származtatott. Ha a játékos összegyűjt egy szívet az élete eggyel nő. Nincs további tagváltozója.

Az egyparaméteres konstruktora beállítja a szív pozícióját a paraméterként átvett pozícióra, az értékét egyre az alap színének RGB kódja pedig: 246, 0, 0.

Az alábbi metódusokat valósítja még meg:

- public void draw(Graphics g): Megrajzolja a szívet két körből és egy háromszögből. A körök 20 egység átmérőjűek, az egyik kör x koordinátája eggyel több, a másik kör x koordinátája tizenkilenccel kevesebb, mint a szív x koordinátája, mindkét kör y koordinátája tízzel kevesebb, mint a szív y koordinátája. A háromszög x koordinátái: a szív x koordinátája, a szív x koordinátája + 40, a szív x koordinátája + 20, az y koordinátái: a szív y koordinátája, a szív y koordinátája, és a szív y koordinátája + 30.
- public void colourChange(Player player): Ha a paraméterként kapott játékos élete egyre csökken, az életek számát jelölő szív elkezd villogni. Ezért, ha a szív piros, akkor a színe a pálya színér vált (RGB kód: 154, 194, 247). Ha a színe a pálya színe, akkor újra piros lesz. Ha a játékos élete egynél több lesz a szív színe újra piros lesz.

FallingCollectibles

Az égből aktuálisan hulló gyűjthető tárgyakat tárolja. Van egy lista tagváltozója, amiben az aktuális tárgyakat tárolja, egy Player tagváltozója, ami az aktuális játékost reprezentálja, egy integer, ami megmondja mennyivel kell változtatni a tárgyak pozícióját, és egy boolean, ami igaz, ha épp fut a roundCheck függvény.

Az egyparaméteres konstruktora beállítja a paraméterként megadott játékost, a boolean változó értékét hamisra állítja, a tárgyak pozíciójának változtatását pedig egyre.

A gettereken és a settereken kívül a következő metódusokat valósítja meg:

- public void increasePositionChange(int num): A paraméterként megadott számmal növeli a pozícióváltás értékét.
- public void positionCheck(Position p): Mivel az új példányok random pozícióban jelennek meg, ezért meg kell akadályozni, hogy átlapolják egymást. Megnézi, hogy az adott pozíció túl közel van-e, minden olyan listában szereplő tárgyhoz, amelyeknek megegyezik valamelyik y koordinátájuk a pozícióéval. Azaz megnézi van-e egyező x pozíciójuk. Ha van, akkor megváltoztatja a pozíció x érétkét, és újra ellenőrzést végez az új pozícióra.

- public void addCollectible(): Ha a roundCheck függvény éppen nem fut, akkor generál egy random számot 0 és 420 közt, ez lesz az új tárgy x koordinátája, az y koordináta mindig -40. Erre a pozícióra meghívja a positionCheck függvényt, hogy megszűnetesse az esetleges átlapolásokat. Generál még egy számot 0 és 1000 között, és elosztja maradékosan tizeneggyel. Ha a maradék egy, a listához hozzáad egy szívet. Ha a maradék 2, 5 vagy 8, akkor narancsot ad hozzá, ha a maradék 3 vagy 6, akkor áfonyát, ha 4 vagy 7, akkor bombát, minden más esetben almát ad a listához. Így randomizált, és a nagyobb értékű tárgyak kisebb valószínűséggel jelennek meg.
- public void roundCheck(): Beállítja a boolean változó értékét igazra. Ha a listában van elem, akkor végigmegy a lista elemein. Ha az adott elemet a játékos begyűjtötte, akkor meghívja az elem collideWith(player) függvényét, majd törli a tárgyat. Ha az adott elem elérte a földet, akkor törli a tárgyat. Ha egyik sem, akkor változtatja a pozíciójukat. A végén beállítja a boolean változó értékét hamisra.

PlayerData

Az eddigi játékosok adatait tároló osztály, az AbstractTableModel osztályból származtatott. A benne tárolt adatokból íródik ki a játék végén a ranglista. Tagváltozója a lista, amely az eddigi játékosokat tartalmazza.

Az alábbi metódusokat valósítja meg:

- public void addPlayer(Player player): Hozzáadja a paraméterként kapott játékost a listához.
- public void Sort(): A lista elemeit csökkenő sorrendbe helyezi a pontszámuk alapján.
- public int getRowCount(): Visszaadja hány sora van a táblázatnak. Ebben az eetben a lista hossza.
- public int getColumnCount(): Visszadja hány oszlopa van a táblázatnak. A ranglistához csak a játékosok neve és pontszáma kell, így ebben az esetben kettő.
- public Object getValueAt(int rowIndex, int columnIndex): Visszaadja egy bizonyos cella értékét. Első oszlopból a játékos nevét, másodikból a pontszámát.
- public String getColumnName(int columnIndex): Megadja az oszlopok neveit. Az első oszlop neve Name, a másodiké Score.
- public Class<?> getColumnClass(int columnIndex): Megadja az oszlop elemeinek az osztályát. Az első oszlopé String, a másodiké Long.

FallThread

Újabb tárgyakat ad a lehulló tárgyak listájához. Mivel időközönként van szükség rá, így időzíteni kell, ehhez implementálja a Runnable interfacet. Tagváltozóként kap FallingCollectibles példányt, aminek listájához az új tárgyakat adja.

Az egy paraméteres konstruktora a paraméterként kapott példányt beállítja. A run metódus pedig meghívja az addCollectible függvényt.

FallPositionThread

Megnöveli az esési sebességet a FallingCollectibles osztály egy példányán. Ezt ugyanúgy csak időközönként kell megtenni, tehát időzíteni kell, így implementálja a Runnable interfacet. Tagváltozóként ugyan úgy kap egy FallingCollectibles példányt.

Az egyparaméter konstruktora ugyanúgy beállítja a paraméterként kapott példányt. A run metódus növeli eggyel a pozícióváltoztatást, ha annak értéke kisebb négynél.

Game

Itt történik a Játék GUI megvalósítása, tehát JFrame osztályból származtatott. Tagáltozót kap az aktuális játékos, illetve az összes játékos adata; ehhez az adatokat a program indulása után betölti egy fájlból, bezáráskor pedig kimenti oda.

A játéknak három része van, a Start menü, ahol a játék szabályai röviden ismertetve vannak, illetve a játékos megadhatja a nevét. Maga a játék. Illetve a Játék vége után a ranglista megjelenítése. Ezek legyen külön JInternalFrameben elhelyezve, tehát mindhárom esetnek lesz egy külön tagváltozója. A ranglistához való kereséshez még lesz egy tagváltozója a TableRowSorternek, illetve a két TextFieldnek, ahova a keresett értékeket lehet beírni.

Belső osztályok

Play

Itt valósul meg maga a játék, és az összes Graphics osztállyal történő rajzolás, ezért a JPanel osztályból származtatott a Play osztály. A gyűjthető tárgyak mozgatása miatt implementálja az ActionListener interfacet, a játékos billentyűkkel való irányításához pedig implementálja a KeyListener interfacet.

Tagváltozói egy timer az Actionlistenernek, egy integer amellyel megadhatjuk mennyit változzon a játékos pozíciója (velx), a FallingCollectibles osztály egy példánya, és a Heart osztály egy példánya (pozíció: 410, 10), hogy ki lehessen rajzolni a szívet, amely az életek számát jelöli. 3 statikus változó a pálya színeinek LIGHT_BLUE (RGB: 154, 194, 247): a pálya háttérszíne, BROWN (RGB: 84, 49, 3) és DARK_GREEN (RGB: 40, 110, 0), a föld színei, amin a játékos áll. Továbbá még két ScheduledExecutorService, hogy időzíteni lehessen a szálakat, amik pozícióváltást és az újabb tárgyak létrehozását kezelik.

Az osztály konstruktora elindítja a Timert, és beállítja a KeyListenert, valamint időzíti a szálakat. A tárgyak esési sebessége minden percben nő. A tárgyak hozzáadását pedig randomizálja.

Az alábbi metódusokat valósítja meg:

- public void paintComponent(Graphics g): Megrajzolja a pálya elemeit. A velx pozícióját nullára állítja. Megrajzolja a pálya alját, ahol a játékos áll, ez egy barna és egy zöld téglalap. Valamint a bal felső sarokba kiírja pontszámot, a jobb felső sarokban pedig az életek számát. Meghívja a heart colourChange fügvényét, majd megrajzolja azt is. Megrajzolja a játékost. Végül megrajzolja a listában szereplő lehulló elemeket.
- public void actionPerformed(ActionEvent e): Az ActionListener kötelezően megvalósítandó függvénye. Ha a játékos élete nagyobb nullánál, megváltoztatja a gyűjthető objektumok pozícióját, ha a jobbra/blara nyíl vagy az A/D betűk le vannak nyomva megváltoztatja a játékos pozícióját, újra rajzolja a pályát. Ha a játékos élete nullára csökkent eltűnteti a pályát, és az internalframet, amibe el lett helyezve, a játékos éltet -1-re állítja, hogy egyik loopba se ugorjon be többet. Meghívja az End metódust.
- public void right(): Beállítja a velx változót tízre, ha a játékos x pozíciója 451-nél nagyobb, akkor nullára állítja.

- public void left(): Ha a játkos x pozíciója kisebb kettőnél a velx változót nullár állítja, egyébként mínusz tízre.
  - public void keyPressed(KeyEvent e): KeyListener kötelezően megvalósítandó metódusa. Ha az A betű, vagy a balra nyíl le van nyomva meghívja a left metódust. Ha a D betű, vagy a jobbra nyíl le van nyomva meghívja a right metódust.
  - public void keyTyped(KeyEvent e): KeyListener kötelezően megvalósítandó metódusa. Nincs rá szükség így nem implementáltam.
  - public void keyReleased(KeyEvent e): KeyListener kötelezően megvalósítandó metódusa. Nincs rá szükség így nem implementáltam.

NameSearchTyped & ScoreSearchTyped

Név és pontszám szerinti kereséshez, implementálja a DocumentListener interfacet. A következő metódusokat valósítják meg:

- public void insertUpdate(DocumentEvent e): Kötelezően megvalósítandó metódus. Ha beleírnak a TextFieldbe, beállítja a rowSorter filterét a textFieldből kapott szövegre. Kezdőbetűvel keres, név szerinti keresésnél az első oszlopot nézi, pontszám szerinti keresésnél a második oszlopot nézi.
- public void removeUpdate(DocumentEvent e): Kötelezően megvalósítandó metódus. Ha törölnek a TextFieldből, beállítja a rowSorter filterét a textFieldből kapott szövegre.
- public void removeUpdate(DocumentEvent e): Kötelezően megvalósítandó metódus, azonban nincs rá szükség, így nincs implementálva.

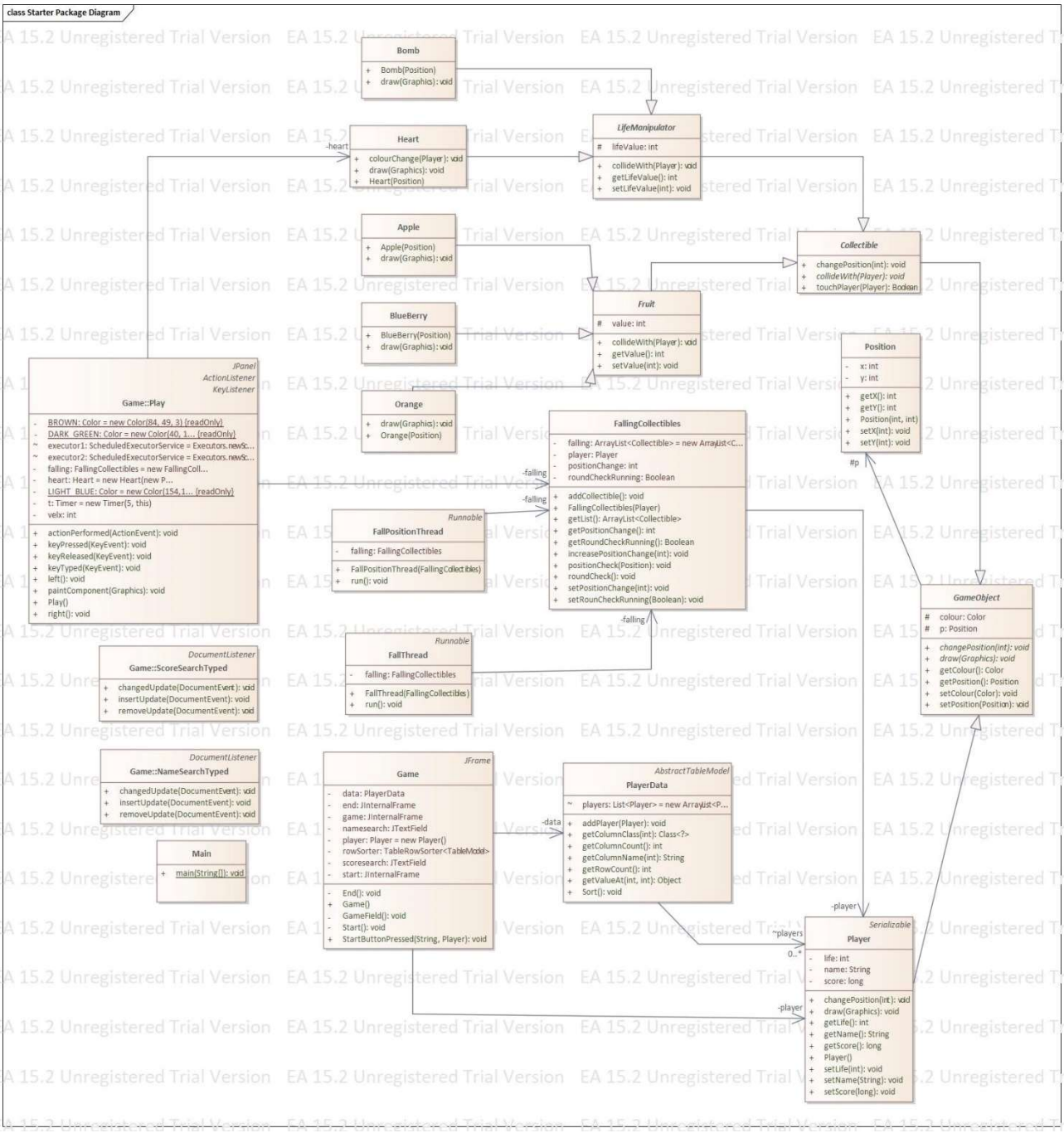
A **Game** osztály az alábbi metódusokat valósítja meg:

- A konstruktora beállítja, hogy a kilépés gomb lenyomásával lépjen ki a program. A játék elindításakor betölti a players.dat fájlból az adatokat a PlayerData példányába. Az ablak bezárásakor pedig kiírja őket a fájlba. Beállítja a méretet (500, 900)-ra. Bekapcsolja, hogy ne lehessen átméretezni és elindítja a Start függvényt.
- private void Start(): Az InternalFrame címét beállítja Start-ra. Megjelenít egy TestFieldet, ami elé Name: van írva. A másik oldalra pedig egy Start! feliratú gombot. Alatta egy nem szerkeszthető TextAreaben ismerteti a játék szabályait. A Start gomb lenyomására meghívódik a StartButtonPressed metódus.
- public void StartButtonPressed(String str, Player p): Ha a paraméterként kapott sztring nem üres, akkor beállítja a paraméterként kapott játékos neveként. Etűnteti a sart ablakot, és meghívja a GameField metódust.
- private void GameField(): Létrehoz egy példányt a Play osztályból, hozzáadja a game InternalFramhez, és láthatóvá teszi azt.
- private void End(): Hozzáadja a PlayerData példányához az aktuális játékost. Létrehozza a JTablet, beállítja rajta a szükséges beállításokat. Létrehozza a két JLabelt és a két JTextFieldet a keresésekhez, a két TextFieldhez pedig hozzáadja a megfelelő documentListenereket.

Main

A program belépési pontja. Létrehoz egy Game példányt, és láthatóvá teszi.

Osztálydiagram



Felhasználói kézikönyv

A játék elindításakor a felhasználónak meg kell adnia egy nevet, majd a „Start!” gomb lenyomására elindul a játék. Ameddig nincs semmi beírva a név mezőbe, addig a gomb lenyomására nem történik semmi.

A játék elindítása után a játékos a pálya alján található bábút a jobb és a bal nyilak, illetve az A és a D betűk lenyomásával tudja irányítani. Az égből folyamatosan gyümölcsök, bombák és szívek potyognak. A játékos úgy tudja összegyűjteni őket, ha alájuk áll miközben esnek és hozzájuk ér. Ha bombát kap el véletlenül, akkor az élete eggyel csökken. De az életet növelni is lehet, ha elkap egy szívet. Ha valamilyen gyümölcsöt kapott el, akkor a gyümölcs értékének megfelelően nő a pontszáma. A gyümölcsöknek három fajtája van: az alma egy pontot ér, a narancs két pontot ér, és az áfonya három pontot ér.

A játék célja, hogy az adott játékos minél több pontot összegyűjtsön, mielőtt a játék véget ér. A játék akkor ér véget, ha a játékos életereje nullára csökken.

A játékablak bal felső sarkában látható az aktuális pontszám. A jobb felső sarokban pedig az aktuális élet értéke. Ha a játékos élete egyre csökken, az életek számát mutató szív elkezd villogni.

Ha a játék véget ért megjelenik az aktuális ranglista, alapból pontszám csökkenő sorrendben. De a táblázat oszlopainak nevére kattintva lehet rendezni őket csökkenő és növekvő sorrendbe is rendezni név és pontszám szerint is. Valamint lehet bennük keresni név és pontszám alapján is. Ha a keresési mezőbe beleírnak, akkor azonnal megtörténik a keresés, ha a keresési mezők üresek, akkor pedig megjelenik újra a teljes ranglista.

Kellemes játékot és sok szerencsét!