ELSŐ FEJEZET
A Java nyelv bemutatása

Mielőtt új programozási nyelv elsajátításába kezdünk, érdemes tisztában lennünk azzal, hogy milyen alapvető sajátosságokkal rendelkezik, és milyen célokra használható. A fejezet célja, hogy röviden ismertesse a Java nyelv legfontosabb ismérveit és gyakori alkalmazási területeit, és ezzel kedvcsinálóként is szolgáljon az Olvasó számára.

1.1. A Java nyelv jellemzői

A Java *általános célú* programozási nyelv. Ez azt jelenti, hogy a nyelv utasításai és a könyvtári komponensek bármilyen algoritmikusan megoldható problémához jól használhatók. A Java ezen kívül *objektumorientált,* azaz az adatokat és a rajtuk végrehajtandó műveleteket osztályok és objektumok segítségével egységbe zárja. A nyelv *típusrendszere statikus és erős,* tehát a változódeklarációknak köszönhetően már a programok lefordításakor könnyen ellenőrizhető a típusbiztosság.

A Java nyelv legjellemzőbb tulajdonsága, hogy a lefordított kódot nem közvetlenül az operációs rendszer, hanem egy *futtatókörnyezet* futtatja. Ezt a speciális programot *virtuális gépnek (virtual machine)* nevezzük. Ez a fogalom különbözik a hétköznapi értelemben használt virtuális géptől. Ez ugyanis nem egy emulációs szoftver, amelyben teljes operációs rendszert futtathatunk, hanem egy szoftverréteg, amely a hordozhatóságot és a biztonságos futást valósítja meg. A Java virtuális gép saját gépi utasításokkal rendelkezik. A Java fordító a Java-programokat nem a célplatform, hanem a virtuális gép utasításkészletére fordítja le. A program futtatásakor a virtuális gép képezi le ezeket az utasításokat a tényleges hardveres utasításkészletre. Ebből következik a programok *hordozhatósága.* Ahhoz, hogy a Java nyelvű programokat más platformon is futtatni tudjuk, csupán a virtuális gép átültetésére van szükség. A Java készítője, az Oracle jelenleg Linux, Windows és Solaris operációs rendszerekhez kínál virtuálisgép-megvalósítást. Mindhárom operációs rendszeren elérhető virtuális gép a szokásos x86-os és x64-es PC-architektúrákhoz, illetve Solaris rendszeren SPARC számítógépeket is támogat az Oracle. A Java-programok hordozhatóságát ezért a *Write Once, Run Anywhere (Egyszer megírni, bárhol futtatni!)* szlogennel szokták jellemezni.

A virtuális gép használatának másik előnye, hogy a programok nem közvetlen a futtató számítógépen hajtódnak végre, ezért nehezebben tudnak kárt okozni. A virtuális gép több különféle védelmi mechanizmust támogat. Ezek segítenek a támadások elleni védekezésben. Mivel a virtuális gép kezeli a futtatandó kódot, az ilyen elven működő programokra sokszor a *felügyelt kód (managed code)* kifejezéssel hivatkoznak.

A Java nyelvre jellemző még az elérhető *osztálykönyvtárak (class library)* széles tárháza. Már az alapértelmezésben feltelepülő könyvtári komponensek is sok problémára kínálnak megoldást, de sok más, nyílt forrású és ingyenesen használható osztálykönyvtár is létezik. Ezek számos gyakori problémára nyújtanak kész meg-

oldást, tehát jelentősen megkönnyítik és felgyorsítják új, összetett alkalmazások kifejlesztését. A magas szintű osztálykönyvtárak és a Java átgondolt kialakítása elősegítik a stabil és hibamentes programok írását. Sokszor ez a szempont fontosabb, mint a program puszta gyorsasága, bár a Java virtuális gép jól finomhangolható, ezért a kész program általában megfelelő teljesítményt nyújt.

1.2. A Java nyelv felhasználási területei

Ugyan a Java általános célú programozási nyelv, mégis kiemelhetünk néhány tipikus, gyakori felhasználási területet. Ehhez először a Java nyelv három változatát tekintjük át. Az általános alkalmazásoknál a *Java Standard Edition* (SE) változatot használjuk. Ez magában foglalja a virtuális gépet és az alapvető funkcionalitásokat megvalósító osztálykönyvtárat. A könyv ezt a változatot ismerteti, a másik kettőről csak röviden lesz szó.

A *Java Enterprise Edition (EE)* a Java SE olyan kibővítése, amely a háromrétegű architektúra kialakítását támogatja. A Java EE speciális komponenseit az alkalmazásszerver kezeli. Ez a Java virtuális gépre épülő, annál több szolgáltatást kínáló futtatókörnyezet. A legalsó réteget a szabadon választott, Java-környezettől független adatbázisszerver képviseli. Erre épül az üzletilogika-réteg, amely megvalósítja az adatbázison elvégezhető műveleteket, és ezeket a legfelső, megjelenítési réteg számára elosztott Java-objektumokon keresztül elérhetővé teszi. Az elosztott objektumokat a Java EE által támogatott *Enterprise JavaBeans (EJB)* szabvány segítségével valósíthatjuk meg. Az alkalmazásszerver az elosztott üzleti komponensek számára middleware-szolgáltatásokat nyújt. A middleware-szolgáltatások az üzleti alkalmazásokban gyakran felbukkanó problémákat oldják meg, ilyen probléma például az adatok perzisztenciája, az aszinkron üzenetkezelés és a munkafolyamatok kezelése. A megjelenítési réteg lehet egy asztali Java SE-alkalmazás vagy a Java EE webes technológiáival megvalósított vékonykliens. A *Servlet* technológia HTTP-kéréseket tud programból kezelni. Ez a gyakorlatban azt jelenti, hogy a felhasználó a böngészőn keresztül lekér egy weboldalt, amelyre a választ a meghívott szervlet állítja elő. A böngészőben visszaadott HTML-oldal azonban nem írható le könnyen Java-kóddal. A Java EE ezért további szabványokat is bevezetett, ilyen a *JavaServer Pages (JSP),* a *Facelets* és a *JavaServer Faces (JSF).* Ezek közelebb viszik a fejlesztést a HTML-programozáshoz, és a háttérben a Servlet technológiára épülnek. A Java EE magában foglal még más technológiákat is, itt csak a legfontosabbakat említettük. A Java EE változatot [1] mutatja be részletesen.

A harmadik Java-változat a *Java Micro Edition (ME).* Ez mobiltelefonokon és egyéb mobil eszközökön használható. Míg a Java SE a Java EE részhalmaza, a Java ME alapvetően különbözik a Java SE-től. A Java ME korlátozottsága miatt manapság már kevésbé használatos. Megemlítjük azonban, hogy az Android operációs rendszerrel rendelkező eszközök programozási nyelve a Javán alapul, de valójában annak egy módosított változata. Az Android által használt Dalvik virtuális gép eltér a szabványos Java virtuális géptől, ezért ezeken a rendszereken a szabványos Java bájtkód nem futtatható. Az Android programozásáról [2] kínál bővebb információt.

1.3. A Java SE-alkalmazások típusai

A három közül a Java SE változata szolgál az általános alkalmazások kifejlesztésére, de ennek segítségével is többféle alkalmazást készíthetünk. Először a normál *asztali alkalmazásokat* érdemes megemlíteni, amelyek ugyanúgy a saját gépünkön futnak, mint a többi felhasználói program. Általában grafikus felhasználói felülettel rendelkeznek. Idetartoznak a szövegszerkesztő programok, a programozási fejlesztőkörnyezetek, a torrentkliensek stb. De olyan program is készült már Javában, amellyel a Nap aktivitását tanulmányozhatjuk.[1] Az is elképzelhető, hogy az alkalmazás nem rendelkezik grafikus felhasználói felülettel, hanem parancssorból futtatható. Ilyenek lehetnek például konverziós programok vagy egyszerű segédprogramok, amelyek nem igényelnek bonyolult felhasználói interakciót. A Java SE-vel együtt települő keytool segédprogram is ilyen. Programok aláírásához használt kulcsokat kezelhetünk vele.

A Java SE-alkalmazások speciális fajtája a *Java-applet*. A Java-applet a böngésző által indított virtuális gépen futó webes alkalmazás. Ehhez a böngészőnek egy kiegészítésre, a Java-pluginra van szüksége. Mivel az applet kódja a Webről érkezik, ezért a Java-plugin ezt alapértelmezésben biztonsági korlátozásokkal futtatja, például tiltja a fájlműveleteket és a távoli géphez való kapcsolódást (kivéve azt a webszervert, amelyről az applet érkezett). A korlátozások miatt a normál Java-appletekkel csak korlátozottabb feladatokat lehet megvalósítani. Ha olyan műveletet szeretnénk elvégeztetni az applettel, amely alapértelmezésben tilos, akkor az appletet digitális aláírással kell ellátni. A hiteles aláírás azt jelzi, hogy az applet készítője vállalja a személyazonosságát, ezért az ilyen appletben biztonsági szempontból megbízhatunk. A Java-plugin az aláírt appleteket biztonsági korlátozások nélkül futtatja. Ha az aláírást olyan kulccsal végezték, amelyet tanúsító hatóság nem hitelesített, akkor nincs garancia a készítő személyazonosságára. Ilyenkor a Java-plugin felugró ablakban kéri a felhasználót az aláírás elfogadására vagy elutasítására.

Az appletek kezdetben többre voltak képesek, mint a natív webes technológiák. Manapság azonban sok webalkalmazás JavaScript- és AJAX-technológiákkal is gazdag funkcionalitást valósít meg. Ezeket a technológiákat a böngésző natívan támogatja, ezért nincs szükség kiegészítő plugin telepítésére, sem digitális aláírásra. Az appletek tehát mára visszaszorultak. Térvesztésük másik oka a *Java WebStart* alkalmazások megjelenése. Az appleteknél szintén nehézséget jelent, hogy ezek implementációs osztályára is megkötések vonatkoznak. A fejlesztésnél az `Applet` osztály leszármazott osztályát kell létrehoznunk, tehát az asztali alkalmazásokat nem tudjuk közvetlenül appletté alakítani. A WebStart technológia ezzel szemben lehetővé teszi, hogy általános Java SE-alkalmazásokat indítsunk el böngészőből. A hálózati indítást a Java Network Launching protokoll (JNLP) valósítja meg, ez `.jnlp` kiterjesztésű fájlból olvassa be a konfigurációs adatokat. A konfigurációs fájl tartalmazza az alkalmazás rövid leírását, a gyártó nevét, a szükséges Java SE-környezet verziószámát, a futtatáshoz szükséges Java-csomagot vagy csomagokat, illetve az alkalmazás belépési pontját. Szintén engedélyezhetjük az alkalmazás automatikus frissítését. A WebStart-alkalmazás ugyanis első futtatáskor települ a számítógépre, de a kapcsolódó adatok, mint az alkalmazás kódjának letöltési helye, eltárolódnak a számítógépen. Ezért, ha az automatikus frissítés engedélyezve van, a WebStart futtatókörnyezet az alkalmazás

[1] JHelioViewer Project: `http://jhelioviewer.org/`

indításakor képes a frissítéseket megtalálni és letölteni. A WebStart-alkalmazások ezeknek a mechanizmusoknak köszönhetően könnyebben kifejleszthetők és használhatók, mint az appletek, ugyanakkor a biztonságra vonatkozó megkötések rájuk is érvényesek. A megkötések feloldásában itt is a digitális aláírás segíthet. Jelenleg a magyarországi elektronikus adóbevalláshoz használható nyomtatványkitöltő program is Java-alkalmazás, amelyet a Nemzeti Adó- és Vámhivatal honlapjáról a WebStart technológiával érhetünk el. A 16.4. alfejezetben bemutatjuk, hogyan tehetjük elérhetővé saját Java-alkalmazásainkat a WebStarton keresztül.

1.4. A Java verziói

A Java nyelvet folyamatosan fejlesztik, hogy lépést tartson az újabb programozási trendekkel és ipari elvárásokkal. A Java SE 7-es, aktuális verziója 2011 júliusában jelent meg. A könyv ezt a változatot mutatja be, de a verziók közti eligazodást segítendő, röviden összefoglaljuk azok számozási rendszerét.

A Java-verziók számozása 1.0-tól indult. Az 1.2-es verzió olyan sok újdonságot hozott, hogy utólag 2-es verziónak nevezték át, ugyanakkor az 1.2-es verziószám is használatban maradt. Ennél a verziónál vált szét a Java a három változatra, és a 2-es verziószám miatt ezekre a *J2SE, J2EE* és *J2ME* rövidítésekkel is hivatkoztak. Ezt követően J2SE 1.3-ról és J2SE 1.4-ről beszélünk, bármilyen meglepő is, hogy egyszerre jelen van a 2-es és az 1.3-as és 1.4-es verziószám. A következő verzió ugyanakkor J2SE 5.0 lett, tükrözve a bevezetett újítások jelentőségét, meghagyva viszont a verziószámok közti zavart. Végül rendezték a verziószámok kétértelműségét, és a következő verziót már Java SE 6 névvel adták ki, majd ezt követte a Java SE 7. Ennek ellenére néhol, például a telepítési mappák nevében, az 1.6 és 1.7 verziószámokkal is találkozhatunk, de ezek az imént említett két verziót jelölik.

A Java EE számozása hasonlóan történik, azaz sokáig a J2EE rövidítést használták a mögé írt, megtévesztő verziószámmal, majd a Java EE 6 megjelenésével rendeződött a helyzet. A Java EE megjelenése mindig az azonos verziójú SE változatot követi jelentős eltéréssel. Míg a Java SE 7 az írás pillanatakor már két éve elérhető, a Java EE 7 éppen csak megjelent.

Fel kell még hívnunk a figyelmet arra, hogy minden Java SE-változat két disztribúcióban érhető el. A *Java Runtime Environment (JRE)* csupán a lefordított Java-programok használatához szükséges futtatókörnyezetet és a lefordított osztálykönyvtárakat tartalmazza, a fejlesztéshez szükséges eszközöket nem. Fejlesztéshez a *Java Development Kit (JDK)* telepítése szükséges, amely a futtatókörnyezeten kívül tartalmazza a fordítót és más fejlesztői segédeszközöket is.

1.5. Termék és szabvány

A Java nyelvet a Sun Microsystems vállalat hozta létre, amelyet később az Oracle felvásárolt. Jelenleg a Java nyelvet tehát az Oracle gondozza. Ő fejleszti és adja ki a nyelv újabb verzióit, a Java nyelvhez terméktámogatást nyújt, tanfolyamokat szervez, illetve vizsgákkal megszerezhető fejlesztői minősítéseket ad. A Java nyelv tehát az Oracle kereskedelmi terméke.

A Java nyelv ugyanakkor szabvány is. A nyelv és az osztálykönyvtár, valamint a virtuális gép specifikációja is teljesen nyilvános, ráadásul ezek nagy része nyílt forrású szoftver is. A kiegészítő keretrendszerek is szabványként vannak specifikálva, és azokat nyílt, közösségi szabványosítási folyamat során dolgozzák ki. Ez a *Java Community Process (JCP)*. Az egyes szabványokat a *Java Specification Request (JSR)* számukkal azonosítjuk. Például a JPA 2.1 szabvány száma JSR 338. A szabványosításnak köszönhetően a Java nyelvből vagy annak részeiből bárki készíthet saját megvalósítást. A hivatalos kiadáson kívül nincs teljes megvalósítás, de egyes Java-szabványokhoz több implementáció is létezik, ilyen például a Java Persistence API (JPA) szabvány (lásd 8.2. alfejezet). Ezek között az alternatív keretrendszerek között is nagyrészt nyílt forrású szoftvereket találunk. A szabványosítás lehetővé teszi, hogy a nyelv egyes részeit alternatív megvalósításokkal lecseréljünk. Ezek a nem szabványosított pontokban el is térhetnek, illetve a szabványon felül extra funkcionalitást is biztosíthatnak. A programozónak ezért lehetőséget adnak a választásra, és a fejlesztett terméket nem teszik függővé egy adott gyártótól.