JBuilder kezdőknek

Angster Erzsébet: Objektumorientált tervezés és programozás Java, 1. kötet

Kivonat a 9. fejezetből, 2003. január.

Ez a rövid útmutató kezdők részére készült. Megmutatja, mi a legegyszerűbb módja az **AE: Objektumorientált tervezés és programozás Java 1. és 2. kötet** Java forráskódjainak fordítására, futtatására, és új programok írására. Az itt bemutatott javaprog.jpx általános projektfájl alkalmazásával kezelhető bármely különálló, csomagolatlan Java forráskód.

Tartalom:

1. A JBuilder letöltése, indítása

A JBuilder **integrált fejlesztői környezet**; fejlesztőeszközöket (szövegszerkesztő, fordító, futtató, nyomkövető stb.) építettek össze (integráltak) benne, és rengeteg kényelmi funkciót is belefoglaltak.

A JBuilder Personal (aktuális verziója a 8.0) a www.borland.com lapról tölthető le, és ingyenesen használható. Több platformra is elkészült: Windowsra, Linuxra és Solarisra. A letöltendő tömörített állomány mérete nagyjából 60 MB. Ajánlatos a dokumentációt is letölteni – további 60 MB-ot. Letöltéskor a felhasználónak regisztrálnia kell magát; válaszul a Borland e-levélben aktivációs állományt küld, s abban a regisztrációs kulcsot.

A JBuilder erőforrásigénye:

- Memória: 256 MB
- Lemezterület: JBuilder: ~120 MB, dokumentáció: ~70 MB
- Processzor: Intel Pentium II 233 MHz vagy ezzel kompatíbilis.

A telepítés előtt vizsgálja meg, van-e elegendő hely a lemezén!

Telepítse először a JBuilder 8.0 fejlesztőeszközt! Futtassa az install programot; a szoftver helyének adja meg valamelyik lemezének JBuilder8 mappáját (pl. C:/JBuilder8)!

Ezután telepítse a JBuilder 8.0 dokumentációját, vagyis a segítséget (helpet)! A JBuilder mappájának adja meg azt a mappát, ahová a JBuildert telepítette (pl. C:/JBuilder8), a JDK (Java Development Kit, Java Fejlesztői Készlet, lásd később) mappájának pedig az az alatti jdk1.4 mappát (pl. C:/JBuilder8/jdk1.4).

A JBuilder első indításakor meg kell adni az e-levélben kapott aktivációs állományt.

> *Megjegyzés:* A könyvtárneveket elválasztó jel hol / lesz, hol meg \, mert lehetetlen következetesnek lenni. Az elválasztójel operációsrendszer-függő, és a JBuilder is minduntalan kicseréli a \ jelet /-re. Egy programban ajánlatos a / jelet használni.

2. A könyv melléklete

Töltse le a könyv elektronikus mellékletét a könyv hátán megadott címről! A melléklet a két kötet közös melléklete, neve javaprog.zip, mérete kb. 4 MB. Bontsa ki a fájlt, és tegye a C lemezegység főkönyvtárába! Ha más helyre teszi, akkor a könyv és a `javaprog` projekt hivatkozásai értelemszerűen átírandók.

Kibontás után a 9.1. ábrán látható könyvtárstruktúrát kapja. A mappákat vastag betűvel szedtük. Az első 3 mappa neve aláhúzással kezdődik, hogy az ábécé szerint előre kerüljenek. A _MyPrograms mappa a készülő, saját programokat tartalmazza; hozzunk létre almappákat a programok csoportosítására (például tanuló neve, dátum vagy téma szerint)!

A mappák mellett megjegyzésben tüntettük fel a tartalmukat. Bizonyos mappákat csak a könyv 2. kötete használ (pl. icons, images, sounds).

```
c:\
  javaprog
    _MyPrograms          // saját programjaink helye
    _OOTPJava1           // az 1. kötet melléklete
      Esettanulmanyok
      Feladatmegoldasok
      Mintaprogramok
    _OOTPJava2           // a 2. kötet melléklete
      Esettanulmanyok
      Feladatmegoldasok
      Mintaprogramok
    doc                  // dokumentációk
    original             // fontosabb fájlok eredeti változata
    icons                // ikonok
    images               // képek
    javaprog_bak         // a javaprog projekt forrásfájljainak másolata
    javaprog_classes     // a javaprog projekt class-fájljai
    javaprog_src // a javaprog projekt alapértelmezett forrásfájljai (üres)
    lib                  // segédkönyvtárak
      javalib.jar        // a javalib könyvtár tömörítve
      javalib_src.jar    // a javalib könyvtár forráskódjai tömörítve
    sounds               // hangfájlok
    work                 // munkakönyvtár (pl. fájlkezeléshez)
    javaprog.jpx         // önálló forráskódok projektfájlja
```

1. ábra. A könyv mellékletének könyvtárstruktúrája

A könyv mellékletében a programoknak kétféle lehet a „kiszerelésük”:

- **Önálló Java forráskód** (kiterjesztése java): Egyetlen java kiterjesztésű, önállóan fordítható és futtatható forrásállomány. Az első kötetben szinte csak ilyen programokkal foglalkozunk. Minden forrásállományt külön mappába tettünk. A program fordításához és futtatásához be kell töltenünk a Java forrásállományt.
- **Projekt** (a projektfájl kiterjesztése jpx): Nagyobb, több állományból álló program. Az állományok lehetnek forráskódok, adat-, kép-, hangfájlok stb. Mindezek az állományok egy projektkönyvtárnak nevezett könyvtár alatt sorakoznak, könyvtárstruktúrába szervezve. A projekt állományait egy projektfájl fogja össze (JBuilderben egy jpx kiterjesztésű fájl). A program fordításához és futtatásához elegendő a projektfájlt betölteni.

Projektkészítéssel majd a 2. kötet foglakozik – ebben a könyvben csak futtatjuk, s ha kell, fordítjuk őket.

Először nagy vonalakban áttekintjük a JBuilder alkalmazásböngésző elemeit, majd lefordítunk és futtatunk egy JBuilder-projektet és egy önálló Java forráskódot.

3. A JBuilder alkalmazásböngészője

A **projekt (project)** egy szoftver fejlesztésében használt, logikailag összetartozó állományok és környezeti beállítások gyűjteménye. Minden projektnek van egy könyvtára; ebben a könyvtárban van a projektet leíró fájl, itt vannak továbbá a projekt alkönyvtárai és állományai.

Például:

```
C:/javaprog/_OOTPJava1/Esettanulmanyok
  GyusziJateka
    GyusziJateka.jpx  // projektfájl
    src               // a projekt forrásfájljai
    classes           // a projekt bájtkódjai
    ...
```

A `GyusziJateka` projekt könyvtára: C:/javaprog/_OOTPJava1/Esettanulmanyok/GyusziJateka; projektállománya: GyusziJateka.jpx.

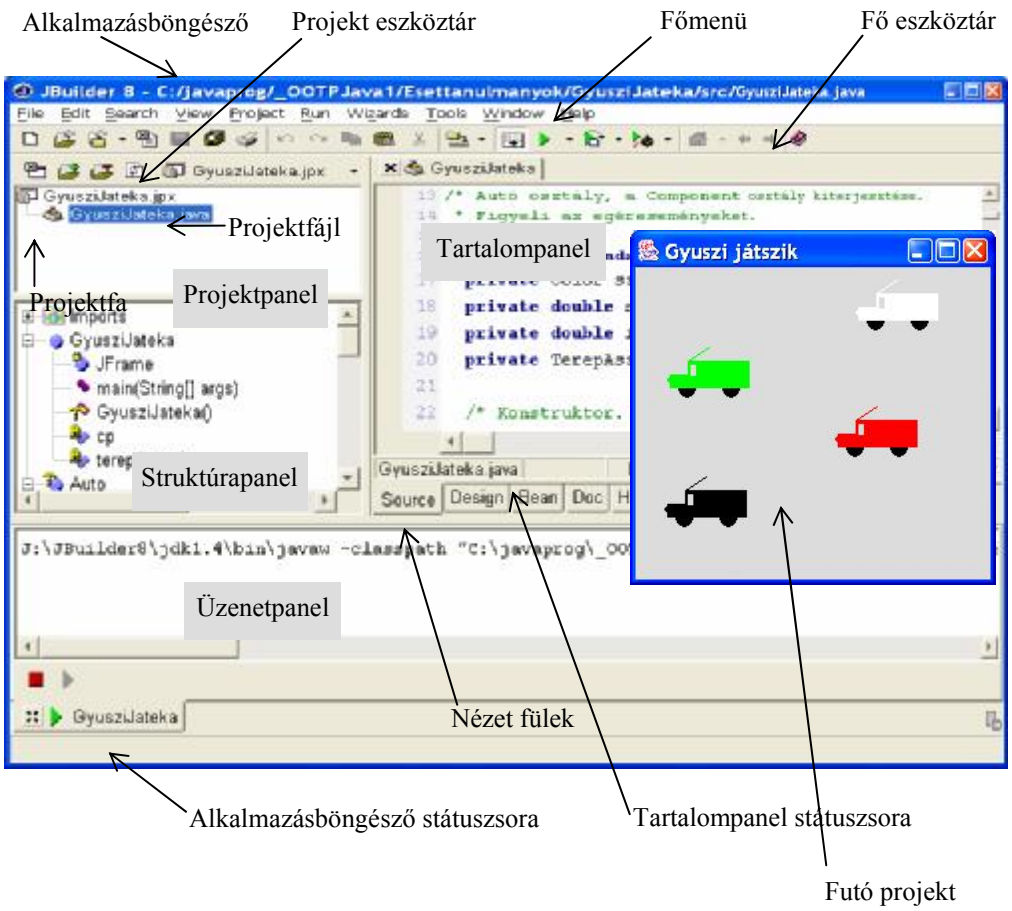
Nyissa meg a `GyusziJateka` projektet:

- *File/Open Project...* Válassza ki a következő állományt:

  C:/javaprog/_OOTPJava1/Esettanulmanyok/GyusziJateka/GyusziJateka.jpx.

A projektet nem kell lefordítani, mert a lefordított kód része a mellékletnek. Ha kettőt kattintunk a projektfa `GyusziJateka.java` elemén (2. ábra), akkor a tartalompanelen megjelenik a forráskód. Futtassa a `GyusziJateka` projektet:

- *Run/Run Project* (*F9*)



2. ábra. A GyusziJateka projekt a JBuilderben

A JBuilder fő ablakát alkalmazásböngészőnek nevezzük (application browser). Ez látható a 2. ábrán: most éppen a `GyusziJateka` projekt fut benne. A program egy ablakot jelenít meg, az üzenetpanelre nem ír semmit.

Az alkalmazásböngésző részei:

- **Főmenü** (Main menu). Menüpontok: File, Edit, Search, View, Project, Run, Wizards, Tools, Window, Help.
- **Fő eszköztár** (Main toolbar). Innen érhetők el az alkalmazásböngészőre vonatkozó főbb menüpontok. A gombok funkció szerint vannak csoportosítva: File, Edit, Search, Build, Run, Debug, Navigation és Help. A különféle csoportok elrejthetők, illetve újra láthatóvá tehetők a *View/Toolbars* menüpontban.
- **Projektpanel** (Project pane). Ez mutatja a projekt elemeit, fastruktúrába szervezve. A panelen van a **projekt eszköztár**a (project toolbar). Mindig van egy aktuális projekt: az, amelyen dolgozunk – ezt a projektet a **projektválasztó listá**ból lehet kiválasztani. A **projektfa** mindig az aktuális projekt elemeit mutatja. A + és – ikonnal elemeket adhatunk a projekthez, illetve elemeket vehetünk le róla.
- **Tartalompanel** (Content pane). A tartalompanelen ott van az összes nyitott állomány (a megnyitott állományoknak egy-egy fül felel meg). Egy állományt úgy nyithatunk meg, hogy duplán kattintunk a projektfa megfelelő elemén. Ha a JBuilder felismeri a megjelenítendő állományt, akkor azt a szokásos módon jeleníti meg – ilyenek például a java, class, txt, html, xml, jpg, gif, wav, zip, és jar kiterjesztésű állományok. A megnyitott állományok között mindig van egy aktuális, ehhez kapcsolódik a struktúrapanel és a tartalompanel státuszsora.

  A projekt elemei és a nyitott állományok elvileg függetlenek egymástól. A projekt elemei megnyithatók, de megnyithatók más, a projekthez nem tartozó állományok is (*File/Open File...*).

  A tartalompanel részei:

  - **Fájlfülek** (felül). A fülre való kattintással kiválasztható az aktuális állomány.
  - **Nézetfülek** (alul: Source, Design ...). Az aktuális állománynak különböző nézetei lehetnek. A megfelelő fülre kattintva nézetet válthatunk. A Source fülön található a forráskód; a Design fület vizuális tervezésnél használjuk stb.
  - **A tartalompanel státuszsora**. Információk az aktuálisan szerkeszthető állományról: az állomány neve, a kurzor pozíciója (sor:oszlop), Modified, Insert/Overwrite.
- **Struktúrapanel** (Structure pane). A tartalompanel aktuális elemének struktúráját mutatja – Java forrásfájlban a csomag- és típusdeklarációkat, valamint a különféle típusok tagdeklarációit.
- **Üzenetpanel** (Message pane). Az üzenetpanel lapjain jelennek meg a hibaüzenetek, valamint a keresések, futtatások és nyomkövetések információi, eredményei. Ez a program „konzolablaka” is: itt kell tehát megadni a program beviteli adatait (például a Console osztály metódusaival bekért adatokat). **Adatbekérés előtt az üzenetpanelt fókuszba kell hozni!**

  Az egyes tevékenységekhez külön lap nyílik:

  - Fordítás (Compiler). A lap megjeleníti legutóbbi fordítás hibalistáját, ha a fordítás nem sikerült volna. A hibaüzenetre kattintva a hibás sorra állhatunk.
  - Futtatás (Az aktív program neve). Külön lap nyílik minden futó programszálnak.
  - Nyomkövetés (Debugger).
  - Keresés (Search Results). A lapon a legutóbbi keresés eredménye látható.

  Minden lapon két gombot találunk: az egyikkel leállítható a szál, a másikkal újraindítható. Az üzenetpanel lapjai törölhetők: *Jobb egérgomb a lap alján/Remove...* Az üzenetpanel elrejthető: *View/Messages*.
- **Az alkalmazásböngésző státuszsora** (Status line): Egysoros információ az alkalmazás állapotáról.

Kedvencek

Adja hozzá a kedvencekhez a javaprog mappát, hogy könnyebben odataláljon! Ezt az *File/ Open Project...* vagy a *Project/Add Files...* ablakban található *Favorites* (kedvencek, jele: szívecske) eszközgombbal teheti meg. A kedvencek használata megkönnyíti a munkát.

4. JBuilder-projekt fordítása és futtatása

Nézzük meg egy kicsit részletesebben, hogyan fordítható és futtatható le egy JBuilder-projekt! Közben keressen a könyv mellékletében és a JBuilder mintaprogramjai között további projekteket, és futtassa őket!

Futtassa például a C:/javaprog/_OOTPJava2/Esettanulmanyok/KissDraw/KissDraw.jpx projektet!

> *Megjegyzés:* Ne ijedjen meg! Egyelőre csak használnia kell ezt a programot. Ilyen programot csak a könyv 2. kötetében fogunk írni.

Projekt megnyitása

- *File/Open Project...* Megjelenik az *Open Project* dialógusablak. Az ablakban csak a projektfájlok jelennek meg (jpr vagy jpx kiterjesztés). Keressük meg a lemezen, és válasszuk ki a kívánt projektfájlt!
- *File/Open File...* (*Ctrl + O*) Megjelenik az *Open File* dialógusablak. Az ablakban feltűnik az összes állomány. Keressük meg a lemezen, és válasszuk ki a kívánt projektfájlt!

Ha projektfájlt nyitunk meg, akkor ez a két funkció csak a megjelenített fájlokban tér el egymástól.

Projekt(ek) bezárása

- *Projekt eszköztár/Kék X* Bezárja az aktuális projektet.
- *File/Close Projects...* Bejelöljük a bezárandó projekte(ke)t.

Projekt fordítása

- *Project/Make Project "[Projektfájl]"* (*Ctrl + F9*, *Fő eszköztár/Sárga téglalap/Make*). Lefordítja a projektben az időközben módosított forrásállományokat (azokat, amelyeknek a dátuma későbbi, mint a már lefordított class állományé). Ha szükséges, akkor az összes forráskódot lefordítja. Fordítás előtt elmenti a forráskódot, ha azt megváltoztattuk volna a szövegszerkesztőben.
- *Project/Rebuild Project "[Projektfájl]"* (*Fő eszköztár/Sárga téglalap/Rebuild*). Lefordítja a projekt összes forrásállományát, feltétel nélkül. A megváltozott forráskódokat előbb lemezre menti.

Mindkét esetben megjelenik egy, a fordítás eredményességéről szóló üzenet az alkalmazásböngésző státuszsorában. Például: *Build succeeded with 1 file(s) built. Build took 1 seconds.* Vagyis: 1 állomány fordítása sikeresen befejeződött. A fordítás 1 másodpercig tartott.

Projekt futtatása

- *Run/Run Project* (*F9*, *Fő eszköztár/Zöld háromszög/[Futási konfiguráció]*). Futtatja a projektet. Ha a projekt forráskódjai még nincsenek lefordítva, akkor lefordítja őket.

Egy projekt tulajdonságai közt ott van a lefordítandó projekt típusa (alkalmazás/applet...), s ha alkalmazásról van szó, akkor a `main` metódust tartalmazó osztály neve is. Egy kész projektben (így a könyv projektjeiben is) minden tulajdonság úgy van beállítva, hogy a projekt futtatható legyen, a projektek beállításaival ezért remélhetőleg nem kell törődnünk.

A könyv bizonyos projektjei megkövetelik a javalib könyvtár jelenlétét; ez is benne van a könyv mellékletében (javaprog/lib/javalib.jar). Ilyenkor a megfelelő könyvtárat hozzá kell adni a környezethez (lásd a fejezet „A javalib könyvtár konfigurálása” pontját).

5. Önálló program fordítása, futtatása

A JBuilderben csak projekt keretein belül lehet programot írni. A programfejlesztést a programozó általában egy projekt összeállításával kezdi: különböző könyvtárakat, fájlneveket és alapértelmezéseket kell megadnia, beállítania. Nagyobb program fejlesztésekor a projekt nagy segítséget ad. Más esetekben viszont kényelmetlen és értelmetlen lehet külön projektet és projekttartozékokat létrehozni – például akkor, ha

- mindössze egyetlen, rövid Java forráskódot készítünk, vagy ha
- egy meglévő Java forráskódot fordítunk és futtatunk. Gyakori eset, hogy a Java forráskódhoz nincs projektfájl vagy ha van, akkor az más fejlesztőrendszerből való. A különböző Java fejlesztőkörnyezetek projektállományai általában nem kompatíbilisek egymással. Egy szabványos Java forrásállomány független a fejlesztési környezettől.

Könyvünkben az 1. kötet programjaihoz és a 2. kötet programjainak nagyjából 80%-ához nem tartozik külön projektfájl.

Egy önálló Java forrásállományt többféleképpen fordíthatunk és futtathatunk. Legegyszerűbb, ha a kívánt forrásállományt hozzáadjuk a könyv mellékleteként megadott `javaprog` projekthez.

Példaként futtassa a 14. fejezet `Csillag.java` programját! A feladat a következő:

Írjunk ki a konzolra 10 darab csillagot egy sorba!

Nyissa meg a `javaprog` projektet:

- *File/Open Project...* Válassza ki:

  C:/javaprog/javaprog.jpx

Nyissa meg és adja a projekthez a `Csillag.java` forráskódot:

- *Projekt eszköztár/+* Válassza ki:

  C:/javaprog/_OOTPJava1/Mintaprogramok/14Iteraciok/Csillag/Csillag.java

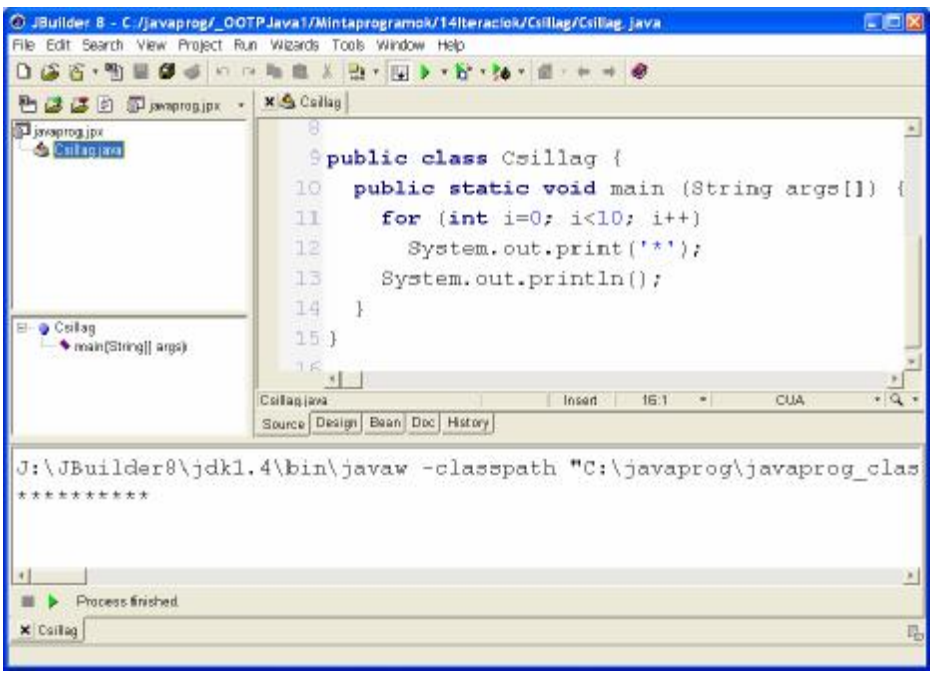
Ha kettőt kattintunk a projektfán a Csillag.java névre, akkor a forráskód nyomban megjelenik a tartalompanelen.

Fordítsa le és futtassa a `Csillag.java` programot (a lefordított kód nem része a könyv mellékletének, futtatáskor azonban automatikusan lezajlik a fordítás):

- Jelölje ki a forrásprogramot a projektfán! *Jobb egérgomb/Run using "javaprog"*

A 3. ábra mutatja a `Csillag.java` forrásprogram futását. Az üzenetpanelen megjelenik 10 darab csillag.

Megjegyzés: A programot egyelőre csak futtatnia kell!



3. ábra. A Csillag.java önálló Java program a JBuilderben

A javaprog projekt különálló forráskódok fejlesztéséhez (fordításához és futtatásához) **használható**. A javaprog.jpx projektfájl a könyv mellékletéhez tartozik, egyszerűen megnyitható és használható. Nem kell bezárni; jó, ha mindig „kéznél van”.

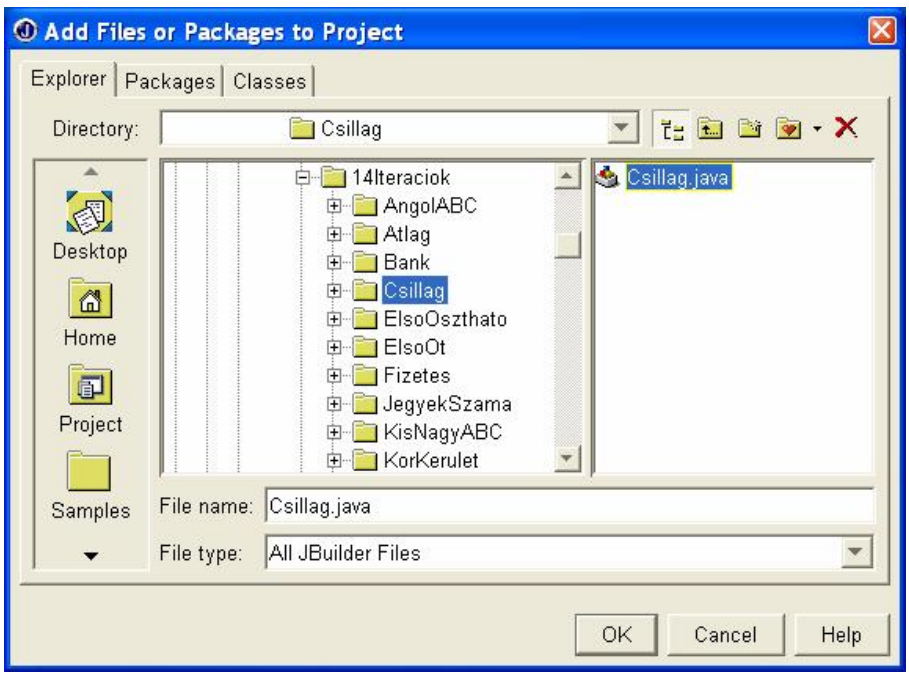
A javaprog projekthez hozzáadott önálló forrásprogramok nem fordíthatók és futtathatók projektként, vagyis rájuk nem alkalmazhatók a *Project/Make Project*, a *Project/Rebuild Project*, valamint a *Run/Run Project* menüpontok.

Most egy kicsit részletesebben megmutatjuk, hogyan kell egy Java forrásállományt kiválasztani vagy létrehozni, majd a `javaprog` projekthez hozzáadni, fordítani és futtatni.

Elem hozzáadása a projekthez

- *Project/Add Files/Packages...* vagy *Projekt eszköztár/+* Megjelenik a 4. ábra dialógusablaka. Az Explorer (böngésző) fül segítségével keresgélhetünk a lemezen. Ha már meglevő Java forráskódot akarunk hozzáadni a projekthez, akkor egyszerűen csak kiválasztjuk. Ha egy új forráskódot akarunk létrehozni, akkor írjuk be a *File name* beviteli mezőbe a kívánt nevet a java kiterjesztéssel, és nyomjuk le az Ok gombot. Az önállóan futó programok forrásállományait ajánlatos külön könyvtárba tenni – új könyvtárat létrehozhatunk itt a böngészőben is. Ha nem létező fájlnevet adunk meg, akkor a rendszer megkérdezi, hogy létre akarjuk-e hozni. A kiválasztott vagy létrehozott elem ezután megjelenik a projektfán.

Új Java forráskód létrehozásakor ne felejtse el megadni a **java kiterjesztést**! Ha nem adunk meg kiterjesztést, akkor az txt lesz, s az nem kezelhető Java forráskódként.



4. ábra. Elem hozzáadása a projekthez

Megjegyzések:

- Egy projekthez elvileg bármilyen állományt (szöveget, bájtkódot, képet, hangot, HTML lapot...) hozzáadhatunk, de mi most Java forrásállományt szeretnénk fordítani és futtatni.
- Egy Java forráskódot a *File/Open* menüponttal is megnyithatunk, a projekthez való hozzáadás nélkül. A projekt aktuális beállításai ilyenkor is érvényesek, de körülményesebb a fordítás és futtatás (futtatáskor például nincs automatikus fordítás).

Elem lekapcsolása a projektről

- *Project/Remove from Project...* vagy *Projekt eszköztár/–* Előtte ki kell jelölni a projektelemet a projektfán. A fájl nem törlődik a lemezről, csupán kikerül a projektből.

Gyakorlásképpen kapcsolja le a programot a projektről, majd újra adja hozzá!

Önálló forrásfájl fordítása

A javaprog projekt nem fordítható, mert a forrásprogramok nem a projekt forráskönyvtárában (C:/javaprog/javaprog_src) helyezkednek el, hanem bárhol a lemezen. A forráskódokat egyedileg kell fordítani. Lehetőségek:

- A projektfán jelöljük ki a forrásfájlt! *Jobb egérgomb/Make*
- Jelenítsük meg a tartalompanelen a forrásfájlt! Ha előzőleg már megjelenítettük, akkor válasszuk ki a tartalompanelen (kattintsunk a nevének megfelelő fülre)! *Project/Make "[Forrásfájl]"* (*Ctrl* + *Shift* + *F9*)

Bájtkódok törlése

A lefordított class állomány az *Output Path* helyen keletkezik – a javaprog projektben a C:/javaprog/javaprog_classes könyvtárban. Mivel a javaprog projekt minden bájtkódot ide tesz, azért ajánlatos ezt a könyvtárat időnként kiüríteni:

- *Jobb egérgomb/Clean*. Ha előtte a projektfájl volt kijelölve, akkor a projekt összes bájtkódja törlődik; ha egy forráskód volt kijelölve, akkor törlődik a neki megfelelő class állomány.

Önálló forrásfájl futtatása

A javaprog projekt nem futtatható; a hozzáadott forrásprogramot csak egyedileg lehet futtatni. Lehetőségek:

- *Run/Run "[Forrásfájl]"*
- *Jobb egérgomb/Run using "javaprog"*. Előzetesen ki kell jelölni a forrásprogramot a projektpanelen.

> *Megjegyzés:* Ha valamiért használhatatlanná válna a javaprog projekt, akkor a javaprog_bak könyvtárban megtaláljuk a eredeti, sértetlen jpx állományt.

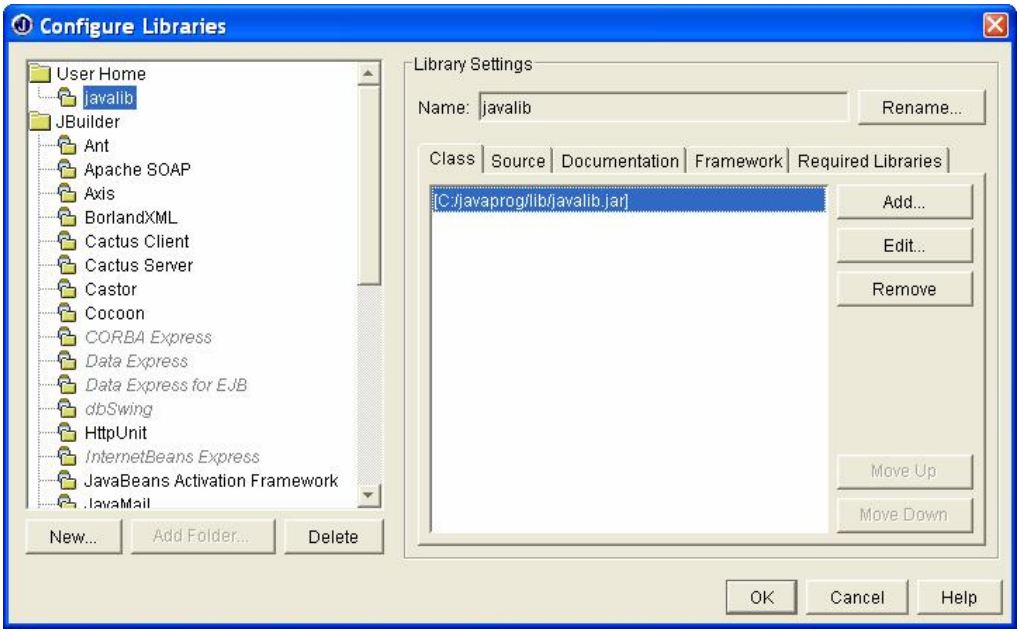
6. A javalib könyvtár konfigurálása

Az eddig futtatott programok mind a rendszerrel együtt szállított osztályokra támaszkodtak. Más programok azonban „nem érik be” az alapértelmezésben jelen levő API (Application Programming Interface) könyvtárakkal – a benne levő csomagok osztályaival –, hanem további könyvtárakra is hivatkoznak. A könyv programjainak többsége használja például a `javalib` könyvtárat – az is benne van a mellékletben. A `javalib` könyvtárban van az `extra.Console` osztály; a benne szereplő statikus metódusok segítségével adatokat kérhetünk be a konzolról (a Java írói nem gondoltak a kezdő programozókra).

A javaprog projekt úgy van beállítva, hogy támaszkodjék a javalib könyvtárra. A tényleges használhatósághoz azonban be kell építenünk ezt a könyvtárat a JBuilder környezetébe, s azután a JBuilder már emlékezni fog rá. Válasszuk ki a *Tools/Configure Libraries...* menüpontot! Válasszuk aktuális könyvtárnak a *User Home* könyvtárat, majd nyomjuk le a *New...* gombot (5. ábra)! Megjelenik a *New Library Wizard* ablaka. Adjuk meg a következő adatokat:

- *Name* (a könyvtár neve): javalib
- *Library Path* (a könyvtár útvonala):
  - *Class:* javaprog/lib/javalib.jar // class fájlok könyvtára tömörített formában
  - *Source*: javaprog/lib/javalib_src.jar // java fájlok könyvtára tömörített formában

Most már a két kötet összes programja futtatható – vagy önálló projektként, vagy a javaprog projekt segítségével.

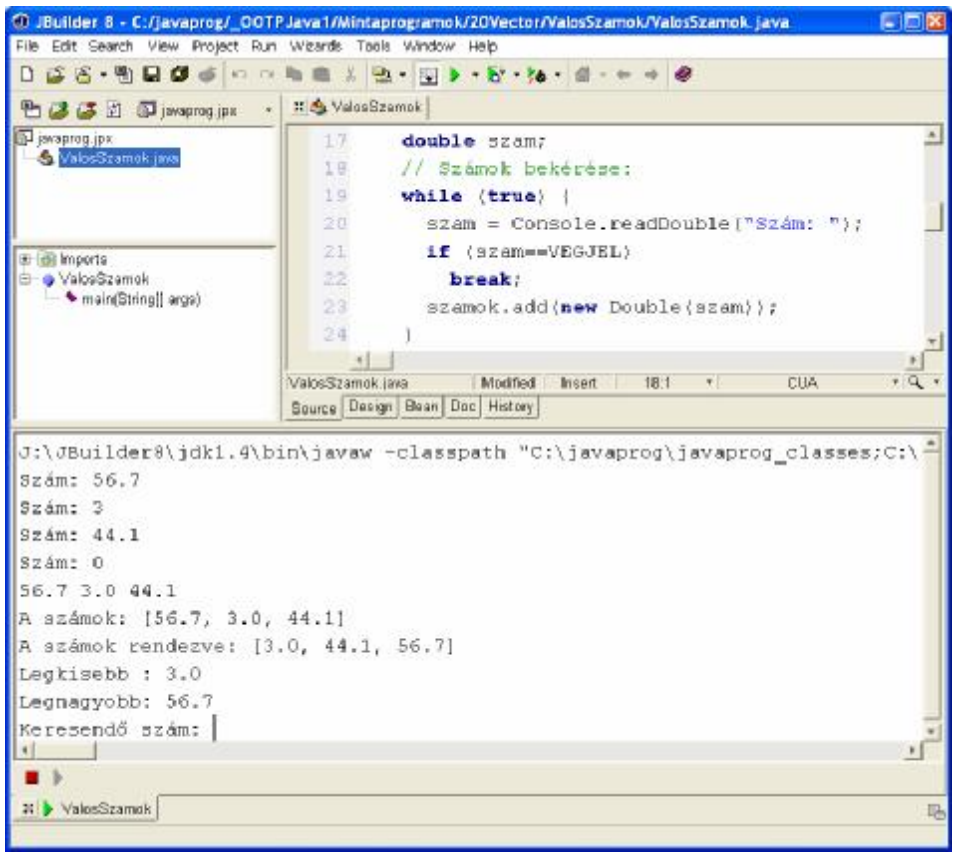


5. ábra. A javalib könyvtár konfigurálása

Próbaként futtassa a 20. fejezet `ValosSzamok.java` programját! A feladat a következő:

Kérjünk be tetszőleges sok valós számot 0 végjelig! Ezután írjuk ki
- a számokat a bevitel sorrendjében!
- a számokat növekvő sorrendben!
- a legkisebb és a legnagyobb számot!

Végül keressünk meg egy számot a bevittek között!



6. ábra ValosSzamok.java – Beolvasás konzolról

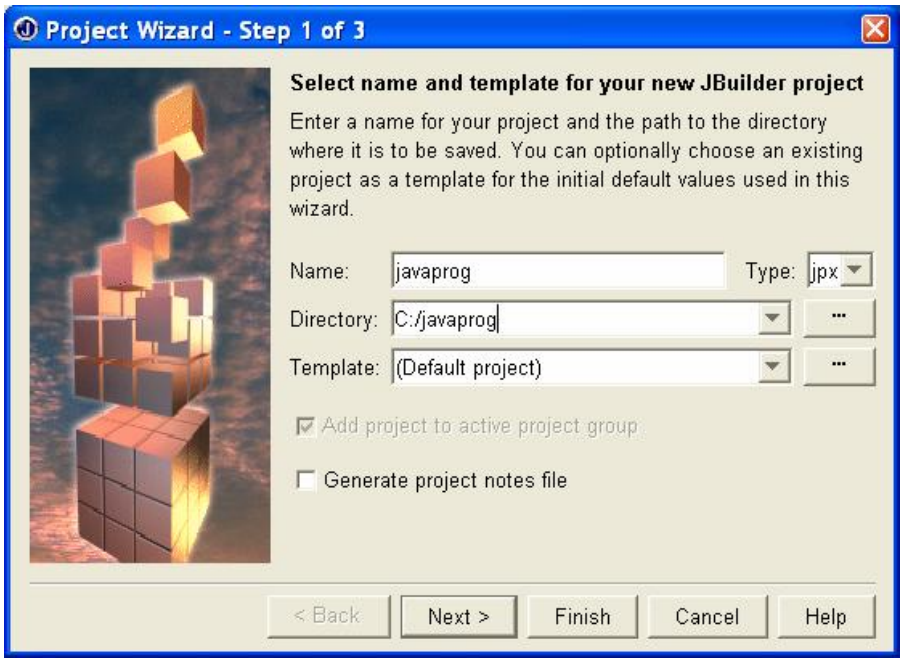
A program futását a 6. ábra mutatja. Az üzenetpanelen megjelenik a "Szám: " szöveg, jelezve, hogy a program bevitelre vár. Át kell váltanunk az üzenetpanelre, és be kell ütnünk egy valós számot. A program ezután kéri a következő számot, egészen addig, amíg a 0 végjelet be nem ütjük. Erre a program befejezi a számok gyűjtését, és kiírja a kért adatokat (az eredeti sorrendben kétféle kiírás történik). Végül kér egy számot, hogy megkereshesse azt a beütött számok között. Az ábrán a kurzor éppen bevitelre vár. A program első sora: `import extra.Console;` (nem látszik), jelezve, hogy használni akarja a `Console` osztályt.

Próbálja meg futtatni a 2. kötet esettanulmányait is! A javalib_projekt kivételével itt minden mappában egy futtatható projekt van (a javalib_projekt önállóan nem futtatható osztálykönyvtár).

7. A javaprog projekt létrehozása

A javaprog projekt a könyv melléklete, azt nem kell létrehozni. S mivel a JBuilder indításkor automatikusan megnyitja az utoljára használt projekteket, azért a javaprog projekttel gyakorlatilag nincs semmi dolgunk... Ha mégis létre szeretné hozni a javaprog projektet, kövesse a következő útmutatást!

- *File/New Project...* Megjelenik a projekt varázsló. Állítsa be a mezőket az ábrákon megadott módon!



7. ábra. Projekt varázsló – 1. lépés

1. lépés (7. ábra):

- *Name:* javaprog. A projekt neve. Ez lesz a projektfájl neve is, `jpx` kiterjesztéssel.
- *Directory:* C:/javaprog A projekt könyvtára. Válasszuk ki a C:\ főkönyvtárat, s az majd automatikusan kiegészül a projekt nevével. Ajánlatos a mappa és a projektfájl nevét egyformának választani.
- *Template:* (Default project) Projektminta: ennek alapján jön létre az új projekt.

2. lépés (8. ábra):

- *JDK:* A használt JDK aktuális verziója. A JBuilder Personalban ez nem módosítható.
- *Output path:* C:/javaprog/javaprog_classes. Itt keletkeznek a projektben lefordított bájtkódok. Időnként ajánlatos törölni őket.

- *Backup path:* C:/javaprog/javaprog_bak. Itt keletkeznek a projektben szerkesztett forráskódok biztonsági másolatai.
- *Working directory:* C:/javaprog. Munkakönyvtár – a projekt forráskódjaiban megadott relatív útvonalak kiinduló könyvtára.
- *Source*: C:/javaprog/javaprog_src. Csak az osztályvarázslóval létrehozott forráskódok keletkeznek itt. Megadása kötelező, de ebben a projektben nem használjuk – ez a mappa tehát általában üres.
- *Required Libraries*: Itt kell beállítani a programban használt könyvtárakat (a CLASSPATH útvonalait, lásd a JDK leírását). A 9. ábra a javalib könyvtár beállítását mutatja. A könyvtára(ka)t előzőleg a *New* segítségével vagy a *Tool/Configure Libraries...* menüpontban be kell állítani. A különféle projektekben a már megadott könyvtárakból válogatunk. Ha a környezetben megváltoztatjuk egy könyvtár útvonalát, akkor a beállítás a JBuilder összes projektjét érinti!

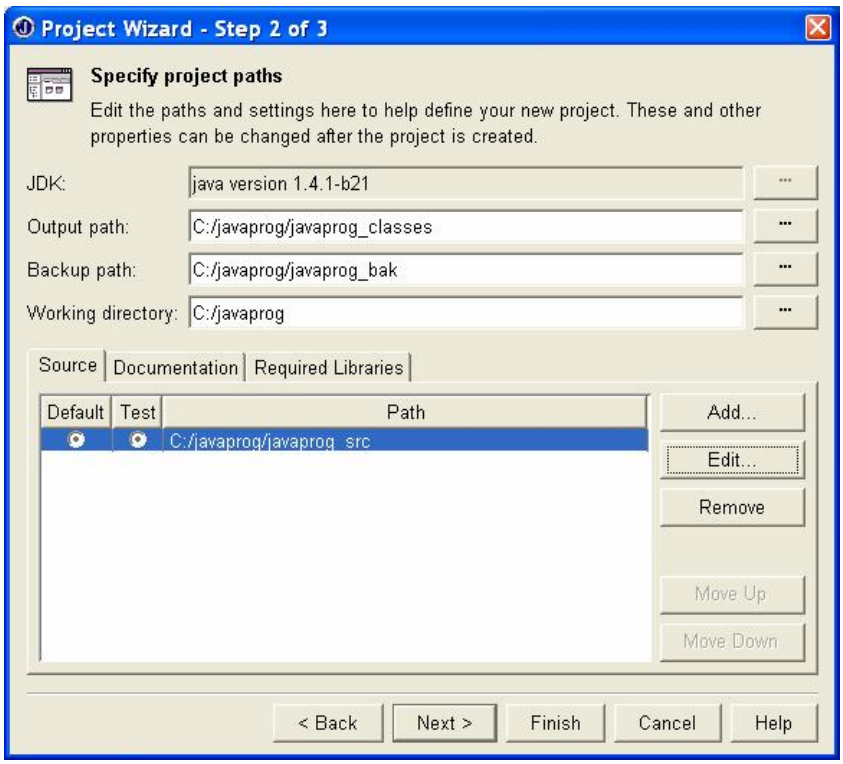
Egy projekt ajánlott mappanevei: classes, bak és src. A javaprog projektben azért adtunk tőlük különböző neveket, hogy a mappák ábécé rendben kövessék egymást – így áttekinthetőbb a javaprog könyvtár.

3. lépés:

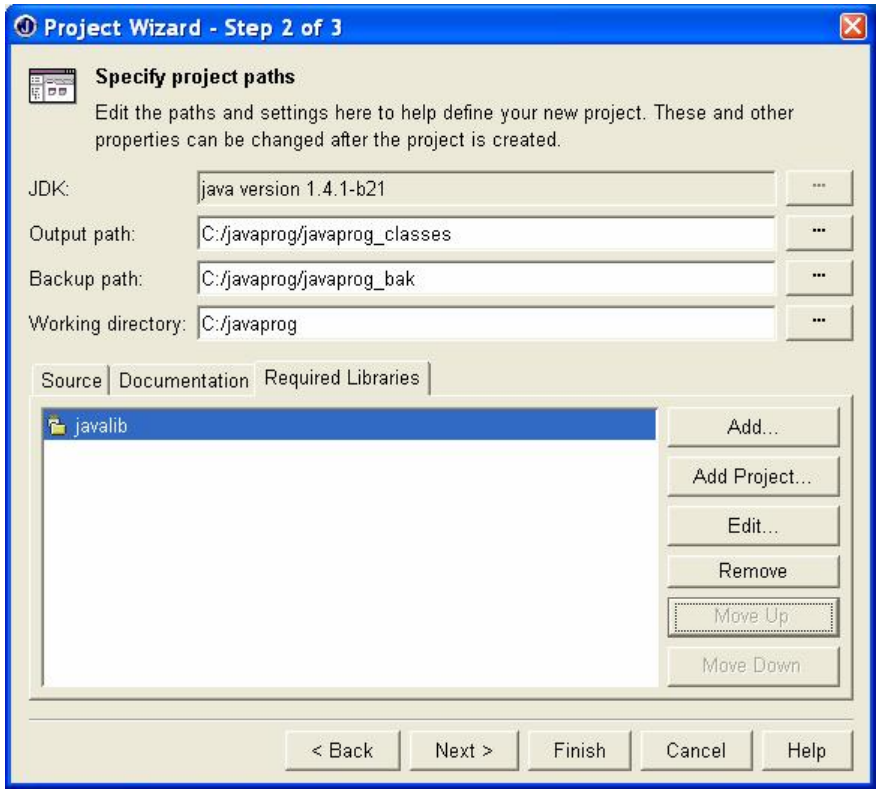
Itt nincs mit tennünk. Nyomjuk meg nyugodtan már a 2. lépés végén a *Finish* gombot! A projekt a c:/javaprog könyvtárban keletkezik, javaprog.jpx névvel.

Ezt a javaprog projektet csak egyszer kell létrehozni. Ezután a megfelelő Java forrásállományt **csak hozzá kell adni a projekthez, illetve le kell venni róla**.

A javaprog projektet nem kell bezárni. Egyszerre több projekt is nyitva lehet.



8. ábra. Projekt varázsló – 2. lépés, útvonalak beállítása



9. ábra. Projekt varázsló – 2. lépés, szükséges könyvtár megadása

8. A JBuilder szövegszerkesztője

A tartalompanelen megjelenő forráskód szerkeszthető. Felsoroljuk a szövegszerkesztő fontosabb lehetőségeit, csoportokba szedve.

Szövegírás

- **Insert:** Váltás a Beszúró mód és Felülíró mód között. Beszúró üzemmódban (a státuszsorban: Insert) a kurzor egy vékony vonal, felülíró üzemmódban (a státuszsorban: Overwrite) pedig egy tömör, az aktuális karakteren villogó téglalap.
- **Karakter**: Beszúró üzemmódban a leütött karakter a kurzor helyére kerül, az addigi karakterek pedig eggyel jobbra tolódnak a sorban. Felülíró üzemmódban az újonnan beírt karakterek felülírják a már ott levőket.
- **Enter:** Új sor kezdése. Felülíró üzemmódban a kurzor a következő sor első pozíciójára áll. Beszúró üzemmódban ezzel egy sort szúrunk be, a kurzor ebben az új sorban a felette álló sor első karaktere alá áll. Szerkesztés közben a program struktúrája automatikusan alakul a beírt vezérlő utasítások alapján (Indent mód).

Pozicionálás

- → A kurzor egy karakterrel jobbra lép.
- ← A kurzor egy karakterrel balra lép.
- Ctrl + → A kurzor egy szóval jobbra lép.
- Ctrl + ← A kurzor egy szóval balra lép.
- ↑ A kurzor egy sorral feljebb lép.
- ↓ A kurzor egy sorral lejjebb lép.
- Home A kurzor a sor elejére ugrik.
- End A kurzor a sor végére (az utolsó karakter mögé) ugrik.
- Ctrl + Home A kurzor a szöveg elejére ugrik.
- Ctrl + End A kurzor a szöveg végére ugrik.
- Ctrl + Page Up A kurzor az aktuális oldal tetejére ugrik.
- Ctrl + Page Down A kurzor az aktuális oldal aljára ugrik.

Törlés

- Delete — A kurzor alatt levő karakter törlése. A kurzor helyben marad.
- BackSpace (←) — A kurzor előtti karakter törlése; a kurzor eggyel balra lép.
- Ctrl + Y — A kurzor sorának törlése. Az alatta levő sorok eggyel feljebb húzódnak, s a kurzor a következő sor elejére áll.

Kódbeillesztés – Ctrl + J

A kódbeillesztés megkönnyíti a kódolást: előre elkészített kódrészletek illeszthetők be vele a programba. A Ctrl + J hatására egy lista tűnik fel, különböző kódrészletek azonosítóival. A kiválasztott kódrészlet bekerül a forráskódba.

A `main` blokkot például minden programba be kell írnunk – ez egyrészt unalmas, másrészt hiba forrása lehet. A main azonosítójú kódrészlet kiválasztására a szövegbe kerül a main blokk.

Fontosabb elemek:

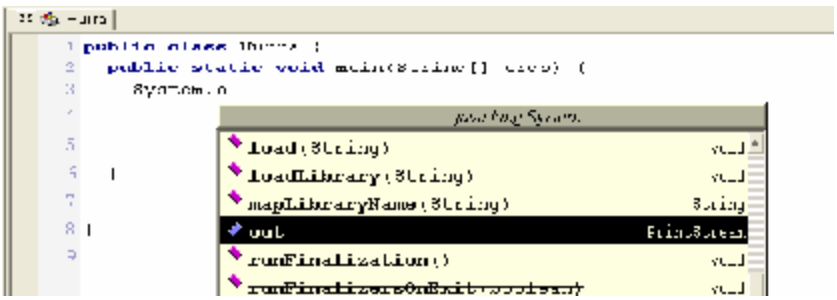
Azonosító	Kódrészlet
classp	`public class {}`
main	`public static void main(String[] args) {}`
out	`System.out.println("");`
addwin	`addWindowListener(new WindowAdapter() {...});`

Automatikus metódusfelkínálás

Ha egy osztály vagy objektum neve után pontot teszünk, akkor a rendszer automatikusan felkínálja azokat az adatokat, metódusokat, amelyeket a fordító ezen a helyen elfogad (11. ábra). Mindez csak akkor megy így, ha a program eddig szintaktikailag hibátlan, és tartalmazza a megfelelő importdeklarációkat.

Hibafeltárás

Ha a kódban szintaktikai hiba van, akkor szerkesztés közben a struktúrapanelen megjelenik egy *Errors* gyökérelem, és felsorolja a hibákat. Ha a hibát kijavítjuk, akkor ez az elem eltűnik. Ajánlatos a hibákat folyamatosan javítani, különben nem működik az automatikus metódusfelkínálás.



11. ábra. Automatikus metódusfelkínálás

Automatikus befejezés – Ctrl + Space

Ha elkezdünk írni egy szót, akkor a Ctrl + szóköz lenyomására a rendszer a szót „legjobb tudása szerint” befejezi. Ha a folytatás még nem egyértelmű, akkor a rendszer felkínálja a lehetőségeket.

A szövegszerkesztőben van ezen kívül **szintaktikai kiemelés** (különböző típusú szavak más színnel jelennek meg), valamint **helyzetérzékeny help** (F1 lenyomására megjelenik annak az osztálynak a leírása, amelyen a kurzor áll).

– Vége –