Elektronikus fizetési eszköz JavaCard 2.0 szabvány alapján

Ondi Attila
att@valerie.inf.elte.hu

1999. május 6.

Tartalomjegyzék

A feladat

Egy JavaCard 2.0 szabvány alapján működő eszköz elektronikus pénztárcaként való felhasználása, például a bank – pontosabban annak ATM automatája – és a Hallgatói jegyzetbolt között. (Természetesen más helyekre – mint pl.: italautomata, vagy BKV bérletvásárlás – is implementálható a – szakdolgozatban – megvalósított kommunikációs metódusokkal, és a megfelelő olvasóval.) A program írása során főleg az eszköz és az azt fogadó terminál közötti kommunikáció biztonságosságára törekedtem, és az ezen kívüli kommunikációval (mint pl. az ATM automata és a Bankközpont között) nem foglalkoztam részletesen.

A dolgozatot a Dallas Semic. Inc. által készített Dallas iButton-ön – pontosabban egy iButton-t magábanfoglaló JavaGyűrű-n (JavaRing) a továbbiakban: gyűrű – teszteltem.

A terminálok – a dolgozatban az `ATMAblak` és a `JegyzetBoltAblak` osztályok – jelentik a kapcsolatot a felhasználó és az eszköz között, így itt fokozottabban kell ügyelni az ember (szándékosan vagy kellő hozzáértés hiányában) okozta hibalehetőségek kezelésére és kivédésére.

JavaGyűrű

JavaGyűrű

Gyűrű és olvasó

A gyűrű

A JavaGyűrű (a többi Java eszközzel egyetemben) egy intelligens eszköz, mely képes Java programokat futtatni. Minimális rendszerkövetelményei: 16 kilobyte csak olvasható memória (ROM), 8 kilobyte EEPROM, 256 byte véletlen elérésű memória (RAM)

Normális esetben a JavaKártya/JavaGyűrű nem rendelkezik kijelzővel, vagy billentyűzettel, hanem soros, vagy párhuzamos kommunikációs felületet használ ahhoz, hogy a külvilággal érintkezzen.

A Java eszközök – más számítógépekhez hasonlóan – ún. csomagokon keresztül kommunikálnak a világgal, melynek neve Alkalmazás-Protokoll Adategység (Application Protocol Data Unit - APDU). Az APDU vagy parancs (command) vagy visszatérési (response) üzenetet tartalmaz. Az eszköz mindig a ,,szolga" szerepét játsza a mester-szolga modellben, mely szerint mindig az eszköz vár egy hozzá címzett APDU-ra. Ha a kapott APDU értelmezhető utasításokat tartalmaz, akkor végrehajtja azokat, és egy választ

küld vissza az olvasó felé. Ellenkező esetben az eszköz a sikert jelentő `0x9000` hexadecimális kód helyett a hiba okára utaló értéket (Status Word) küld visszza. A parancs és visszatérési APDU-k így felváltva küldődnek az olvasó és az eszköz között.

A parancs APDU a következőképpen épül fel:

- **Fej**
  - **CLA** – Osztály byte (Class): az alkalmazás azonosítására szolgál
  - **INS** – Utasítás byte (Instruction): az utasítás kódja
  - **P1** – Paraméter byte: az utasítás finomítására
  - **P2** – lásd: P1
- **Törzs**
  - **Lc** – az adatmező hosszát jelzi
  - **Adatmező** – a küldött adat
  - **Le** – a válasz várt maximális hossza

A visszatérő APDU szerkezete:

- **Törzs**
  - **Adatmező** – a válasz adat
- **Farok**
  - **SW1** – (StatusWord1) a végrehajtás sikerességét jelző byte
  - **SW2** – lásd: SW1[1]

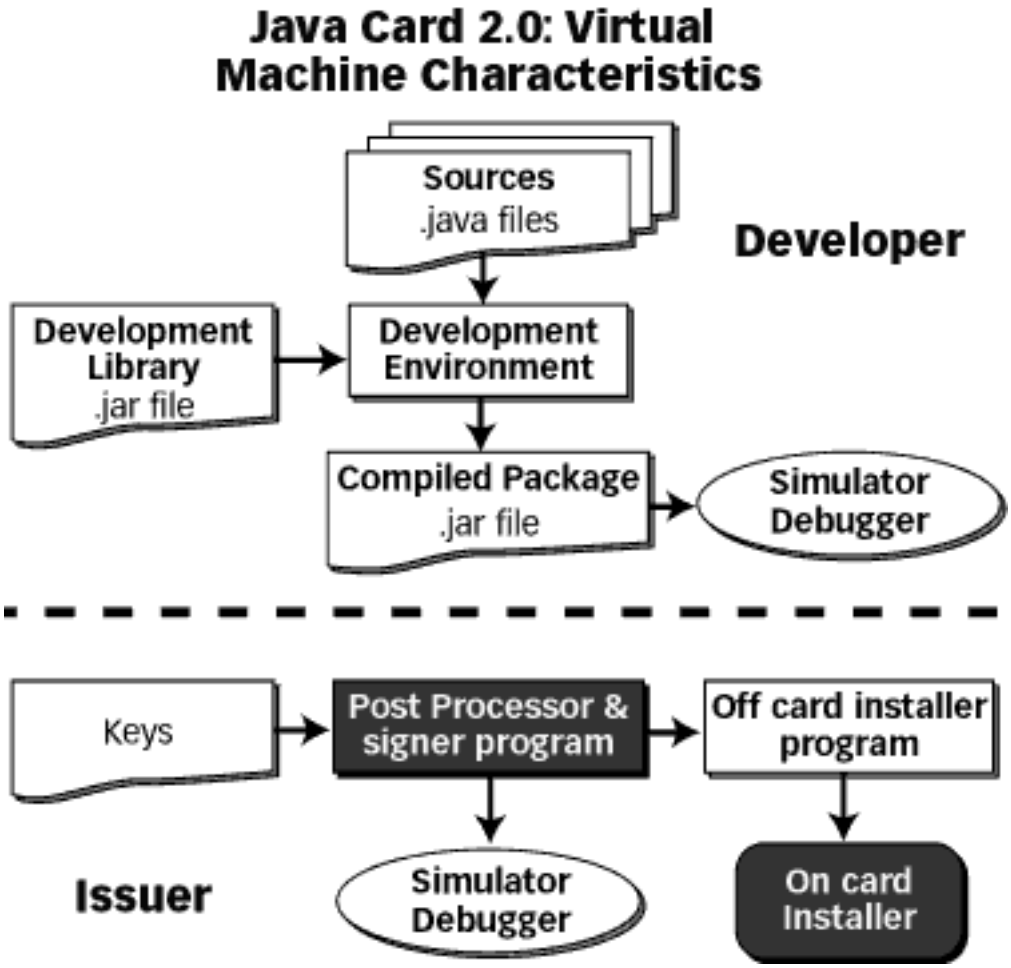
Egyéb jellemzők:

- Platformfüggetlenség: ugyanazon alkalmazás több gyártó eszközén is használható
- Több alkalmazás támogatása: egy eszközön több alkalmazás is lehet egyszerre, de ezek közül csak egy lehet aktív
- Alkalmazások a gyártás után is feltölthetők az eszközre

[1]SW1 és SW2 együtt adja meg a hibakódot (alsó, felső byte)

- A JavaGyűrű legfeljebb 115 byte hosszúságú APDU-t képes fogadni. Az ennél hosszabb adatot tördelve kell (max. 115 byte hosszú csomagokban) elküldeni, amit majd a gyűrűn levő alkalmazás fog a megfelelő sorredben összepakolni

- File-rendszere hasonló a személyi számítógépekéhez: egyetlen gyökér file/könyvtár van, mely alatt további elemi (elementary) és alkönyvtár file-ok helyezkedhetnek el. Az elemi file-ok a sokoldalú felhasználhatóság érdekében több különböző módon hozhatók létre (pl.: ciklikus, rekord, ... file)



Az alkalmazás útja a megírástól a gyűrűig

A szabvány

A gyűrű programozásának szabványosítására a Sun Microsystems létrehozta a JavaCard szabványt (mely jelenleg a 2.1-es változatnál tart), mely alapján készült programok kompatíbilisek maradnak egymással.

A gyűrű kis memóriakapacitása miatt néhány korlátozást vezettek be, melynek értelmében a gyűrű (és a többi JavaKártya) nem támogatja a:

- Dinamikus osztálybetöltést (Dinamic class loading)
- Biztonsági ellenőrzést (Security manager)
- Szálakat és a szinkronizációt (Threads and synchronization)
- Objektum másolást (Object cloning)

- Objekum befejezést (Finalization)

- Nagyméretű adattípusokat (float, double, long, char)

Ezért az ezeket jelző kulcsszavakat is kitiltották a nyelvből.

A Virtuális Gép (Virtual Machine) támogatja a 32 bit-es egész típust és a később feltöltött alkalmazásokat[2].

Az Europay, Mastercard és a Visa együttműködésével létrejött az EMV szabvány, melynek célja, hogy üzleti alkalmazásoknak megfelelő feltételekkel bővítse az intelligens kártyákra vonatkozó ISO 7816-os szabványt. A könnyebb implementálhatóságért[3] egy könnyű és átlátható programozási felületet biztosít, ami elrejti a gyűrű felépítését.

A biztonság

A Java alkalmazásoknak a Java nyelv biztonsági követelményeinek kell megfelelni, bár a JavaKártya biztonsági modellje számos helyen különbözik: a Java alkalmazás létrehozhat objektumokat, melyek adatokat tárolnak és/vagy módosítanak. Ugyan egy alkalmazás hivatkozhat egy objektumra, de nem hívhatja meg annak metódusait, kivéve, ha az objekum az övé, vagy osztott. Egy alkalmazás megoszthatja az objektumait néhány, vagy akár az összes alkalmazással.

Minden alkalmazás önálló egység az eszközön. Kiválasztása (selection), végrehajtása (execution) és működése nem hat más alkalmazásokra ugyanazon eszközön.

[2]ezek azok az alkalmazások, melyeket a gyűrű hordozója tölt fel, miután gyűrűjét/kártyáját a gyártótól megkapta

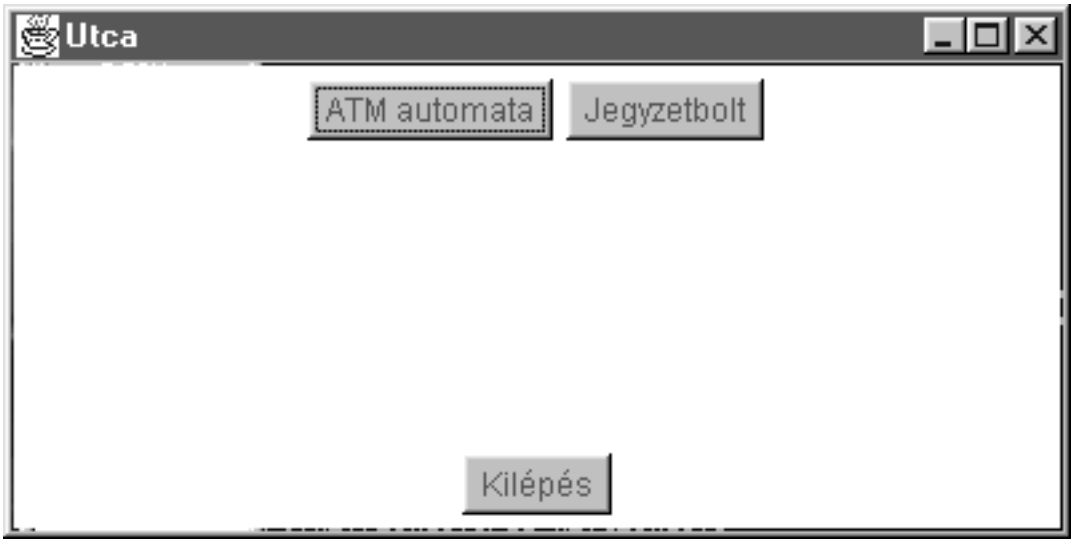
[3]eszközre ültethetőség

Használati útmutató

Indítás

A programot a '`java Gyuru`' parancsal indíthatjuk.

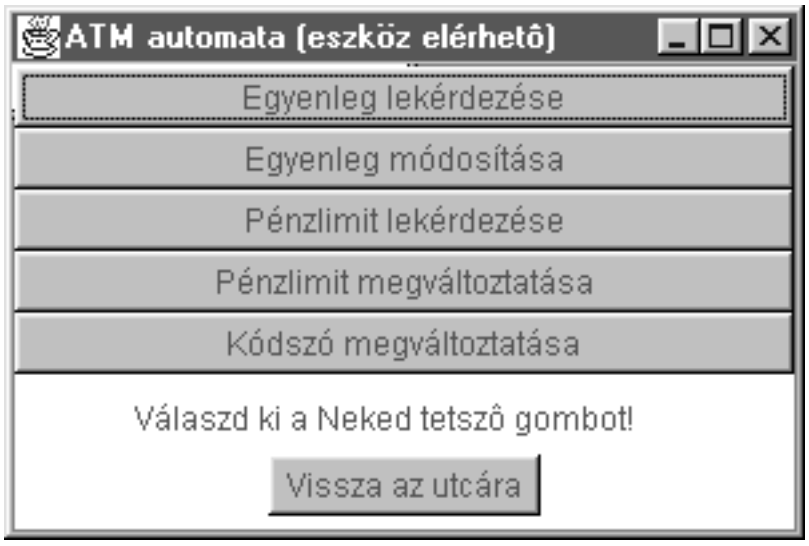
Utca



A program indításakor megjelenő ablak

A megjelenő ablak képviseli az utcát a való világban, ahol eldönthetjük – egyebek melletti –, hogy elmegyünk egy – a választott bankunkhoz tartozó – ATM automatához ('**ATM automata**'), vagy az egyetemi Jegyzetboltot látogatjuk meg ('**Jegyzetbolt**').

ATM automata



A banki ATM automata fő ablaka, csatlakoztatott eszköz esetén

A banki ATM automata fő ablaka, eszköz nélkül

Ez az ATM automata rendelkezik egy illesztőfelülettel, ahova csatlakoztatni kell az eszközünket (pl. JavaGyűrűt) az automata ,,felélesztéséhez" (a gombok felíratai visszkapják színüket /aktív állapot/). Az automata alapállapotban passzív helyzetben van (a gombok felíratai szürkék - kivéve a '**Vissza az utcára**' felíratút), és mindaddig így marad, míg olyan eszközt nem érintünk hozzá, melyen megtalálható az elektronikus-pénztárca alkalmazás, mellyel az automata képes kommunikálni.

Ekkor kiválaszthatjuk, hogy az alábbiak közül mit szeretnénk csinálni:

- Lekérdezzük az eszközön tárolt elektronikus pénz mennyiségét ('**Egyenleg lekérdezése**')
- Megváltoztatjuk pénzünk elosztását az eszköz egyenlege és a banki számla között ('**Egyenleg módosítása**')
- Megnézzük az eszközön beállított limit értékét ('**Pénzlimit lekérdezése**')
- Új limitet állítunk be az eszközön ('**Pénzlimit megváltoztatása**')
- Megváltoztatjuk a jelenlegi jelszavunkat egy másikra ('**Kódszó megváltoztatása**')
- Elhagyjuk az automatát, és visszamegyünk az utcára ('**Vissza az utcára**')

Az automata a fentiek kiválasztása után is erre az ablakra vált, ha az eszközt eltávolítják az érzékelőjétől, vagy nem ugyanazt az eszközt illesztik vissza. Természetesen az eredetileg behelyezett eszköz visszaillesztése után innen folytathatjuk a munkánkat (az eddigi nem megerősített változtatások elvesztésével).
(Az ebben a bekezdésben írtak igazak a fenti lehetőségek választásakor előjövő ablakokra is - kivéve a '**Utcá**'-t.)

Egyenleg lekérdezés (ATM automata)

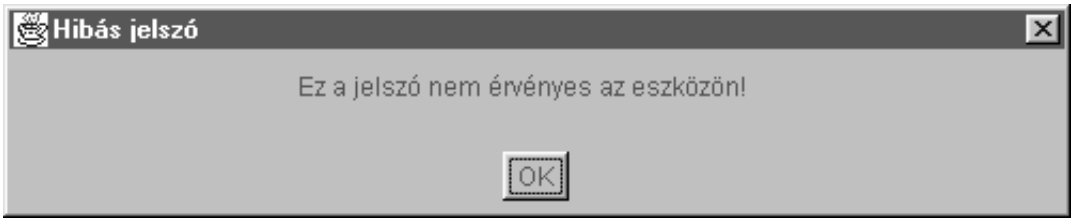
A képernyőn található ablakon láthatjuk az eszközön aktuálisan tárolt egyenlegünk mennyiségét. Ezután visszatérhetünk az ATM automata fő ablakához a '**Vissza az ATM automatához**' felíratú gomb megnyomásával.

Jelszó megadás

Itt adhatjuk meg az eszközön tárolt jelszavunkat

Ha az ATM automatánál az egyenleg vagy a limit módosítását választjuk, esetleg új jelszót szeretnénk beállítani, ill. a később tárgyalt 'Jegyzetvásárlás'-kor fizetni kívánunk a kiválasztott jegyzetekért, akkor egy új ablak lesz látható. Itt meg kell adnunk az eszközön tárolt jelszót, különben újra az automata fő ablakánál találjuk magunkat. A jelszót a billentyűzet kis-, ill. nagybetűi és a számok segítségével írhatjuk be.
A jelszó titkosságát jelzendő a leütött billentyűk helyén csak csillagot ('*****') láthatunk.
Az elrontott jelszót kitörölhetjük a '**Clear**' vagy '**Esc**' billentyű lenyomásával, vagy a '**törlés**' gombbal, ill. törölhetünk egy jelet a '**Backspace**' ('←') billentyűvel.
A helyesen begépelt jelszót az '**Enter**' vagy '**Return**' billentyűvel, ill. az '**OK**' gombbal fogadhatjuk el.
Ahhoz, hogy visszatérjünk az ATM automata fő képernyőjéhez (nem akarjuk beírni a jelszót) nyomjuk meg a '**Cancel**' billentyűt, ill. gombot.

Ha a megadott jelszó és az eszközön szereplő nem egyezik meg, akkor a következő hibaüzenetet kapjuk:



Innen az '**OK**' gomb megnyomásával térhetünk vissza az automata fő ablakához.

Egyenleg módosítás

Itt rendezhetjük át a pénzünket a banki számla és az eszközön futó virtuális pénztárca között /a '**+**' gomb növeli, a '-' csökkenti egyel az eszközön tárolt pénz mennyiségét, és természetesen ezzel együtt ellentétesen változtatja a számla pénzmennyiségét/. (A jelenlegi megvalósításban a számlán végtelen mennyiségű pénz áll rendelkezésünkre, melyet 5000-es egységekben kezelhetünk.)
Természetesen se a számlán, se az elektronikus pénztárcánkon tárolt pénz mennyisége nem csökkenhet 0 alá.
A módosításokat a '**Feltöltés**' gombbal tudjuk valóssá tenni.
A '**Vissza az ATM automatához**' gombbal visszatérhetünk az automata fő ablakához egyenlegváltoztatás nélkül.

Limit lekérdezés

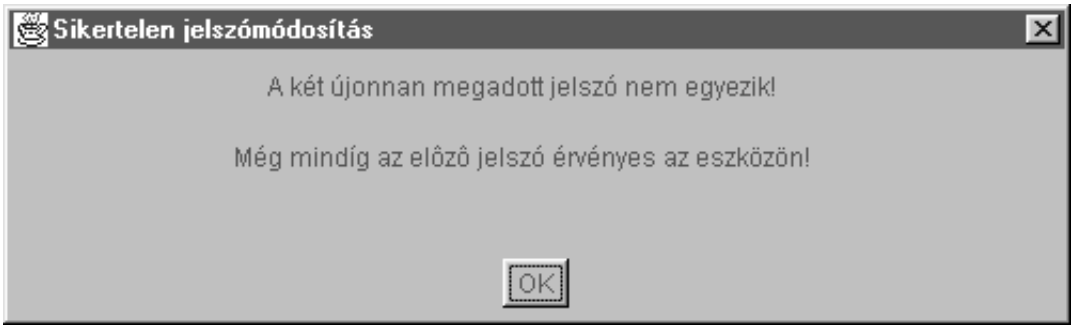
lásd: Egyenleg lekérdezés

Limit változtatás

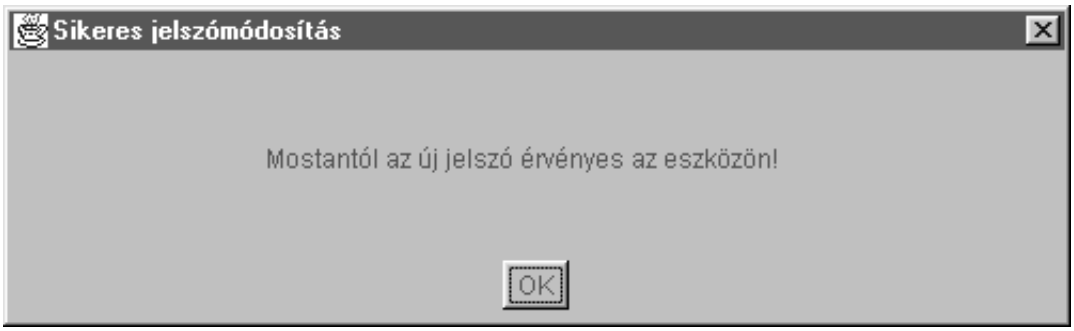
lásd: Egyenleg módosítás

Jelszó változtatás

Itt kétszer kell megadnunk az új jelszót (nehogy elgépeljük), melyek ha nem egyeznek akkor kapunk erről egy hibaüzenetet:



Ha a két megadott jelszó megegyezik, akkor az alábbi üzenet tájékoztat a sikeres módosításról:

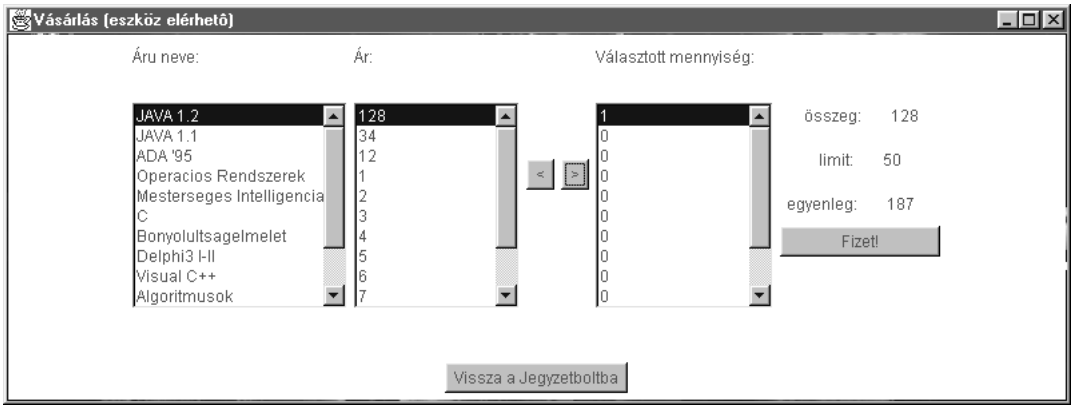


JegyzetBolt

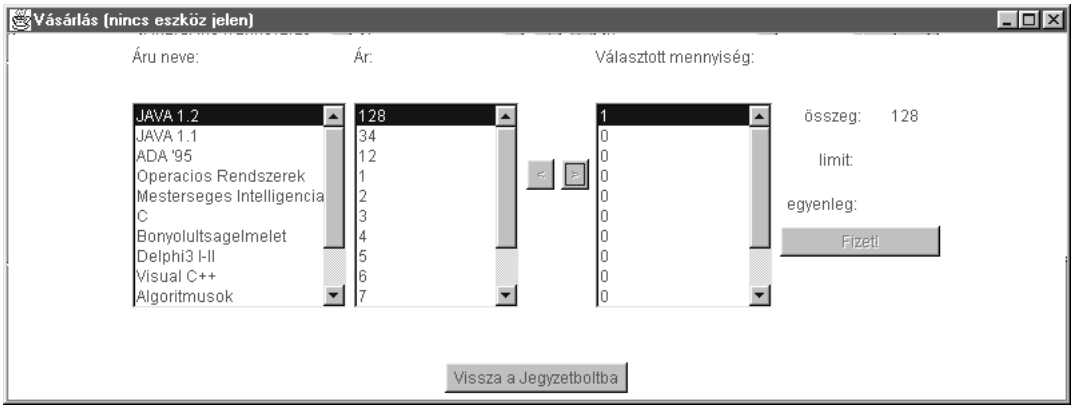
Egyenleg lekérdezés (Jegyzetbolt)

lásd: Egyenleg lekérdezés - ATM automata.

Jegyzet vásárlás



A jegyzetvásárlás ablak csatlakoztatott eszköz esetén



A jegyzetvásárlás ablak eszköz nélkül

Itt vásárolhatjuk meg a jegyzetboltban kapható könyveket. A megvásárolandó könyv kiválasztásához jelöljük ki a kívánt könyvet, majd nyomjuk meg a '>' gombot. Egy könyvből legfeljebb kilencet választhatunk ki. Ha a már kiválasztott könyv darabszámát szeretnénk csökkenteni, akkor azt a '<' gombbal tehetjük meg.

A könyvek darabszámának megfelelően változik azok árának összege (**összeg** címke), mely nem nőhet az eszközön tárolt pénz mennyisége fölé.

Ha az eszközt eltávoltítják az érzékelőtől, akkor nem lehet fizetni (és az egyenleg lekérdezéseknél megszokottakhoz hasonlóan az **egyenleg**, és a **limit** címke is üresre változik.)

Fizetéskor, ha a könyvek árának összege meghaladja az eszközön tárolt limit értékét, akkor csak a helyes felhasználói jelszó megadásával, egyébként minden kérdés nélkül[4] megy végbe a tranzakció. Ekkor az eszközön levő pénz mennyisége az összeggel csökken, és az automata lejegyzi a végrehajtott tranzakciót /log/.

[4]csak egy figyelmeztető ablakot fogunk látni

Rendszerleírás

Osztályok

`Szakdolgozat.class`

A ,,világ". Itt hoztam létre a dolgozat során használt főbb változókat (`eszkoz`, `bank`, ...) főleg azért, hogy azok kényelmesen elérhetők legyenek, és ne foglalják az őket hívó osztályok a memóriát.

Utca (`Valasztas.class`)

Az ,,utca" a Szakdolgozat osztályban megvalósított ,,világ"-ban. Ez a legelsőnek megjelenő ablak, és innen lehet elérni a banki ATM automatát, valamint a jegyzetboltot. Nem érzékeli a gyűrű jelenlétét, vagy annak hiányát, igazodva azon szemlélethez, hogy mindegy, hogy az utcán nálunk van-e a pénztárcánk vagy sem (legfeljebb majd a boltban nem tudunk venni semmit ...).

Banki ATM automata (`ATMAblak.class`)

Az ATM bankautomatát megvalósító osztály. Itt választható ki, hogy az automata mely szolgáltatásait kívánjuk használni:

- az eszközön tárolt egyenleget kívánjuk megnézni (`EgyenlegATMAblak.class`)
- pénzt akarunk mozgatni az eszköz és a bankban levő számla között (`EgyenlegModositasAblak.class`)
- az eszközhöz tartozó limit mennyiségét kívánjuk megnézni (`LimitATMAblak.class`)
- meg akarjuk változtatni az eszköz limitjét (`LimitModositasAblak.class`)
- új jelszót szeretnénk beállítani az eszközön (`JelszoModositasAblak.class`)

Egyenleg lekérdezése (`EgyenlegATMAblak.class`)

Megjeleníti az eszközön tárolt pénz mennyiségét

Egyenleg módosítása (`EgyenlegModositasAblak.class`)

A banki számla és az eszközön tárolt egyenleg közötti pénzmozgást teszi lehetővé (A dolgozatban a bankszámlán korlátlan mennyiségű pénz áll rendelkezésre 5000-es egységekben.)

Limit lekérdezése (`LimitATMAblak.class`)

Megjeleníti az eszközön levő limit értékét

Limit módosítása (`LimitModositasAblak.class`)

Az eszköz limit étékének változtatása

Jelszó módosítása (`JelszoModositasAblak.class`)

Az eszközhöz tartozó felhasználói jelszó módosítása. A program ügyel arra, hogy az új jelszó és annak megerősítése megegyezzen. A megengedett karakterek: kis- és nagybetűk, ill. számok.

Jegyzetbolt (`JegyzetBoltAblak.class`)

A Hallgatói jegyzetbolt. Lekérdezhetjük az egyenlegünk aktuális menynyiségét (`EgyenlegAblak.class`), és megtekinthetjük a bolt árukínálatát (`Vasarlas.class`).

Egyenleg lekérdezése (`EgyenlegAblak.class`)

lásd: `EgyenlegATMAblak.class`

Vásárlás (`VasarlasAblak.class`)

Itt lehet vásárolni az eszközön levő pénz erejéig.

Bővítési lehetőségek:

- A hallgatói kedvezmények támogatására a gyűrű egyedi azonosítója alapján lehetne eldönteni, hogy ki jogosult kedvezményes árú vásárlásra, és ki nem. Ezt a jegyzetbolt-automata tárolná egy külön adatbázisban.
- A vásárlásról készült lejegyzések /log/ minden adata külön adatbázisokba kerüljön, amiket csak egy adott kulcs birtokában lehet összeilleszteni egymással, így biztosítva méginkább a személyi jogok minél nagyobb védelmét. (Ezt a fajta bővítést lehetne alkalmazni a banki ATM-automatán is /Egyenleg, Limit, Jelszó módosítása/.)

Bank (`Bank.class`)

A Bankközpont. Az ATM automaták innen kapják a kért adatokat.

Eszköz ,,meghajtó" (`Eszköz.class`)

Ezen osztály metódusain keresztül lehet kommunikálni az eszközön levő elektronikus-pénztárca alkalmazással.

Elektronikus pénztárca (`EszkozApplet.class`)

Az eszközön futó program, mely az elektronikus fizetést valósítja meg.

Bővítési lehetőségek:

- Még egy - a `Limit`-hez hasonló változó bevezetése, mely azt a pénzértéket adná meg, ami fölött semmiképpen sem (még helyes felhasználói jelszó birtokában sem) lehetne fizetni, így biztosítva egy újabb védelmi lehetőséget az illegális eszközhasználat ellen.
- A kód optimalizálása a lehető legnagyobb mértékben, hogy az alkalmazás a lehető legkevesebb helyet foglalja az eszköz memóriájában, így támogatva a több alkalmazás (pl. beléptető-rendszer) egyidejű futását ugyanazon eszközön.
- stb...

Fontosabb változók az eszközön

`int Limit`

A `Limit` értékét nem meghaladó összegű vásárláskor a felhasználói jelszó megadása nem kötelező (A legnagyobb biztonságot 0 `Limit` értéknél érhetjük el, mivel ekkor minden egyenlegmódosításhoz szükség van jelszóra.)

`private int Egyenleg`

Az eszközön tárolt elektronikus pénz mennyisége. Módosítása csak jelszó megadásával történhet. Az eszköz ügyel, hogy mindig nemnegatív szám legyen.

`private byte[] PIN`

Az eszköz tulajdonosának jelszava. Változtatása csak a bank jelszavának megadásával történhet.

`private byte[] BankJelszo`

A bank jelszava. Használatával túlléphetünk a `PIN` korlátain, így használatára csak a bank jogosult.

`private short bankJelszoProbalkozas`

Számolja a helytelen `BankJelszo` hozzáférési kísérletet. A jelenlegi beállítások mellett nincs tévesztési lehetőség engedélyezve (azaz az első sikertelen kísérlet után letilt minden `BankJelszo`-t igenylő tranzakciót).

`private short userJelszoProbalkozas`

Számolja a helytelen `PIN` hozzáférési kísérletet. A jelenlegi beállítások mellett 2 tévesztési lehetőség engedélyezett (azaz a harmadik téves `PIN` megadása után letilt minden `PIN`-hez kötött tranzakciót. Ezt az állapotot csak a helyes `BankJelszo`-t igénylő `jelszoModositas(...)` metódus tudja helyreállítani. Az eddigi helytelen hozzáférések számát 0-ra csökkenthetjük, ha bármilyen tranzakcióban helyesen adjuk meg a felhasználói jelszót).

Fontosabb metódusok

`Eszkoz.class`

- `boolean eszkozJelen()`

  Visszaadja, hogy van-e gyűrű az olvasóhoz csatlakoztatva

- `boolean appletJelen()`

  Visszaadja, hogy a gyűrűn megtalálható-e az ,,`EszkozApplet`" alkalmazás

- `String getAdapterID()`

  Visszaadja `String` típusként azt a hexadecimális számot, ami az olvasóhoz tartozik

- `String getiButtonID()`

  Visszaadja azt - a gyártó által garantáltan egyedi - hexadecimális számot `String` típusként, ami az eszközhöz tartozik

- `int egyenlegKerdez()`

  Visszaadja a gyűrűn az `EszkozApplet` alkalmazás által tárolt egyenleget, vagy -1 -et, ha a lekérdezés során hiba történt

- `boolean egyenlegModosit(boolean bank, String Jelszo, int mennyivel)`

  Egy üzenetet küld az eszköznek az egyenleg módosítására (a `bank` logikai érték azt jelzi, hogy a `Jelszo` a felhasználó jelszava, vagy a banké, mivel pozitív mennyivel érték esetén csak banki jelszóval lehet elvégezni a szükséges módosításokat)

- `int limitKerdez()`

  Visszaadja a gyűrűn az `EszkozApplet` alkalmazás által tárolt limit értéket, vagy -1 -et, ha a lekérdezés során hiba történt

- `boolean limitModosit(String bankjelszo, int mennyire)`

  Egy üzenetet küld az eszköznek a limit módosítására (csak a bank jogosult a limit értékének módosítására). Visszatérési értéke a tranzakció sikeressége

- `boolean jelszoHasonlit(String Jelszo)`

  Visszaadja, hogy a `Jelszo` megegyezik-e az eszközön tárolt felhasználói jelszóval

- `boolean jelszoModosit(String bankjelszo, String ujJelszo)`

  Egy üzenetet küld az eszköznek a felhasználói jelszó módosítására (csak a bank jogosult a jelszó módosítására). Visszatérési értéke a tranzakció sikeressége

Biztonság

A dolgozat írásakor ügyeltem olyan alapvető biztonsági követelményekre, mint jelszó megadása után két gyűrű kicserélése (illetéktelen számlahasználat), vagy fizetés előtt az eszköz elvétele az olvasótól (szándékos hibaokozás), és a terminál↔gyűrű közötti kapcsolat kódoltsága (illetéktelenül hallgatózó harmadik személy), viszont csak érintőlegesen foglalkoztam a bank és a jegyzetbolt biztonságával (pl. tranzakciók lejegyzése /log/)

Jelenleg a felhasználói és a banki jelszó gyengén kódolva küldődik a tranzakciók során, ám célszerű kihasználni a két különböző kódolási algoritmust (`userKodol`, `bankKodol`) a felhasználói és a banki jelszó titkosításához, olyan algoritmust használni, mely nem statikus, hanem függ pl. az eszköz belső órájától (mivel azt az eszköz saját maga állítja) és a kódoló eljárásból nem számolható a dekódoló eljárás (= módosított RSA), hogy még egy jegyzetbolt-, vagy bankautomata tulajdonában se lehessen a jelszavakat illetéktelennek feltörni. Ehhez célszerű még minden eszközhöz különböző banki jelszó használata, melyet minden automata a bank központi egységétől kérdez meg. (Természetesen itt sem árt hasonlóan kódolni a tranzakciót.)

Az eszköz tárolja az utoló 5 jelszó-, egyenleg- vagy limitmódosítás típusú tranzakciót, melynek lényeges részeit vissza lehet kérdezni a megfelelő jogosultsággal.

Irodalomjegyzék

[1] Nyékiné G. Judit et al.: Java 1.1 útikalauz programozóknak

[2] Szakács Előd: Hallgató/oktató nyilvántartó rendszer

[3] Szatmáry Botond: Beléptető rendszer készítése előre meg nem határozott objektumra Java Kártya ill. Java Gyűrű felhasználásával

[4] http://www.iButton.com

[5] http://www.java.sun.com

[6] http://www.dalsemi.com

[7] ftp://ftp.javasoft.com/docs/javacard/