Tartalomjegyzék

0.1. Készítette

Pilipár Botond

Email-cím: pilipar.botond@gmail.com

Neptun-kód: KD417D

Csoportszám: 12

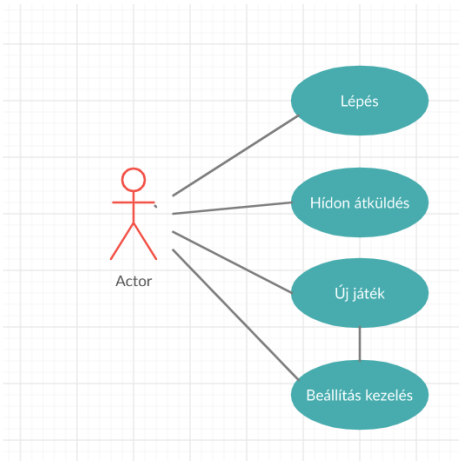
1. Feladat elemzése

1.1. Feladat

Készítsünk programot, amellyel az utazó kereskedők problémáját mutatjuk be. Ebben néhány kereskedő éjszaka ér egy mély szakadék felett átívelő rozoga hídhoz. Ezen a hídon a sötétben csak lámpával lehet biztonságosan átkelni, de az utazóknak sajnos mindössze egyetlen lámpájuk van, továbbá egyszerre legfeljebb hárman juthatnak át, és valakinek vissza kell menni a lámpával a többiekért. A kereskedők különböző korúak (fiatal, középkorú, idős), ezért eltérő idő kell nekik a hídon való átkeléshez. Természetesen, amikor többen mennek át egyszerre, akkor a lassúbbhoz kell igazítani a lépést. Jelenítsük meg a szakadék két partján lévő kereskedőket, biztosítsuk azt, hogy mindig felváltva tudjunk egyik vagy másik partról 1-3 személyt kiválasztani, amely átkerül majd a másik oldalra. A program folyamatosan számolja az átkelés idejét (természetesen nem valós időben, hanem az előre megadott átkelési idők segítségével). A program biztosítson lehetőséget új játék kezdésére a fiatal, középkorú és idős kereskedők számának megadásával (0-tól 5-ig, minimum 3 különböző összeállításból választva), és ismerje fel, ha vége a játéknak. Ekkor jelenítse meg, milyen idővel győzött a játékos, majd kezdjen automatikusan új játékot.

1.2. Elemzés

A játékot egy játékos játssza, aki kezeli mind a táblán szereplő játékosokat (kereskedők), mind a játék menetét. A játékos megállíthatja, folytathatja, újrakezdheti a játékot. A felhasználói esetek közül az új játék kezdése és a beállítások módosítása valójában ugyanaz a tevékenység, hiszen a kereskedők számának megváltoztatásával változik a játék nehézsége. Emiatt, ha a felhasználó beállításokat módosít, muszáj neki új játékot kezdenie.



2. Felhasználói esetek

2.1. Esettáblázat

1.	Aklakmazás elindítása	GIVEN:	Az alkalmazás feltelepült
		WHEN:	Futtható alkalmazás elindítása
		THEN:	Megjelenik a játéktábla, melyben az alapértelmezett számú és gyorsaságú játékos helyezkedik el.
2.	Lépés	GIVEN:	A játék aktív állaptoban van, a kontroll gombon "Megállítás" felirat látható.
		WHEN:	Felhasználó rákattintott valamely, eddig ki nem választott, aktuális oldalon várakozó játékosra
		THEN:	A kiválasztott játékos a híd várakozó oldali feléhez lép
3.	Lépés	GIVEN:	A játék aktív állaptoban van, a kontroll gombon "Megállítás" felirat látható, valamely játékos a hídnál várakozik
		WHEN:	Felhasználó valamely hídnál várakozó játékosra kattint
		THEN:	A hídnál várakozó játékos visszatér az aktuális oldal ki nem választott játékosaihoz
4.	Hídon átküldés	GIVEN:	A játéktábla aktív állapotban van, a kontroll gombon "Megállítás" felirat látható, korábbról valamelyik játékos a hídnál várakozik
		WHEN:	Felhasználó a "Mehet" gombra kattint
		THEN:	A híd adott oldalán az összes játékos átkel a hídon, a legnagyobb pontszámú (leglassab) játékos átkelési ideje a játék összpontjához hozzáadódik
5.	Beaállítások kezelése	GIVEN:	Az alkalmazás fut
		WHEN:	Felhasználó az "Beállítások" gombra kattint
		THEN:	Megjelenik a beállítások ablak, ahol a kiválasztott játékosszámokkal kezdődik új játék, minden játékos a kezdeti oldalon fog várakozni

2.2. Felületi terv

A játék egy háttérrel rendelkező *Widget*-ben zajlik, melynek jobb alsó sarkában egy *GroupBox* található, benne a játék vezérléséhez szükséges gombok:

- Új játék gomb
- Beaállítások gomb
- Megállítás/Folytatás gomb

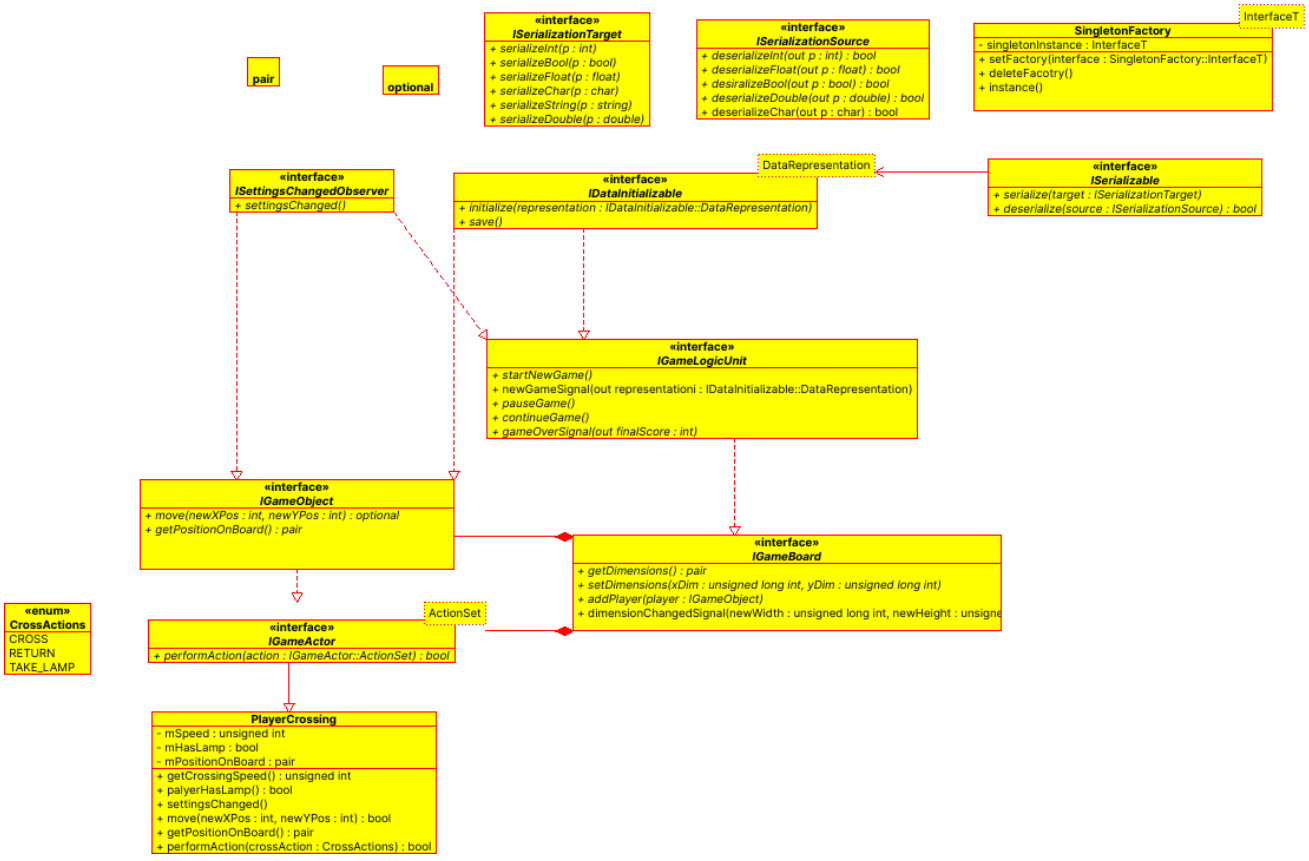
A beállítások vagy új játék gomb hatására megjelenik a beállítások ablak, egy *DialogBox* ahol *SpinBox*-ok segítségével állíthatjuk az idős, fiatal és középkorú kereskedők számát.

Ha egy játékos megnyeri a játékot, egy *MessageBox* tájékoztatja az elért pontszámról.

A játékosok eleinte a bal alsó sarokban várakoznak, majd kiválasztva a megfelelő part oldali hídfőhöz ugranak, majd átküldéskor a jobb felső sarokban foglalnak helyet. A mozgatási állapotokat *Layout*-ok reprezentálják.

3. Logikai terv

3.1. Osztálydiagram



3.1.1. Vezérők, adattagok

Az alkalmazásnak két fő logikai kompnense van, a játékos és tábla. A tábla tartalmazza a játékost, ő hozza létre és semmisíti meg. Minden játékosnak egyforma akciói vannak, amiket a megfelelő felsorolási típus ír le.

A nézet osztály tartalmazza a logikát és események fogagásával tájékozódik, ha a tábla a felhasználó tevékenysége miatt változik. Tájékoztatja továbbá új játék kezdetéről, játék végeztéről nyerés esetben. A nézet rétegnek nincs tudomása a játék éppeni folyásáról, nem tudja ki melyik oldalon van, vagy éppen milyen stádiumban jár a kiválasztás, milyen lépések megengedettek.

3.1.2. Metódusok

A játékos mozogni tud a *MOVE_TO_BRIDGE*, CROSS, RETURN paraméterekkel, amiket a **performAction** metódusa kap meg. Egy játékos nem léphet miden állapotból minden állapotba, ezt a belső logikája figyeli. A játékos általános esetben tájékozódni szeretne arról, történt-e változás a beállításokban. Ezt a működést az **Observer** pattern-ből ismert értesítő **settingsChanged** metódus látja el.

A tábla a külső szemlélő számára vezérlőként működik, mert lehet elindítani, megállítani, folytatni a rajta levő játékot, valamint lehet ezen kersztül játékosokat mozgatni (vagy legalábbis próbálni). Ezeket a funkciókat rendre a **startNewGame**, **pauseGame**, **continueGame** valamint **movePlayer** metódusok látják el.

3.1.3. Eseménykezelés

Eseménykezelés megvalósul a nézet-tábla valamint a tábla-játékos aktorok között. A nézet által kezelt nyomógombok a tábla bizonyos metódushívásait idézik elő. Ezeket a hívásokat a tábla továbbítja a játékosoknak, majd eseményben tájékoztatást kap annak sikerességéről. Természetesen az is lehet, hogy a játékos által észlelt lépés engedélyezett, de a jelenlegi játékállapot ezt nem teszi lehetővé (például a kezdeti oldali játékosválasztás közben az érkező oldalról akarunk játékost választani). Ahhoz hogy a nézet nyomógombjai ne zavarják össze a játéklogikát, ezeket egy állapotgép kezeli.

sender	signal	reciever	slot
BridgeCrossingPlayer	actionPerformedSignal	BridgeCrossingBoard	onPlayerActionPerformed
BridgeCrossingBoard	newGameSignal	BridgeCrossingViewManager	onNewGameStarted
BridgeCrossingBoard	boardChangedSignal	BridgeCrossingViewManager	onBoardChanged
BridgeCrossingBoard	scoredPointChangedSignal	BridgeCrossingViewManager	onScoredPointChanged
BridgeCrossingboard	gameOverSignal	BridgeCrossingViewManager	onGameOver