A FEJEZET TARTALMÁBÓL:

- Bemutatjuk a Helló, világ! programot
- Megismerjük a Java-programok alapelemeit
- Megjegyzéseket adunk a programokhoz
- Elsajátítjuk az objektumorientált programozás alapvető ismereteit
- Megismerjük az osztályok importálásának módszereit

3. fejezet

Bevezetés a Javába

Ebben a fejezetben a Java-programok írásának legalapvetőbb elemeit találod. Az itt látható programok nagyon fapadosak: egyszerű információkat jelenítenek meg egy konzolon vagy egy windowsos parancsablakban. Még néhány fejezetet fel kell dolgoznod, mielőtt olyan programokat írsz, amelyek bármi érdemlegeset csinálnak. De az ebben a fejezetben látott egyszerű programok elegendőek a Java-programok alapvető felépítésének szemléltetésére.

Figyelmeztetünk, hogy ebben a fejezetben számos olyan Java-programozási funkciót mutatunk be, amelyeket a későbbi fejezetekben részletesebben is ismertetünk. Láthatsz például néhány változódeklarációt, metódust, sőt még `if` utasítást és `for` ciklust is. Ennek a fejezetnek nem az a célja, hogy azonnali jártasságra tegyél szert ezekkel a programozási elemekkel, csupán az, hogy megismertessen velük.

A híres Helló, világ! program

Sok programozási könyv egy egyszerű példaprogrammal kezdődik, amely a `Helló, világ!` szöveget jeleníti meg a kimenetben. A 3.1. kód egy ilyen Java-programot mutat be.

3.1. KÓD **A Helló, világ! program**

```
public class HelloApp                                    →1
{                                                        →2
     public static void main(String[] args)              →3
     {                                                   →4
          System.out.println("Helló, világ!");           →5
     }                                                   →6
}                                                        →7
```

A fejezet későbbi részében részletesen megismerheted a program minden elemét, de előbb szóról szóra végigvezetünk rajta.

Az 1. és 2. sor egy nyilvános osztály deklarációját jelöli, melynek neve `HelloApp`:

→1 **`public`**: A Java nyelv egyik *kulcsszava*. Azt jelzi, hogy a következő elem elérhető más Java-elemek számára. Ebben az esetben a következő elem egy `HelloApp` nevű osztály. Tehát ez a kulcsszó azt jelzi, hogy a `HelloApp` osztály egy *nyilvános osztály*, ami azt jelenti, hogy más osztályok is használhatják.

`class`: Egy másik Java-kulcsszó. Azt jelzi, hogy az itt definiált elem egy *osztály*. Minden Java-program egy vagy több osztályból áll. Egy osztálydefiníció olyan kódot tartalmaz, amely meghatározza a program által létrehozott és használt objektumok viselkedését. Bár a legtöbb valós program egynél több osztályból áll, az ebben a fejezetben bemutatott egyszerű programok csak egy osztályt tartalmaznak.

`HelloApp`: Az itt definiált osztályt azonosító *név*. Míg a kulcsszavak – mint például a `public` és a `class` – a Java nyelv által meghatározott szavak, a nevek olyan szavak, amelyeket a programban használt különböző elemek azonosítására hozunk létre. Ebben az esetben a `HelloApp` név az itt definiálandó nyilvános osztályt azonosítja. (Bár a *név* a technikailag helyes kifejezés, néha a neveket *azonosítóknak* is mondjuk. A név egyfajta azonosító, de nem minden azonosító név).

→2 **`{`**: A 2. sorban lévő nyitó kapcsos zárójel az osztály *törzsének* (body) kezdetét jelöli. A törzs végét a 7. sorban lévő kapcsos zárójel jelzi. Minden, ami ezeken a zárójeleken belül megjelenik, az osztályhoz tartozik. Ahogy a Javával dolgozol, tapasztalni fogod majd, hogy milyen gyakran használod ezeket a zárójeleket.

A 3–6. sorok a `HelloApp` osztály `main` metódusát definiálják:

→3 **`public`**: Ismét ezt a kulcsszót használjuk, ezúttal annak jelzésére, hogy az itt definiált metódusnak nyilvános hozzáféréssel kell rendelkeznie. Ez azt jelenti, hogy a `HelloApp` osztályon kívül más osztályok is használhatják. Minden Java-programnak rendelkeznie kell egy olyan osztállyal, amely definiál egy `main` nevű nyilvános metódust. A `main` metódus tartalmazza azokat az utasításokat, amelyek a program futásakor végrehajtódnak.

`static`: A Java nyelv megköveteli a `static` kulcsszó megadását a `main` metódus definiálásakor.

`void`: A Javában a metódus olyan kódegység, amely képes értéket kiszámítani és visszaadni. Létrehozhatsz például egy olyan metódust, amely kiszámítja az eladások végösszegét. Ekkor ez az összeg a metódus visszatérési értéke. Ha egy metódusnak nem kell értéket visszaadnia, akkor ezt a `void` kulcsszóval kell jelezned. Mivel a Java megköveteli, hogy a `main` metódus ne adjon vissza értéket, annak definiálásakor meg kell adnod a `void` kifejezést.

`main`: Végül itt van az az *azonosító*, amely a metódus nevét adja. Mint már említettük, a Java megköveteli, hogy a metódus neve `main` legyen. A `main` metóduson kívül további metódusokat hozhatsz létre bármilyen névvel. A metódusok létrehozásának módját a 8. fejezetben ismerheted meg. Addig a programjaink csak egy `main` nevű metódusból állnak.

`(String[] args)`: Hát, ez a Java-elem túlságosan fejlett ahhoz, hogy most el tudjuk magyarázni. *Paraméterlistának* hívják, és arra való, hogy adatokat adjunk át egy metódusnak. A Java megköveteli, hogy a `main` metódus egyetlen paramétert kapjon, amely egy `String` objektumokból álló tömb. Konvenció szerint ezt a paramétert `args`-nak nevezzük. Ha nem tudod, mi az a paraméter, a String vagy a tömb, ne aggódj emiatt. A Stringről a következő fejezetben, a paraméterekről a 8. fejezetben, a tömbökről pedig a 12. fejezetben olvashatsz. Addig is vedd észre, hogy minden programban a `main` metódusok definiálásánál beírjuk, hogy (`String[] args`).

→4 **`{`**: A 4. sorban egy újabb kapcsos zárójelpár kezdődik és a 6. sorban végződik. Ezek a zárójelek a `main` metódus törzsét jelölik. Figyeld meg, hogy a 6. sorban lévő záró zárójel a 4. sorban lévő nyitó zárójelhez van párosítva, míg a 7. sorban lévő zárójel a 2. sorban lévő zárójelhez tartozik. Ez a fajta párosítás a Javában általános. Röviden, amikor a Java egy záró zárójelhez ér, akkor azt a legutolsó olyan nyitó zárójelhez párosítja, amelyet még nem zártunk be – vagyis amely még nem volt párosítva más záró zárójellel.

→5 **`System.out.println("Helló, világ!");`**: Ez az egyetlen utasítás az egész programban. Egy `println` nevű metódust hív meg, amely a `System.out` objektumhoz tartozik. A `println` metódus egy sornyi szöveget jelenít meg a konzolon. A megjelenítendő szöveget a `println` metódusnak paraméterként adjuk át a `println` szót követő kerek zárójelek között. Ebben az esetben a szöveg az idézőjelek

közé zárt `Helló, világ!` karakterlánc. Ennek eredményeképpen ez az utasítás a konzolon a `Helló, világ!` szöveget jeleníti meg.

Jegyezd meg, hogy a Javában a legtöbb (de nem minden) utasításnak pontosvesszővel kell végződnie. Mivel ez az egyetlen utasítás a programban, ez a sor az egyetlen, amelyet pontosvessző zár.

`→6` `}`: A 6. sorban lévő záró kapcsos zárójel zárja a `main` metódus törzsét, amelyet a 4. sorban lévő zárójel nyitott meg.

`→7` `}`: A 7. sorban lévő záró kapcsos zárójel zárja a `HelloApp` osztály törzsét, amelyet a 2. sorban lévő zárójel nyitott meg. Mivel ez a program csak egy osztályból áll, ez a sor egyben a program végét is jelzi.

A program futtatásához először egy szövegszerkesztővel, például a Notepad++ vagy a TextPad segítségével meg kell írni a programot – pontosan úgy, ahogy a 3.1. kódban szerepel – egy `HelloApp.java` nevű szövegfájlban. Ezután le kell fordítani a következő parancs futtatásával a parancssorban (lásd a 13. fejezetet):

```
javac HelloApp.java
```

Ez a parancs létrehoz egy `HelloApp.class` nevű osztályfájlt, amely tartalmazza a `HelloApp` osztályhoz lefordított Java-bájtkódot.

A programot a következő parancs megadásával futtathatod:

```
java HelloApp
```

Most, hogy már láttad, hogyan is néz ki egy Java-program, jobban megértheted, pontosan mit is csinál ez a parancs. Először is betölti a Java Virtual Machine-t (JVM) a memóriába. Ezután megkeresi a `HelloApp` osztályt a `HelloApp.class` nevű fájlban, majd lefuttatja a `HelloApp` osztály `main` metódusát, amely megjeleníti a `Helló, világ!` üzenetet a konzolon.