**Kezdetleges horgász játék Java nyelven**

A program írása során próbáltam figyelni a helyes felépítésre, véleményem szerint sikerült is legtöbbször. Alapvetően a legtöbb különálló dolgot szétbontottam külön osztályra, és külön csomagokba.

Alapvetően a játék nem ilyen formában készült volna el, de időhiány miatt, és egyéb nehézségek miatt lett ennyire kezdetleges. Eleinte azt szerettem volna, hogy fent, a horog középen jobbra-balra forogjon, majd szóköz lenyomására amerre néz, arra induljon el. Ezek nyomai a Kepmunka osztályban még bent maradtak. Itt olyan dolog történt, hogy ahogy forgattam 5-10 fokonként, valamiért változott a mérete, pozitív irányba, elkezdett növekedni, így teljesen elcsúszott. A másik probléma az volt, hogy balról rögzítve volt a mérete, így mikor arra forgattam, és a kép a forgatás után kilógott a képről, olyankor levágta a szélét.   
A halak is máshogy működtek volna. Szerettem volna azt, hogy arra nézzenek a halak, amerre mennek, ezt forgatással megoldani. Itt szerintem az én metódusomban voltak gondok, mert volt, hogy jó fele nézett, de olyan is akadt, hogy teljesen rossz irányba nézett, vagy esetleg ellentétesen, és hátrafele úszva. Ezeket lebutítva jutottam el a mostani projekthez. Mindenképp szerettem volna adatbázist használni, de mivel eléggé kezdő vagyok benne, ezért hibás is.  
A projekttel sokat tanultam a képek kezeléséről, az adatbázisokról. Legfőképpen az sql lekérdezek vették el a projekt írásának több mint a felét.

Az adatkezelés csomagban a LoginDAO és a FelhasznaloDAO található. Ezek hivatottak az adatbázissal való kapcsolat tartásra. LoginDAO felel a bejelentkezésért, és a regisztrációért. Itt Whirlpool algoritmust használok a jelszavak hash-eléséhez. Minden felhasználóhoz egyedi "salt"-ot generál, mellyel a jelszót összefűzve, összehasheljük, így elkerülhető, hogy ugyanazon jelszóhoz ugyanazon hash tartozzon. Ezt regisztrációkor hozza létre véletlenszerűen. A bejelentkezésnél először felhasználónévre rákeresünk, és ha találtunk lekérjük a hash értékét, és a saltot, majd ezen salt-al, az input mezőbe beírt jelszót összefűzzük, majd hasheljük. Ha az adatbázisban tárolt jelszó megegyezik az így kapott jelszóval a bejelentkezés sikeres lesz.  
Minden bejelentkezésnél, mind regisztrációnál lekéri hogy foglalt e a név, és ha igen, akkor hibával tér vissza. Azért a Whirlpool algoritmus mellett döntöttem, mert az md5 ellen sok véleményt olvastam, hogy nem ajánlott a nagy hash adatbázis miatt (szivárvány tábla), és egy 'erősebb' algoritmus mellet döntöttem.  
A FelhasznaloDAO lényegében a felhasználó adatait kezeli, legfőképp az Eredményeket menti, és írja adatbázisba. ////Jelenleg ismert hiba, hogy duplikálja az adatokat, ez ellen egy unique mezővel próbáltam védekezni, de így pedig hibát dob, és nem frissíti////

Az alaposztályok minden olyan osztály, és segédosztály, mely nem kapcsolattartásra hivatott és nem a vezérlő csoporthoz tartozik. Ide soroltam be minden olyat, ami csak pár funkció megvalósítására hivatott, vagy esetleg egy függvénynek két értékkel kellett volna visszatérnie, és szükség lett egy osztályra, melyben eltárolom az értékeket, hogy később kiolvassam (Password,Pozicio).  
Az Akadaly osztály felel a vízben úszkáló mínusz pontot érő elemekért. Jelenleg két ilyen elem van, egy hínár és egy uszadék fadarab. Egyes nézőpontok szerint felesleges ez az osztály, hisz nem valósít meg semmit, csak örököl a ViziElem osztályból, és nem 'terjeszti' ki azt. Legfőképp azért hoztam létre, hogy meg tudjam különböztetni az 'instanceof' kulcsszóval.  
Az Eredmeny osztály ír le egy eredményt. Az eredmény tárolja a dátumát, és a pontját, és formázott toString metódussal rendelkezik, és megvalósítja a Comparable interfészt.  
A Felhasznalo osztály tárolja lényegében a felhasználónk adatait. Tárolja a felhasználó nevet, és egy listát az eredményekről. A setterek-gettereken kívül egy metódust valósít meg, mely az addEredmény(Eredmény e), amely hozzáad egy átadott eredményt a listához.  
A Hal osztályunk hasonló az Akadaly-hoz, itt is az instanceof kulcsszó lényeges volt mikor megalkottam ezt az osztályt. Itt felülírom az eredeti rajzolas metódust, hogy egy esetlegesen elforgatott, átméretezett képet rajzoljon ki. Erre azért van szükség, hogyha hátrafele úsznak, olyankor átfordul a megfelelő irányba.  
A Horgasz osztály felel a kis horogért (és megvalósítja Runnable intefészt), melyet a nyíl billentyűkkel és a szóközzel irányítunk. Tárolja a horog képét, az x,y értékét, hogy a szál aktív e, tárolja a szál várakozási idejét, a vezérlőt, a dx,dy értéket, azaz hogy mennyit mozogjon, és adatokat a játéktér nagyságáról, plusz egy változót, hogy lefele mozog e éppen. A gettereken és settereken kívül felülírja a run metódust, melyben csak a mozgás, a várakozás, és a frissítés szerepel. A rajzolás metódus felel azért, hogy kirajzoja a képet, azaz a horgot, és ha lefele megy a horgunk, akkor rajzoljon egy "madzagot" a játéktér tetejétől, a horog tetejéig. A mozog metódus felel a mozgásárért a horognak. Lényegében a 'lefele' változó változására reagál, mely ha igaz, olyankor a horog elkezd lesüllyedni addig, amíg nem ütközik egy úszó játékelemmel, vagy nem éri el a játéktér alját. Ha eléri, olyankor kétszeres sebességgel felemelkedik. Itt adódott egy hiba, miszerint volt, hogy 'túl magasra emelkedett', így nem tudtuk leküldeni újra, mivel védve van azellen, hogy ment közben újra leküldjük. Itt azzal javítottam, ha túl magasra kerülne, olyankor visszaállítjuk 0-ra az értékét, ezen hiba túl közeli halak elkapásánál jött elő. Mihelyt elér valamit vagy leér az aljára a lefele változó hamis lesz. A varakozik metódus csak a szálat altatja a megadott ideig. A frissít metódus megkéri a vezérlőt hogy frissítse a rajz panelt. Az inditas metódus elindítja a szálat. Az iranyba metódus vizsgálja hogy melyik gombot nyomtad le. Itt egy láncolat van jelen, miszerint a panel átküldi hogy épp mit nyom le, a vezérlő tovább küldi ide, melyet itt lekezelünk. -1,0,1 értékeket vehet fel, ahol a –1 a balra, a 0 a lefele, az 1 a jobbra. Ha -1, olyankor az x csökken dx értékkel, viszont itt van egy feltétel vizsgálat, hogy semmiképp ne fusson túl, az 1-es értéknél ugyan ez, csak pozitív irányba vizsgál, és léptet. A 0, azaz lefele értéknél megnézzük hogy a horog fent van e, és ha igen, olyankor a lefele változót igazra állítjuk.  
A Kepmunka osztály csak statikus metódusokból áll. Itt a nevéből is adódóan a képmanipilációk foglalnak helyet. Itt a toBufferedImage átalakítja a sima image típust bufferedImage típusra, hogy alkalmazhassuk rá a transzforméciókat. A resize metódus csak átméretezi a képet a megadott értékekre. A flipImageVertial metódus megfordítja a képet függőlegesen. Ez hasznos ha a hal hátrafele úszna, és nem akarjuk azt.  
Ebben az osztályban bent hagytam még két toRotatedImage metódust, melyek más más paraméterek alapján fordítják el a képet. Az alap elképzelésben a halak amerre úsznak, arra néztek volna, de ezt időhiány és egyéb nehézségek miatt elvetettem. Érdekes, ha még nem is egészen jól működő kódok.  
A Password osztály tárolja a jelszóHash-t és a salt-ot. Ezt csak visszatérési érték miatt hoztam létre.  
A Pozicio osztály tárol egy x,y értéket, melyek a halak mozgásához kellettek. Így egybefogva egyszerűbbnek tűnt.  
A RendezhetőListModell osztály az órákról jól ismert, és használtosztály csak felhasználtam a szebb kiíratás miatt.  
A ViziElem osztály megvalósítja a Runnable intefészt, hogy minden vizi elem szálként viselkedhessen .Tárol egy képet, xy,dx,dy értékeket. Egy poziciót, magát a vezérlőt, egy szál futásáért felelős aktív értéket, és a játéktér méretére vonatkozó információkat. Felülírja a run metódust, melyben amíg aktív a szál, addig a mozog,várakozik,frissít hármas szerepel. A konstruktor hívásakor irényba is állítjuk, az iranybaAllit metódussal. A rajzolas metódus kirajzolja a megadott képet. A mozog metódus csak a mozgásáért felel azaz az x, és y értéket növeli a dx, dy értékével. Ha kikerülünk a játéktérből olyankor tötöljük az egyedet, és leállítjuk a szálat. A várakozik metódus egyszerűen várakoztatja a szálat a megadott idővel. A frissít metódus megkéri a vezérőt hogy frissítse a rajz panelt. Az irányba állít metódus a cél pozíció alapján kiszámolja az dx, dy értékét, hogy megfelelő irányba induljanak a dolgok.

A felületek csomag tartalmazza a paneleket, melyeket használtam. A BelépésPanel tartalmazza a regisztrációhoz szükséges beviteli mezőket és gombokat, mely adatokat gombnyomásra tovább küldi a vezérlőnek hogy lekezelje azokat. Az Eredmeny panel felelős az adatok betöltéséért. Itt mikor megnyílik kéri a vezérlőt hogy töltse be a ranglistát. Beállítja a játékos nevét, ha a vezérlő meghívja, és kiírja a játékos, és a globális eredményeket is. A MenuPanel-en kaptak helyet a gombok, melyek megnyomására más, más ablak nyílik, vagy épp elindul a játék. Itt jelezzük a hátralévő időt is és a pontokat. A RajzPanel a dolgok kirajzolására való.

A vezerlo csomagban a vezérléshez szükséges dolgok találhatóak. Itt van egy EgyébFrame, mely csak azért létezik, hogy ez ugorjon fel, a menü, és eredmény panelekkel, gombnyomás hatására.  
A global osztály tartalmazza a globális változókat.  
A horgász frame szerepe hogy elindítsa a vezérlőt, mely beállít így minden referenciát.  
A vezérlő osztály irányítja a program futását, reagál a hívásokra, és továbbítja azokat a megfelelő helyekre. Tartalmaz minden olyan listát, változót mely a futáshoz szükséges.  
Az inditas metódus hatására beállítja a statikus változókat, a panelekre átadja a vezérlőt, adatbázis kapcsolatot épít, és létrehozza a rajzPanel hátterét.  
Az eredmenyClick hatására előtűnik az eredményPanel, az egyébFrame-n és kiírja az eredményeket.  
A belepesClick hatására előjön a belépésPanel, és lehetőségünk lesz bejelentkezni.  
A startClick hatására elindul a játék futása. Pontokat lenullázza, törli a listát, ha van benne elem. Létrehozza a horgászunkat, elindítja azt és értkéket ad neki, majd saját magát is elindítja szálként.  
A sugoClick előhozza a súgót. A sikereBelepes metódus bejelentkezteti a játékost, vagyis megkéri a LoginDAO-t a művelet elvégzésére.  
A regisztracio regisztrálja a játékost, a LoginDAO segítségével.  
A ranglistaBetoltes betölti a ranglistát, és átadja az eredmenyPanel-nek.  
A kapcsolatEpit létrehozza azadatbázis kapcsolatot, ha nincs tábla, létrehozza.  
A rajzolas kirajzolja a viziElemeket, és a horgasz-t.  
A frissites repaintot kér a rajzPanel-től.  
A run metódus felel a viziElemek generálásáért, és a 30 másodperces játékidőért.  
A gomblenyom metódus továbbküldi az adott értéket a horgasz-nak.  
A pozicio egyezés metódus figyeli hogy az átadott értékek alapján találunk e viziElemet.