

Programdokumentáció

Tárgy: A programozás módszertana I.

Feladat: Projektmunka

Készítette: Debreceni Eszter

Bevezetés

A feladatom egy olyan adatnyilvántartó program készítése volt, amelynek az alapját egy XML típusú fájl képezi. Témaként én a cicapanziót választottam, mert a cicák nagyon közel állnak a szívemhez és nekem is van egy hét éves cicusom, aki rengeteget jelent a számomra.

Fejlesztőkörnyezet

A programot a saját laptopomon írtam meg, illetve teszteltem.

- 🖨 Modell: Asus VivoBook S15 S533EA
- 🖨 Processzor: 11. generációs Intel Core i5-1135G7, 2,4 GHz, 4 mag, 8 szál
- 🖨 Memória: 8 GB LPDDR4 RAM
- 🖨 Háttértár: 512 GB SSD
- 🖨 Videokártya: Intel Iris Xe Graphics
- 🖨 Operációs rendszer: Microsoft Windows 11 Home 22H2

Fejlesztőkörnyezetnek az IntelliJ IDEA Community Edition 2022.3.2-es verzióját választottam, mert számomra ez a legátláthatóbb, valamint leghatékonyabb IDE. A fejlesztői készlet, amit a programom megírásához használtam az Oracle OpenJDK 19.0.2-es verziója.

Adatszerkezet

A programomhoz kettő XML fájlt hoztam létre, egyet a cicák adatainak, egyet pedig a gazdájuk adatainak a tárolására.

A [cats.xml](#) fájl tartalmazza az éppen a panzióban tartózkodó cicákat („cat”) az adataikkal együtt. Minden cicához tartozik egy név („name”), egy fajta („breed”), egy életkor („age”) és egy gazda („owner”).

Az [owners.xml](#) fájl tartalmazza az összes olyan gazdát („owner”), akiknek a cicája jelenleg a panzióban tartózkodik vagy régebben tartózkodott már ott. Itt minden gazdának el van tárolva a neve („name”), a születési ideje („birthdate”), a telefonszáma („phone”) és természetesen a cicájának a neve („cat”) is.

A közös tulajdonságok a két XML fájlban a cica neve és a gazda neve, ezek alapján tudjuk, majd „összekötni” a cicát a gazdájával.

Metódusok

A programomban létrehoztam egy „Cat” nevű osztályt, amelynek az osztály tagváltozói azonosak a [cats.xml](#) fájlban nyilvántartott tulajdonságokkal, illetve létrehoztam egy „Owner”

nevű osztályt is, amelynek az osztály tagváltozói pedig az [owners.xml](#) fájlban található tulajdonságoknak feleltethetők meg.

A „Cat” osztályban található metódusok:

Beolvasás-kilistázás - [getCatsFromXml](#):

Ez a metódus egy „Cat” típusú listát ad vissza értékként, amelyet úgy tölt fel, hogy beolvassa a [cats.xml](#) fájlt, majd annak elemeit Cat típusba rakja, amit hozzáad a listához.

Konzolra kiírás - [printCatsFromXml](#):

Ez a metódus a [getCatsFromXml](#) meghívásával létrehoz egy „Cat” típusú listát, majd a lista elemeit, azaz a cicákat és az adataikat kiírja tagoltan a konzolra.

Új elem hozzáadás - [addNewCatAndOwner](#):

Természetesen, mint minden panzió ez is rendelkezik egy bizonyos férőhellyel. Először a metódus megvizsgálja, hogy jelenleg hány cica van bejelentkezve a panzióba, ha 20 vagy annál több, akkor hibaüzenetet ír ki a program. Hogyha kevesebb akkor először bekéri az új cica adatait, kivéve a gazdája nevét és a cica neve alapján megvizsgálja, hogy az [owners.xml](#) fájlban, valamelyik gazdi rendelkezik-e ilyen nevű cicával. Ha igen, akkor kiírja a gazda nevét és rákérdez, hogy ehhez a gazdihoz tartozik-e a cica. Ha több gazdinál is egyezik a cica neve, akkor rákérdez az összes gazdára, amíg a felhasználó meg nem találja a keresett gazdát. Ha igen a válasz („i”), akkor ezt a nevet menti el az újonnan becsekkolt cica gazdájához és kiírja, hogy a gazdi már szerepel a rendszerben, ezért nincs több teendő és beírja az új cicát a [cats.xml](#) fájlba. Ha nem a válasz („n”) és nem szerepel több gazdinál a cica, akkor bekéri a gazdája adatait is (kivéve a cica nevét, mert azt már egyszer bekérte) és beírja ugyan úgy az új cicát a [cats.xml](#) fájlba, utána pedig az új gazdit is az [owners.xml](#) fájlba. Ha sikeresen lefutott a metódus, akkor a program kiírja a sikeres becsekkolást.

Elem törlés - [deleteCatFromXml](#):

Ez a metódus először bekéri annak a cicának a nevét, aki szeretne kicsekkolni, utána megszámolja, hogy hány cica van becsekkolva jelenleg ezen a néven a panzióba. Ha egy sincs, akkor hibaüzenetet ír ki a program és leáll. Ha egy találat van, akkor megvan a keresett cicánk. Ha pedig egynél több akkor bekéri a gazdája nevét is, hogy megtalálja a pontos cicát és amennyiben nincs ilyen nevű cica a megadott gazdival, akkor szintén hibát ír ki a program aztán leáll viszont, ha van, akkor megtaláltuk a keresett cicát. Ezután kitörli a cicát a [cats.xml](#) fájlból és kiírja a sikeres kicsekkolást.

Elem módosítás - [updateCatAndOwner](#):

Ez a metódus ugyan úgy keresi meg a módosítandó cicát, mint a [deleteCatFromXml](#), először bekéri a cica nevét, megvizsgálja és a találatok alapján pontosan úgy jár el, mint az utóbb említett metódusnál. Ha megtalálja a keresett cicát, akkor rákérdez az összes adatról, hogy szeretnék-e módosítani. Ha igen („i”), akkor bekéri az új adatot és kicseréli, ha pedig nem („n”), akkor ugrik a következő adatra. Amikor elér a gazdához előtte rákérdez, hogy egyáltalán a gazda adatait szeretnék-e módosítani, ha igen akkor ott is végig megy és kicseréli a kívánt adatokat. Ha a felhasználó módosítani szeretné a gazda nevét, az mind a két XML fájlban módosulni fog. Amennyiben történt módosítás a cica nevében, azt nem csak a [cats.xml](#), de az

[owners.xml](#) fájlban is ki cseréli a program. Ha minden kívánt adatot sikerült kicserélni, akkor a program kiírja a sikeres adat módosítást.

Cica számláló – catCounter:

Ez a metódus beolvassa a [cats.xml](#) fájlt, majd kiírja, hogy hány cica van becsekkolva jelenleg.

XML széttagoló – prettyPrintCatsXml:

A metódus meghívásával szét tudjuk tagolni a [cats.xml](#) fájlt, hogy átláthatóbb legyen.

Az „Owner” osztályban található metódusok:

Beolvasás-kilistázás – getOwnersFromXml:

Majdnem teljesen azonos a [getCatsFromXml](#) metódussal, viszont az [owners.xml](#) fájlt olvassa be, illetve Owner típusú listát ad vissza a gazdák adataival.

Konzolra kiírás – printOwnersFromXml:

Meghívja a [getOwnersFromXml](#) metódust és kiírja tagoltan a gazdákat az adataikkal a konzolra.




XML széttagoló – prettyPrintOwnersXml:

A metódus meghívásával szét tudjuk tagolni az [owners.xml](#) fájlt, hogy átláthatóbb legyen.

XML feldolgozás elméleti alapjai

DocumentBuilderFactory és DocumentBuilder segítségével kinyerjük a DOM-ot az XML fájlból. A DOM-ból kiszedjük az elemeket egy NodeList-be, így azokat egyenként tudjuk vizsgálni. Az egyes elemek szöveges értéket tartalmazó Node-jainak tartalmát kinyerve végezzük el a vizsgálatokat. Miután a megfelelő műveleteket elvégeztük TransformerFactory és Transformer, valamint StreamResult és DOMSource segítségével írjuk vissza az eredményt az XML fájlba.

A program írása során felhasznált források:

-  https://www.w3schools.com/java/java_date.asp
-  <https://mkyong.com/java/how-to-modify-xml-file-in-java-dom-parser/>
-  <https://www.tabnine.com/code/java/methods/javax.xml.transform.Transformer/setOutputProperty>