آموزش گام به گام برنامه-نویسی جاوا

تالیف

دکتر جواد وحیدی

دکتر همایون موتمنی

دکتر رمضان عباس‌نژادورزی

فن‌آوری نوین

سرشناسه	:	وحیدی، جواد، ۱۳۴۸ -
عنوان و نام پدیدآور	:	آموزش گام به گام برنامه‌نویسی جاوا/ تالیف جواد وحیدی، همایون موتمنی، رمضان عباس‌نژادورزی.
مشخصات نشر	:	بابل: فناوری نوین ، ۱۳۹۸.
مشخصات ظاهری	:	۳۳۶ ص.: مصور، جدول.
شابک	:	۹۷۸-۶۰۰-۷۲۷۲-۲۷-۵:۵۵۰۰۰۰ ریال
وضعیت فهرست نویسی	:	فیپا
موضوع	:	جاوا (زبان برنامه‌نویسی کامپیوتر)
موضوع	:	Java (Computer program language) (
شناسه افزوده	:	موتمنی، همایون، ۱۳۵۰ -
شناسه افزوده	:	عباس‌نژاد ورزی، رمضان، ۱۳۴۸ -
رده بندی کنگره	:	QA۷۶/۷۳ ۱۳۹۸ ۳و۲ج /
رده بندی دیویی	:	۰۰۵/۱۳۳
شماره کتابشناسی ملی	:	۵۶۱۷۰۲۹

فن‌آوری نوین

www.fanavarienovin.net

بابل، صندوق پستی ۴۷۱۶۷-۷۳۴۴۸ تلفن: ۳۲۲۵۶۶۸۷-۰۱۱

آموزش گام به گام برنامه‌نویسی جاوا

تالیف: دکتر جواد وحیدی- دکتر همایون موتمنی- دکتر رمضان عباس‌نژادورزی

ویراستار: فن‌آوری نوین

ناشر: فن‌آوری نوین

چاپ اول: بهار ۸۷

جلد :۱۰۰

شابک: 9786007272275

حروفچینی و صفحه‌آرایی: فن‌آوری نوین **قیمت:۵۵۰۰۰ تومان**

تهران، خ اردیبهشت، نبش وحید نظری، پلاک ۱۴۲ **تلفکس: ۶۶۴۰۰۱۴۴-۶۶۴۰۰۲۲۰**

فهرست مطالب

مقدمه

جاوا توسط شرکت سان میکرو سیستم، به‌عنوان یک‌زبان شیء‌گرا توسعه داده شده است. **معماری بی‌طرف** و امنیت زبان برنامه‌نویسی جاوا باعث شده‌اند که این زبان به محبوبیت برسد. معماری بی‌طرف، یعنی این‌که می‌توانید با جاوا برنامه‌ای بنویسید که بر روی تمام سیستم‌عامل‌ها به خوبی کار کند. به همین دلیل، در برخی از دانشگاه‌های ایران در برنامه‌سازی پیشرفته جاوا تدریس می‌شود. یکی از راه‌های آموزش زبان‌های برنامه‌نویسی، آموزش مفاهیم برنامه به همراه مثال‌های متعدد است. کتاب حاضر شامل ۸ فصل است که عبارت‌اند از:

فصل اول، مفاهیمی از قبیل آشنایی با جاوا، عملگرها و دستورات ورودی و خروجی را با مثال‌های مختلف آموزش می‌دهد. فصل دوم، به ساختارهای کنترلی نظیر ساختارهای تصمیم (if، switch) و ساختارهای تکرار (for، do while، while، break و continue) می‌پردازد. فصل سوم، متدها و روش‌های پیاده‌سازی و فراخوانی آن‌ها را آموزش می‌دهد. فصل چهارم، آرایه یک‌بعدی و دوبعدی را با مثال‌های مختلف آموزش می‌دهد. فصل پنجم، شامل مفاهیمی از قبیل کلاس‌ها، شیء‌گرایی است، فصل ششم وراثت، چندریختی و پیاده‌سازی مجدد عملگرها را آموزش می‌دهد. فصل هفتم، مفاهیم رشته‌ها و کاراکترها را بیان کرده و متدهای کار با کاراکترها و رشته‌ها را آموزش می‌دهد. فصل هشتم، ورودی و خروجی فایل‌ها (استریم‌ها) و کلاس‌های موجود برای کارکردن با استریم‌ها را شرح می‌دهد.

علاوه بر فصول بیان شده به کتاب الکترونیکی یک پیوست اضافه شده است که برخی از سوالات ACM در آن حل شده است.

برای تمرین بیش‌تر انتشارات اقدام به انتشار **کتاب حل مسائل جاوا** را نموده است (**حدود ۶۰۰ برنامه جاوا در آن حل گردیده است**) که مکمل این کتاب است.

از ویژگی‌های بارز این کتاب این است که به‌صورت گام‌به‌گام با جملات کوتاه و ساده بیان گردیده است.

برنامه‌های متن کتاب را می‌توانید از سایت انتشارات فن‌آوری نوین به آدرس **www.fanavarienovin.net** دریافت نمایید.

در پایان امیدوارم این اثر موردتوجه جامعه انفورماتیک کشور، اساتید و دانشجویان عزیز قرار گیرد.

مؤلفین

fanavarienovin@gmail.com

فصل ۱ آشنایی با زبان جاوا

۱-۱. ویژگی‌های جاوا

جاوا ویژگی‌های زیادی دارد که آن را به یک زبان برنامه‌نویسی فوق العاده تبدیل کرده است. جاوا را می‌توان این‌گونه توصیف کرد: **زبانی ساده، شیءگرا، مفسری، قابل‌حمل، مقاوم، امن، پویا، توزیع‌شده و با کارایی بالا است که چند نخی و برنامه‌نویسی موازی را پشتیبانی می‌کند.** این ویژگی‌ها جاوا را به یک زبان جهانی و پرکاربرد تبدیل کرده است. ویژگی‌های جاوا عبارت‌اند از:

- **سادگی (Simple) :** اغلب یادگیری یک زبان برنامه‌نویسی جدید سخت است. ولی، جاوا به دلیل سادگی در ساختار زبانش با سرعت بیش‌تری قابل‌فهم و درک است. یکی از اهداف اصلی توسعه‌دهندگان جاوا حذف پیچیدگی‌های زبان‌هایی مثل C و C++ بود. اما، گرامر جاوا خیلی شبیه زبان‌های C و C++ است. ازآنجایی‌که اکثر توسعه‌دهندگان با زبان‌های C و C++ آشنا هستند، پس یادگیری جاوا بسیار آسان و سریع است.
- **شیءگرایی (Object-Oriented) :** الگوی شیءگرایی یک پارامتر استاندارد و بسیار رایج در توسعه‌ی نرم‌افزارهای امروزی به شمار می‌آید. یک شیء مدل نرم‌افزاری است که دارای ویژگی و رفتار است. از اشیاء می‌توان برای نمایش هر چیزی استفاده کرد. به‌عنوان‌مثال، یک ماشین دارای ویژگی‌هایی مثل تعداد درب، میزان سوخت، رنگ، مدل و همچنین رفتار و عملکردهایی مثل روشن شدن، خاموش شدن، ترمز گرفتن، شتاب گرفتن، تعویض دنده و غیره می‌باشد. برنامه‌نویسی شیءگرا بر روی ویژگی و رفتار تک‌تک اشیاء متمرکز است. این اشیاء می‌توانند با یک‌دیگر در ارتباط باشند و منطق بسیار پیچیده‌ی برنامه‌های امروزی را شکل بدهند. همچنین، اشیاء علاوه برداشتن ویژگی و رفتار دارای هویت نیز می‌باشند. به این معنی که هر شیء دارای آدرسی یکتا در حافظه‌ی کامپیوتر است.
- **مفسری بودن (Interpreted) :** کامپایلر، برنامه جاوا را دریافت و بایت کد آن را تولید می‌کند. بایت کد تولیدشده توسط ماشین مجازی جاوا (Java Virtual Machine) تفسیر و سپس اجرا می‌شود. فقط JVM است که وابسته به پلتفرم است، اما بایت کد تولیدشده مستقل از پلتفرم می‌باشد.
- **قابل‌حمل بودن (Portable):** برنامه‌های جاوا عملاً می‌توانند در هرجایی اجرا شوند. به عبارتی یک‌بار کد جاوا را می‌نویسید و سپس در هرجایی که بخواهید آن را اجرا می‌کنید. چون کد برای پلتفرم خاصی کامپایل نمی‌شود. برنامه‌های جاوا می‌توانند در هرجایی که JVM نصب باشد، اجرا

شوند. در مقابل، بسیاری از زبان‌های برنامه‌نویسی مانند COBOL، C++، Visual Basic و Smalltalk، برنامه را به یک فایل باینری کامپایل می‌کنند. فایل‌های باینری، مختص پلتفرم خاصی هستند. بنابراین، فایل باینری برنامه‌ای که برای ماشین‌های ویندوز نوشته‌شده، نمی‌تواند روی یک ماشین مک یا یک ماشین مبتنی بر لینوکس اجرا شود.

- **مقاوم بودن (Robust) :** کدهای جاوا مقاوم هستند. یعنی، کم‌تر اتفاق می‌افتد که برنامه نوشته‌شده با جاوا crash کند. چون، ماشین مجازی جاوا بررسی‌های خاصی را روی نوع هر شیء انجام می‌دهد تا از یکپارچگی آن مطمئن شود. اشیا جاوا تنها می‌توانند به اشیا واقعی ارجاع دهند و نه به هر جای دلخواه از حافظه. مکانیزم‌های قوی جاوا برای تخصیص و آزادسازی خودکار حافظه و مدیریت خطا نیز بر مقاوم بودن برنامه‌ها اضافه می‌کند. بررسی در زمان کامپایل و مدیریت خودکار حافظه و عدم استفاده از اشاره‌گرها باعث جلوگیری از ایجاد خیلی از باگ‌های پیچیده می‌شود. همچنین، مکانیزم زباله روبی (garbage collection) در زمان اجرا و مانیتور حافظه باعث مدیریت بهتر و پاک‌سازی حافظه می‌شود. جاوا، همچنین دارای مکانیزم توسعه‌پذیر اداره کردن استثنا (exception handling) شبیه سیستم مورداستفاده در C++ است.
- **امن بودن (Secured) :** بررسی در زمان کامپایل و مدیریت خودکار حافظه همکاری مهمی در امنیت جاوا دارند. کد جاوا قادر به نصب ویروس یا کرم بر روی سیستم نیست. چون دسترسی مستقیم به حافظه ندارد. جاوا در مواجهه با کدی که در یک شبکه در دسترس قرار می‌گیرد، واکنش نشان می‌دهد و به‌هیچ‌وجه به آن اعتماد نمی‌کند. البته این به این معنی نیست که کد نمی‌تواند اجرا شود، بلکه اجازه‌ی دسترسی به سیستم فایل محلی یا وسایلی مانند چاپگر و مودم را نمی‌دهد. این واکنش پیش‌فرض جاوا برای محافظت از سیستم است که کاربر می‌تواند آن را تغییر دهد.
- **پشتیبانی از چند نخی (Multithreaded) :** برنامه چند نخی در یک‌زمان می‌تواند بیش از یک کار انجام دهد، مثل مرورگرها. البته هر اجرا در زمان خودش انجام می‌شود، ولی به دلیل سریع بودن سوئیچ بین اجراها این‌طور به نظر می‌رسد که هم‌زمان در حال اجرا هستند. ازآنجایی‌که جاوا ذاتاً قابلیت چند نخی بودن را دارد امکان استفاده از این روش برای برنامه‌های شما قابل‌اجرا است.
- **کارایی بالا (High Performance):** دلیل کند بودن برنامه‌های جاوا، مفسری بودن زبان جاوا است. یک زبان مفسری دستورات را تک‌تک می‌خواند و آن‌ها را به دستورات قابل‌فهم برای سیستم کامپایل می‌کند. این مراحل باعث کندشدن سرعت می‌شود. بااین‌وجود، پیشرفت در مسیر کاری JVM باعث شده است سرعت اجرای برنامه‌های آن قابل‌مقایسه با سرعت برنامه‌های نوشته‌شده با زبان C++ در بیش‌تر وضعیت‌ها باشد. جاوا برای نوشتن درایورهای سطح پایین دستگاه‌ها طراحی نشده است اما به طرز شگفت‌انگیزی در کاراهایی مثل کدنویسی شبکه کارآمد است.
- **مستقل از بستر اجرا یا پلتفرم (Platform independent) :** پلتفرم عبارت است از محیط سخت‌افزاری و نرم‌افزاری که برنامه در آن اجرا می‌شود. پلتفرم به دو بخش تقسیم می‌شود:

۱. مبتنی بر نرم‌افزار ۲. مبتنی بر سخت‌افزار

جاوا با بستر اجرای نرم‌افزاری سروکار دارد. گفتنی است که پلت فرم جاوا با سایر پلتفرم‌ها تفاوت دارد. از این نظر که یک بستر اجرای نرم‌افزاری است که بر پایه‌ی سایر پلتفرم‌های مبتنی بر سخت‌افزار ساخته‌شده است و بر روی آن‌ها اجرا می‌شود. جاوا از دو کامپوننت تشکیل‌شده است که عبارت‌اند از:

۱. محیط اجرا (Runtime Environment)

۲. توابع کتابخانه‌ای (Application programming interface)

جالب است بدانید که کد جاوا بر روی چندین محیط قابل‌اجرا می‌باشد که ازجمله‌ی آن‌ها می‌توان به ویندوز، لینوکس، سولاریس و مک اشاره کرد. کامپایلر، کد جاوا را ابتدا ترجمه کرده و به بایت کد که یک زبان میانی است، تبدیل می‌کند. بایت کد یک کد مستقل از پلت فرم است که بر روی محیط‌های مختلف اجرا می‌شود. جاوا برای سیستم‌عامل‌های معروف ویندوز، لینوکس، مکینتاش و غیره شرکت اوراکل مفسرهای مخصوص را تولید کرده است. پس مستقل از پلتفرم بودن به این معناست که برنامه‌های جاوا بر روی سیستم‌عامل‌های معروف که شرکت اوراکل مفسر آن‌ها را ارائه کرده است، قابل‌اجرا هستند. درنتیجه، اگر شخص یا گروهی سیستم‌عاملی تولید کنند برنامه‌های جاوا روی آن‌ها اجرا نمی‌شوند (مگر این که مفسر آن نیز نوشته شود).

- **توزیع‌شده (Distributed):** جاوا زبانی است که امکاناتی برای تولید برنامه‌های کاربردی بزرگ و توزیع‌شده دارد. RMI و EJB دو فریم ورکی هستند که برای پیاده‌سازی برنامه‌های کاربردی توزیع شده به کار می‌روند. با ا ستفاده از جاوا می‌توان با فراخوانی متدها از هر د ستگاهی در اینترنت به فایل‌های موردنظر دسترسی داشته باشید.
- **معماری خنثی (Architecture neutral):** یعنی، نرم‌افزار تو سعه‌یافته با جاوا طوری طراحی شده است که از محیط یا بستر اجرایی که قرار است در آن نصب و راه‌اندازی شود، کاملاً مستقل است. به‌عنوان‌مثال، اندازه‌ی انواع داده‌ای اولیه این زبان در انواع محیط‌های سخت‌افزاری و نرم‌افزاری ثابت است. در زبان برنامه‌نویسی C، انواع داده‌ای(int عدد صحیح) در معماری 32 بیتی 2 بایت را به خود تخصیص می‌دهد و در معماری 64 بیتی 4 بایت را اشغال می‌کند. این در حالی است که نوع داده‌ای int جاوا در هر دو معماری مقداری یکسان از حافظه را اشغال می‌کند.
- **پویا (Dynamic) بودن:** زبان جاوا بسیار پویاتر از زبان‌های C++ و C است. چون این زبان طراحی‌شده است تا با محیط‌های گوناگون خود را وفق دهد.

۲-۱. مفهوم بسته و کاربردهای آن

جاوا کلاس‌های زیادی دارد. برای دسته‌بندی کلاس‌ها، تمام کلاس‌های مرتبط به هم در یک **بسته** [۲] قرار می‌گیرند. اصلی‌ترین بسته، **بسته System** است. این بسته، شامل کلاس object و تعدادی کلاس پایه دیگر است. کلاس object یک کلاس پایه است. کلاس‌هایی که در بسته‌ها وجود دارند، **کلاس‌های آماده** نام دارند. علاوه بر این کلاس‌ها، برنامه‌نویس می‌تواند کلاس‌های جدیدی را ایجاد کرده و از آن‌ها استفاده کند. چون، ممکن است کلاس‌های موجود در بسته‌ها همه نیازهای برنامه‌نویس را برطرف نکنند. چگونگی ایجاد این کلاس‌ها را در فصل‌های ۵ و ۶ می‌آموزیم. در این بخش می‌خواهیم به کلاس‌های موجود در بسته‌ها بپردازیم.

کلاس‌ها با توجه به کاربرد شان در بسته‌های مختلف قرار می‌گیرند. این عمل دو مزیت زیر را برای برنامه‌نویس در پی دارد:

۱. موجب دسته‌بندی کلاس‌ها می‌شود. یعنی، کلاس‌هایی که به هم مرتبط هستند، در یک بسته قرار می‌گیرند تا اولاً بتوان به‌راحتی آن‌ها را به پروژه اضافه نمود و ثانیاً بسته‌های که در پروژه به آن‌ها نیازی نیست، به پروژه اضافه نگردند.

۲. در کلاس‌های مختلف از نام‌های تکراری استفاده نمود. بسته و کلاس‌ها موجب می‌شوند تا نام‌های تکراری از یک‌دیگر تفکیک شوند.

بسته‌های مورد نیاز را می‌توانید به پروژه‌تان اضافه کنید. برای این منظور، می‌توانید از دستور import به‌صورت زیر استفاده نمایید:

```
import نام بسته;
```

به‌عنوان‌مثال، دستور زیر را ببینید:

```
import  java.lang;
```

این دستور بسته java.lang را به برنامه اضافه می‌کند تا بتوانید از کلاس‌های آن استفاده کنید.

چنانچه در ابتدای برنامه با دستور import بسته را اضافه نکنید، در کلیه مکان‌هایی که می‌خواهید از کلاس‌های موجود در آن بسته استفاده نمایید باید مسیر کامل بسته را ذکر کنید (به‌صورت زیر):

System.out.println

این دستور از متد println() کلاس out موجود در بسته System استفاده می‌کند.

> هر پروژه جدیدی که ایجاد می‌شود، یک بسته جدید همنام با پروژه نیز ایجاد می‌گردد.

۴ – ۱. آموزش زبان‌های برنامه‌نویسی

آموزش زبان‌های برنامه‌نویسی مانند زبان‌های طبیعی زنده دنیا است. یعنی، برای آموزش زبان‌های برنامه‌نویسی باید مراحل زیر را انجام داد:

۱. مانند هر زبان طبیعی ابتدا باید علائم تشکیل‌دهنده زبان را شناخت. به‌عنوان‌مثال، زبان فارسی از علائم الف تا ی، ارقام ۰ تا ۹ و علائم خاص مانند ! ، : ، ؟ و غیره تشکیل شده است. هرکدام از این علائم (نمادها) مفهوم خاصی دارند. زبان جاوا، نیز از علائم a تا z ، A تا Z ، ۰ تا ۹ ، علائم ویژه نظیر ; ، : ،] ، [، / و غیره تشکیل‌شده است. ابتدا باید مفاهیم هریک از این علائم را در زبان جاوا آموخت.

۲. همان‌طور که می‌دانید از ترکیب علائم هر زبان کلمات به وجود می‌آیند. برخی از کلمات دارای معنی و مفهوم هستند و برخی دیگر معنی و مفهوم خاصی ندارند. به‌عنوان‌مثال، کلمات **"بابا"**، **"آب"**، **"داد"**، در زبان فارسی مفهوم خاصی دارند. ولی کلمات **"تپتانم" و "بکیاپ"** مفهوم خاصی ندارند. به کلماتی که در زبان دارای مفهوم خاص هستند، **کلمات کلیدی** می‌گویند. در زبان جاوا کلمات کلیدی نظیر for، if، else، while و int وجود دارند. در آموزش یک زبان ابتدا باید کلمات کلیدی آن را شناخت. باید معنی و کاربرد هرکدام از آن‌ها را آموخت.

۳. در هر زبان طبیعی از ترکیب کلمات کلیدی با یک قواعد خاص، جمله ایجاد می‌شود (مانند جمله **"بابا آب داد"**). همان‌طور که می‌دانید در زبان فارسی ابتدا فاعل، سپس مفعول و در پایان فعل قرار می‌گیرد. در زبان جاوا نیز برای ایجاد جملات (دستورات) قواعد خاصی وجود دارد. به‌عنوان‌مثال، int برای تعریف داده‌های نوع صحیح به کار می‌رود و به‌صورت زیر استفاده می‌گردد:

int متغیر۱ , متغیر۲ , ... , متغیر n ;

۴. همان‌طور که می‌دانید، در زبان‌های طبیعی با کنار هم قرار گرفتن جملات مرتبط به هم پاراگراف ایجاد می‌شود. در زبان‌های برنامه‌نویسی نیز با کنار هم قرار دادن دستورات مرتبط به هم، بلاک ایجاد می‌شود. در زبان جاوا، هر بلاک با { شروع و با } خاتمه می‌یابد.

۵. چند پاراگراف صفحات و فصول را ایجاد خواهند کرد و این روند ادامه می‌یابد تا یک کتاب نوشته شود. در زبان‌های برنامه‌سازی نیز نوشتن برنامه‌ها هم همین روند را دارد. تعدادی بلاک، فایل، و چند فایل مرتبط به هم، برنامه را ایجاد می‌کنند.

در ادامه کتاب به آموزش زبان جاوا با این شیوه می‌پردازیم.

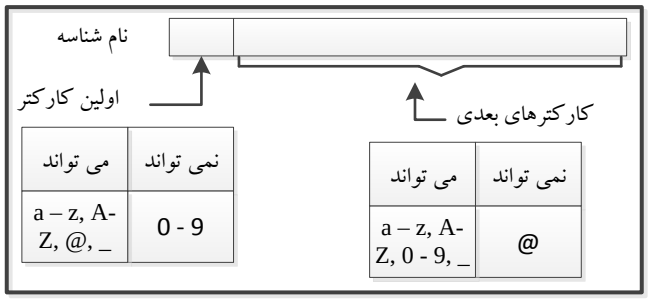
۴ – ۱. شناسه‌ها

شناسه‌ها[1]، نام‌هایی هستند که برنامه‌نویس برای عناصر جاوا از قبیل کلاس‌ها[2]، نام بسته‌ها[3]، متدها[4]، فیلدها[5]، خواص[6] و غیره انتخاب می‌کند. به‌عنوان‌مثال، (شکل ۱ – ۱) را مشاهده کنید. در این شکل هر کلمه‌ای که در داخل هاشور قرار دارد، شناسه است. قبل از استفاده از شناسه‌ها باید آن‌ها را نام‌گذاری نمود. قوانین نام‌گذاری شناسه‌ها در زیر آمده‌اند (شکل ۲ – ۱).

- کاراکترهای الفبایی (A تا Z، a تا z) و خط زیر (_) می‌توانند در هر مکان نام شناسه قرار گیرند.

```
package ch1_20;
import java.util.*;
public class Ch1_20 {
  public static void main(String[] args) {
    Scanner input = new Scanner(System.in);
    System.out.print("Enter a number[10 to 99]:");
    int num = input.nextInt();
    System.out.println((num/10) +"+"+ num % 10 + " = " +( num / 10 + num % 10));
  }
}
```

شکل ۱ – ۱ برخی از شناسه‌های جاوا در یک برنامه.



شکل ۲ – ۱ روش نام‌گذاری شناسه‌ها.

[1] .Identifiers [2].Classes [3].Namespaces [4].Methods [5]. Fields [6].Properties

- ارقام صفر تا ۹ نمی‌توانند در اولین مکان نام شناسه قرار گیرند، ولی می‌توانند در مکان‌های دیگر نام شناسه مورداستفاده قرار گیرند.
- کاراکتر @ می‌تواند در اولین مکان نام شناسه قرار بگیرد، اما، نمی‌تواند در مکان‌های دیگر نام شناسه قرار گیرد.
- نام شناسه نسبت به حروف بزرگ و کوچک حساس است. یعنی، شناسه‌های myVar و MYVar دو نام مختلف برای دو شناسه می‌باشند.

۵ – ۱. کلمات کلیدی

کلماتی که در زبان شناخته‌شده‌اند و مفهوم خاصی در آن زبان دارند، **کلمات کلیدی**[1] نامیده می‌شوند. برخی از کلمات کلیدی را در (شکل ۳ – ۱) می‌بینید. این کلمات در داخل ها شور قرار دارند. جاوا از کلمات کلیدی زیادی تشکیل می‌شود که برخی از آن‌ها را در جدول ۱ – ۱ می‌بینید (کلمات کلیدی در برنامه جاوا با رنگ آبی مشخص می‌شوند).

```
package ch1_20;
import java.util.*;
public class Ch1_20 {
  public static void main(String[] args) {
    Scanner input = new Scanner(System.in);
    System.out.print("Enter a number[10 to 99]:");
    int num = input.nextInt();
    System.out.println((num/10) +"+"+ num % 10 + " = " +( num / 10 + num % 10));
  }
}
```

شکل ۳ – ۱ برخی از کلمات کلیدی در برنامه جاوا.

۶ – ۱. فضای سفید

فضای سفید (whitespace)، کاراکترهایی هستند که قابلیت چاپ ندارند. این کاراکترها توسط کامپایلر نادیده گرفته می‌شوند، اما برنامه‌نویس برای افزایش خوانایی برنامه از این کاراکترها در برنامه‌اش استفاده می‌کند. برخی از این کاراکترها عبارت‌اند از : ۱. کاراکتر فضای خالی (space) ۲. کاراکتر Tab ۳. خط جدید (New Line) و ۴. کلید Enter (carriage return).

۷ – ۱. لیترال‌ها

لیترال‌ها[2]، داده‌هایی هستند که به‌صورت ثابت در کد برنامه‌تان وارد می‌کنید. لیترال‌ها می‌توانند مقادیر عددی، رشته‌ای (که در بین جفت کتیشن قرار می‌گیرند) یا منطقی (True یا False) باشند. در (شکل ۴ – ۱) برخی از لیترال‌ها را می‌بینید. در این شکل لیترال‌ها با هاشور نمایش داده شده‌اند.

---

[1].Keywords
[2].Literals

```
package ch1_20;
import java.util.*;
public class Ch1_20 {
  public static void main(String[] args) {
    Scanner input = new Scanner(System.in);
    System.out.print("Enter a number[10 to 99]:");
    int num = input.nextInt();
    System.out.println((num/10) +"+"+ num % 10 + " = " +( num / 10 + num % 10));
  }
}
```

شکل ۴ – ۱ برخی از لیترال‌ها در جاوا.

جدول ۱-۱ کلمات کلیدی.

abstract	assert	Boolean	break	byte	case	catch	char	class
continue	default	Do	double	else	enum	extends	final	finally
for	goto	If	implements	import	instanceof	int	interface	long
new	package	private	protected	public	return	short	static	strictfp
switch	this	throw	throws	transient	try	void	volatile	while
false	null	const	float	native	super	true		

۸ – ۱. توضیحات

توضیحات[1] توسط کامپایلر نادیده گرفته می‌شوند و موجب افزایش خوانایی برنامه می‌گردند (شکل ۵ – ۱). در این شکل توضیحات در داخل هاشور قرار دارند. توضیحات در برنامه با رنگ سبز پررنگ نمایش داده می‌شوند. به دو روش می‌توان توضیحات را به برنامه اضافه کرد:

۱. **کاراکترهای //**، تمام کلمات بعد از // **توسط** کامپایلر نادیده گرفته می‌شوند. یعنی، این کلمات توضیحات در نظر گرفته می‌شوند. به‌عنوان‌مثال، دستور زیر را در نظر بگیرید:

```
int i = 0;    //Define   I  and initial .
```

این دستور i را تعریف کرده، مقدار صفر را در آن قرار می‌دهد و در ادامه دستور، توضیح برای آن آمده است.

۲. **کاراکترهای * \ و / ***، توضیحات می‌توانند با کاراکترهای * \ شروع می‌شوند و با کاراکترهای / * خاتمه یابند. یعنی، تمام کلماتی که بین * \ و / * قرار می‌گیرند، توضیحات در نظر گرفته می‌شوند. با این روش می‌توان توضیحات چند سطری(خطی) نیز ایجاد نمود. به‌عنوان‌مثال، دستورات زیر را ببینید:

```
/*
This text is ignored by the compiler.
Unlike single-line comments, delimited comments
```

[1].Comments

```
like this one can span multiple lines.
*/
```

این دستورات توضیحات چند سطری را ایجاد می‌کنند.

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package javaapplication652;
/**
 *
 * @author Fansno_Net
 */
public class JavaApplication652 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
    }
}
```

شکل ۵ – ۱ توضیحات برای افزایش خوانایی برنامه.

در جاوا نمی‌توانید توضیحات تودرتو تعریف کنید. به‌عنوان‌مثال، دستورات زیر را ببینید:

```
/*This is an attempt at a nested comment.
? /*Ignored because it's inside a commentInner comment
? */Closes the comment because it's the first end
delimiter                    encountered
? */Syntax error because it has no opening delimiter*/
```

این دستورات موجب تولید خطا توسط کامپایلر خواهند شد. چون در جاوا نمی‌توان توضیحات تودرتو ایجاد کرد. اکنون دستورات زیر را ببینید:

```
//Single-line comment      /* Nested comment?
? /*Incorrect because it has no opening delimiter*/
```

این دستورات نیز موجب تولید خطا توسط کامپایلر خواهند شد.

۹ – ۱. کاراکترهای ویژه

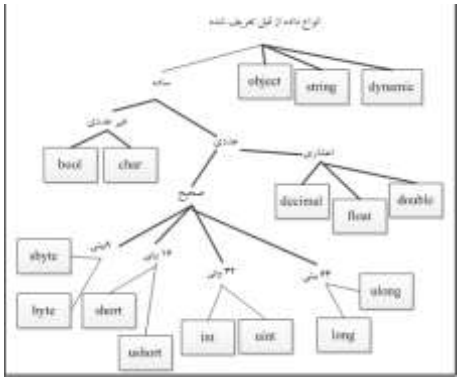
این کاراکترها برای نگه‌داری گروهی از اجزا یا جداکننده‌ها به کار می‌روند. برخی از این کاراکترها را در (شکل ۶ – ۱) می‌بینید. در این شکل، کاراکتر **";"** انتهای دستورات جاوا را مشخص می‌کند. کاراکتر }، بلاک جاوا را باز می‌کند و کاراکتر {، بلاک بازشده جاوا را می‌بندد. هر بلاک در جاوا با کاراکتر } شروع و با کاراکتر { خاتمه می‌یابد. در جاوا کاراکترهای } و { می‌توانند به‌صورت تودرتو به کار روند(شکل ۶ – ۱ را ببینید).

۱۰ – ۱. انواع داده

در جاوا، دو نوع داده مقدار و ارجاع وجود دارند. انواع داده‌های جاوا را در (شکل ۷ – ۱) می‌بینید. خلاصه این داده‌ها در جدول ۲ – ۱ آمده‌اند.

```
package ch1_20;
import java.util.*;
public class Ch1_20 {
  public static void main(String[] args) {
    Scanner input = new Scanner(System.in);
    System.out.print("Enter a number[10 to 99]:");
    int num = input.nextInt();
    System.out.println((num/10) +"+"+ num % 10 + " = " +( num / 10 + num % 10));
  }
}
```

شکل ۶ – ۱ نمایش کاراکترهای خاص.



شکل۷ – ۱ انواع داده‌های جاوا.

انواع مقدار

متغیر نوع مقدار (value-type)، به‌طور مستقیم داده‌اش را نگه‌داری می‌کند. یعنی، محتوی متغیر نوع مقدار، مقدارش است. به‌عنوان‌مثال، دستور زیر مقدار ۲۵ را به متغیری به نام i نسبت می‌دهد:

```
int   i = 25;
```

این عمل به شکل مقابل انجام می‌شود:

نوع	نام	مقدار
int	i	25

وقتی متغیر نوع مقدار ذخیره می‌شود، جاوا نوشته‌ای از نوع، نام شناسه و مقدار را نگه‌داری می‌کند و هنگامی که متغیر نوع مقدار را کپی می‌کنید، متغیر دوم جداگانه‌ای با همان نوع و مقدار ایجاد می‌شود. به‌عنوان‌مثال، دستور زیر کپی i را در j نشان می‌دهد:

```
int   j = i;
```

این عمل به شکل مقابل انجام می‌گردد:

نوع	نام	مقدار
int	i	25
int	j	25

این متغیرها ارتباطی باهم ندارند. اگر مقدار متغیری عوض شود، متغیر دیگر مقدار قبلی‌اش را حفظ می‌کند. به‌عنوان‌مثال، دستور مقابل را ببینید:

```
i = 50;
```

این دستور، مقدار i را به ۵۰ تغییر می‌دهد، اما، مقدار j همان ۲۵ باقی خواهد ماند (شکل مقابل را مشاهده کنید).

نوع	نام	مقدار
int	i	50
int	j	25

تمام انواع تعریف‌شده در جدول ۲ – ۱ (به‌جز نوع object و string) متغیرهای از نوع مقدار را تعریف می‌کنند.

جدول ۲-۱ انواع داده‌های جاوا.

نام	هدف	محدوده	مقدار پیش‌فرض
sbyte	عدد صحیح ۸ بیتی با علامت	-128….127	**0**
byte	عدد صحیح ۸ بیتی بدون علامت	0….255	**0**
short	عدد صحیح ۱۶ بیتی با علامت	-32768…32767	**0**
ushort	عدد صحیح ۱۶ بیتی بدون علامت	0….65535	**0**
int	عدد صحیح ۳۲ بیتی با علامت	-2147483648… 2147483647	**0**
uint	عدد صحیح ۳۲بیتی بدون علامت	0….4294964295	**0**
long	عدد صحیح ۶۴ بیتی با علامت	-9223372036854775808… 922337203685477807	**0**
ulong	عدد صحیح ۶۴ بیتی بدون علامت	0… 18446744073 709551615	**0**
float	عدد اعشاری با دقت معمولی	105×10^{-45} ... 3.4×10^{38}	0.0f
double	عدد اعشاری با دقت مضاعف	5×10^{-324} ... 1.7×10^{-308}	0.0d
boolean	نوع منطقی	True, False	False
char	کاراکتر یونیکد،	U+0000 …. U+ffff	\x0000
decimal	عدد ده‌دهی با ۲۵ رقم اعشار	$\pm 1.0 \times 10^{28}$ ... $\pm 7.9 \times 10^{28}$	0m
object	کلاس پایه‌ای که همه انواع دیگر از آن مشتق می-شوند	-----	null
string	مجموعه‌ای از کاراکترهای یونیکد	------	null

انواع ارجاع

انواع ارجاع (Reference Types) به دو شکل یک شیء و ارجاع به آن شیء می‌باشند. به‌عنوان‌مثال، دستورات زیر را ببینید:

```
StringBuilder  obj1 = new StringBuilder ("نوع ارجاع");
```

نوع StringBuilder برای تعریف رشته‌ای از کاراکترها به کار می‌رود. این دستور به شکل زیر عمل می‌کند:

مقدار	نوع
نوع ارجاع	StringBuilder

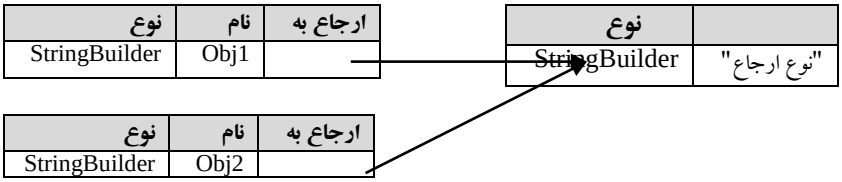
ارجاع به	نام	نوع
	Obj1	StringBuilder

همان‌طور که در این شکل می‌بینید، نمی‌توان به‌طور مستقیم به شیء و مقدار آن دستیابی داشت. برای انجام این کار باید از طریق ارجاع استفاده نمایید. در این شکل متغیر ارجاع و شیء را مشاهده می‌کنید. ارجاع، شامل نوع متغیر، نام آن و پیوند که آن را به شیء ارجاع می‌دهد و شیء شامل مقدار است.

وقتی‌که متغیری با نوع ارجاع را کپی می‌کنید، کپی جدیدی از ارجاع ایجاد خواهد شد، اما، یک شیء جدید نیست. به‌عنوان‌مثال، دستور زیر را مشاهده کنید:

```
StringBuilder obj2 = obj1;
```

این د ستور، بیان می‌کند که obj2 و obj1 هر دو شی به مقدار "نوع ارجاع" ا شاره می‌کنند(شکل زیر را ببینید):



همان‌طور که در این شـــکل می‌بینید، فقط یک شـــیء از نوع StringBuilder داریم، اما دو ارجاع به نام‌های obj2 و obj1 داریم که به آن اشـاره می‌کنند و برخلاف نوع مقدار، مقدار هر دو ارجاع یکی اسـت (یعنی، مقدار "نوع ارجاع" را دارند). با تغییر مقدار یکی، مقدار دیگری نیز تغییر خواهد کرد. جاوا دو نوع ارجاع اولیه به نام‌های String و object دارد.

دو ارجاع به یک شیء می‌توانند انواع مختلف داشته باشند.

متغیرهای نوع ارجاع را می‌توان طوری تعریف کرد که به‌جایی اشاره نکنند. به‌عنوان‌مثال، دستور زیر را در نظر بگیرید:

```
StringBuilder    obj = null;
```

نوع	نام	ارجاع به
StringBuilder	Obj	Null

دستورات

دستورات در زبان جاوا دو نوع‌اند:

۱. **د ستورات ساده**، هر دسـتور که با یک **;** خاتمه می‌یابد، **د ستور ساده** (Simple statement) نام دارد. به‌عنوان‌مثال، دستور زیر را مشاهده کنید:

```
int    i = 5 ;
```

این دستور، متغیر i را با نوع int تعریف کرده، مقدار اولیه ۵ را در آن قرار می‌دهد.

۲. **بلاک**، مجموعه‌ای از صفر یا چند د ستور مرتبط به هم که در داخل بلاک باز ({) و بلاک بسته (}) قرار می‌گیرند، را **بلاک** گویند.

به‌عنوان‌مثال، دستورات مقابل را ببینید:

```
{
  int   i = 5;
  int   j = 10;
   …
}
```

این دستورات تشکیل بلاک را می‌دهند.

۱۱–۱. مراحل آماده‌سازی و اجرای برنامه

مراحل آماده‌سازی و اجرای برنامه جاوا در زیر آمده است:

۱. **تایپ برنامه** : اولین گام برای آماده‌ســازی برنامه تایپ آن می‌باشـد. برای این منظور باید برنامه را در یک ویراســتار متنی تایپ کرده و آن را بر روی دیســک با پســوند Java ذخیره نمایید. در این کتاب ویراســتار NetBeansIDE… برای تایپ و ذخیره فایل جاوا استفاده شده است.

۲. **کامپایل :** دومین مرحله، کامپایل کردن فایل جاوا می‌باشد. کامپایلر فایل با پسوند java را دریافت می‌کند و کامپایل می‌نماید. در این مرحله اگر خطای گرامری در برنامه وجود نداشته باشد، فایلی با همان نام و با پسوند class ایجاد خواهد شد. فایل با پسوند class حاوی بایت کد است. **فایل بایت کد،** حاوی دستوراتی است که ماشین مجازی جاوا می‌تواند آن را اجرا کند. در این کتاب این مرحله نیز با استفاده از NetBeans IDE... انجام می‌شود.

۳. **بار کردن کلاس** : در این مرحله برنامه برای اجرا باید بایت کدها موجود در فایل class را خوانده آن را در حافظه اصلی قرار دهد. این کار توسط برنامه‌ای به نام **بارگذار کلاس** (یکی از اجزای اصلی ماشین مجازی است) انجام می‌شود. در این مرحله چنانچه بایت کدها نیاز به بایت کدهای کلاس دیگر داشته باشند، بارگذار کلاس‌های مورد نظر را نیز خوانده و آن‌ها را در حافظه اصلی قرار می‌دهد.

۴. **اعتبار سنجی بایت کدها :** یکی دیگر از اجزای ماشین مجازی اعتبار سنجی کلاس است. در این مرحله اعتبار سنجی کلاس فعال می‌گردد تا بایت کدهای که در حافظه قرار دارند را بررسی کند و تعیین کند تا از قوانین جاوا تخلف نکرده باشند. چنانچه هیچ تخلفی موجود نباشد، برنامه برای اجرا آماده می‌گردد.

۵. **تبدیل بایت کد به دستورات زبان ماشین،** در این مرحله ماشین مجازی جاوا بایت کدهایی را که اعتبار سنجی شده‌اند و به تایید رسیده‌اند، به زبان ماشینی که بر روی آن قرار دارد، ترجمه کرده و آن‌ها را اجرا می‌نماید.

فرآیند تایپ و اجرای برنامه جاوا در مثال ۱-۱ آمده است.

۱۲-۱. ساختار برنامه جاوا

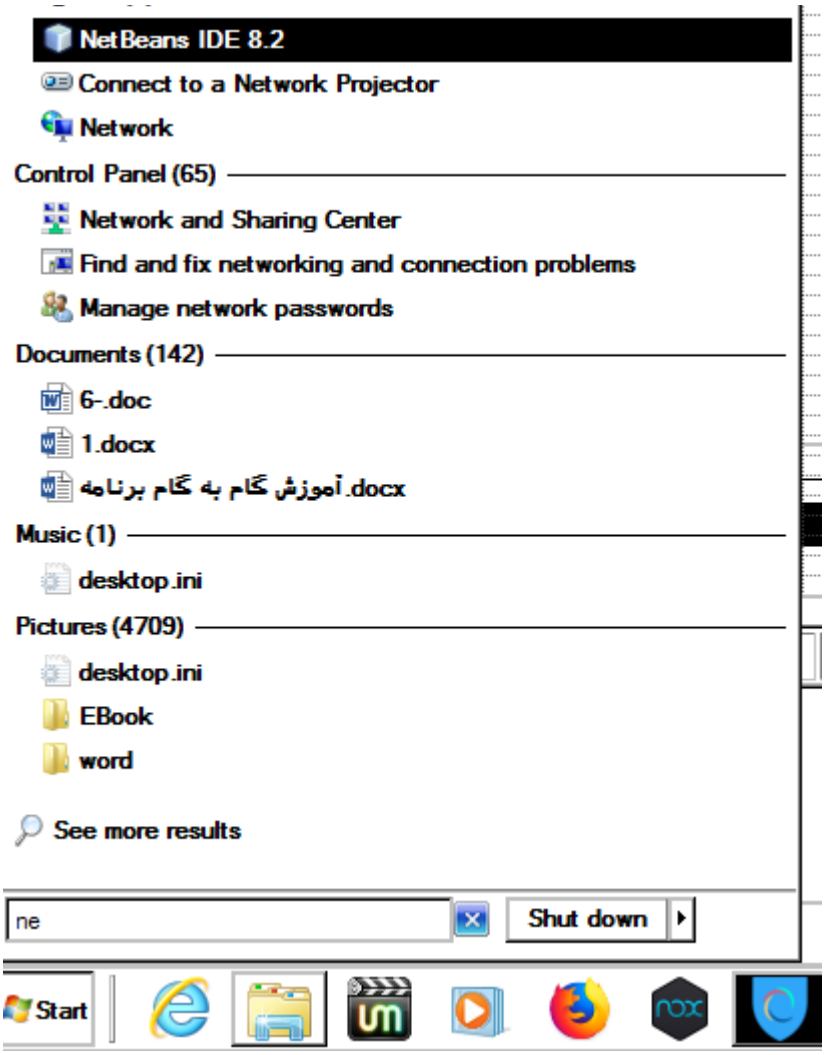
برای این‌که با ساختار برنامه جاوا آشنا شویم، یک برنامه جدید ایجاد می‌کنیم و از روی برنامه جدید ساختار برنامه را می‌آموزیم. برای ایجاد برنامه جدید، مثال ۱-۱ را ببینید.

مثال ۱- ۱. برنامه‌ای که مراحل ایجاد و اجرای یک برنامه در جاوا را نشان می‌دهد (هدف این برنامه آشنایی با ایجاد و اجرای برنامه در جاوا است).

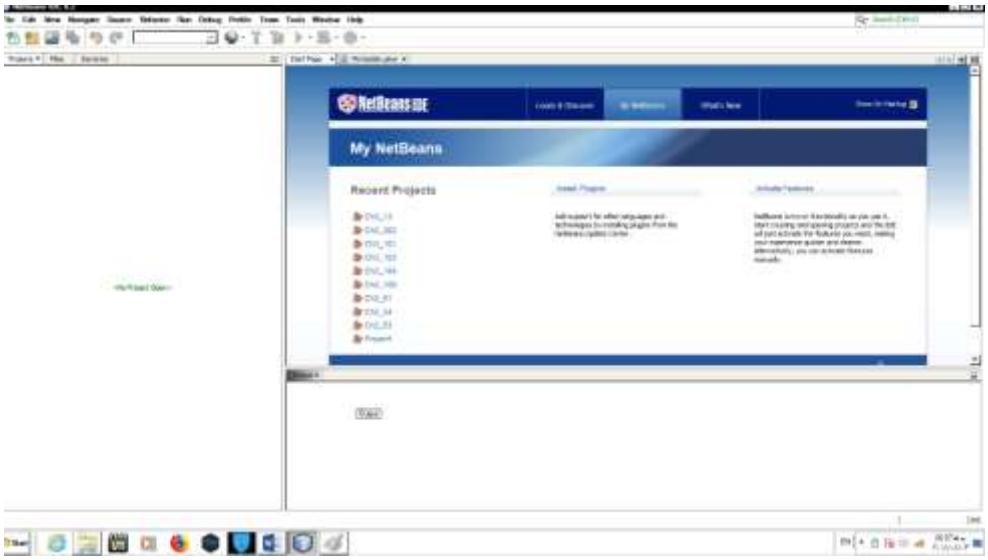
مراحل طراحی و اجرا

۱. نرم‌افزار جاوا را اجرا کنید. برای این منظور، در منوی Start نرم افرار ... NetBeans IDE را پیدا کرده (مانند شکل ۷-۱) و اجرا کنید.

۲. اکنون صفحه اول نرم افرار ... NetBeans IDE ظاهر می‌شود(شکل ۸ - ۱).

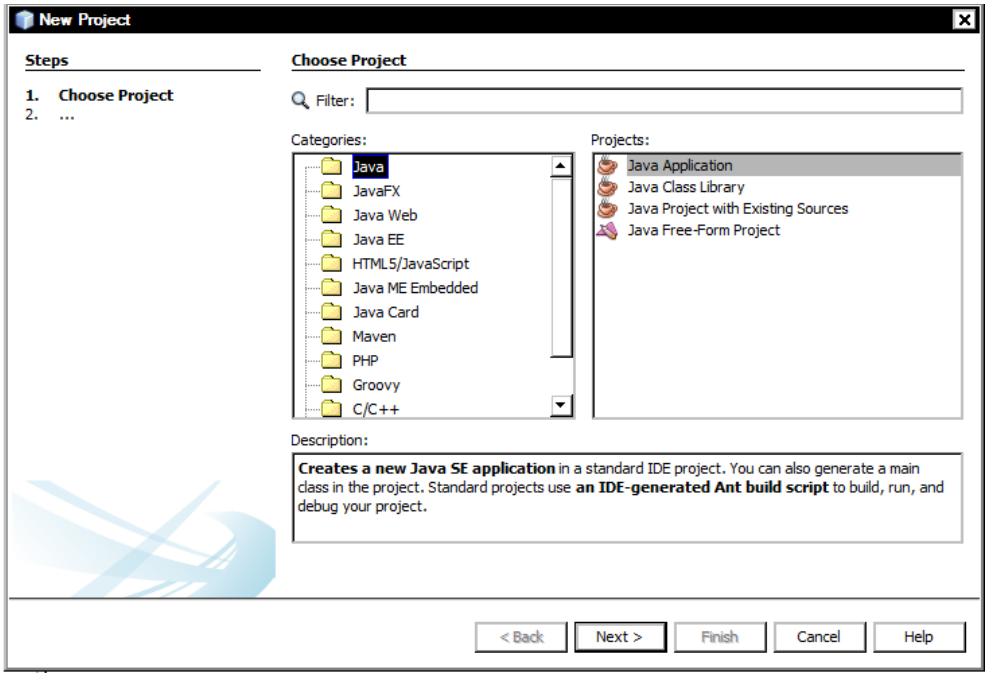


شکل ۷-۱ جست‌وجو و اجرای نرم افزار نرم افرار ... NetBeans IDE.



شکل ۸-۱ نرم افزار نرم افزار ... NetBeans IDE.

۳. گزینه File/New /Project (کلید های Ctrl+Shift+N) را اجرا کنید تا پنجره New Project ظاهر شود (شکل ۹ - ۱). در این پنجره اطلاعات زیر را انتخاب کنید:



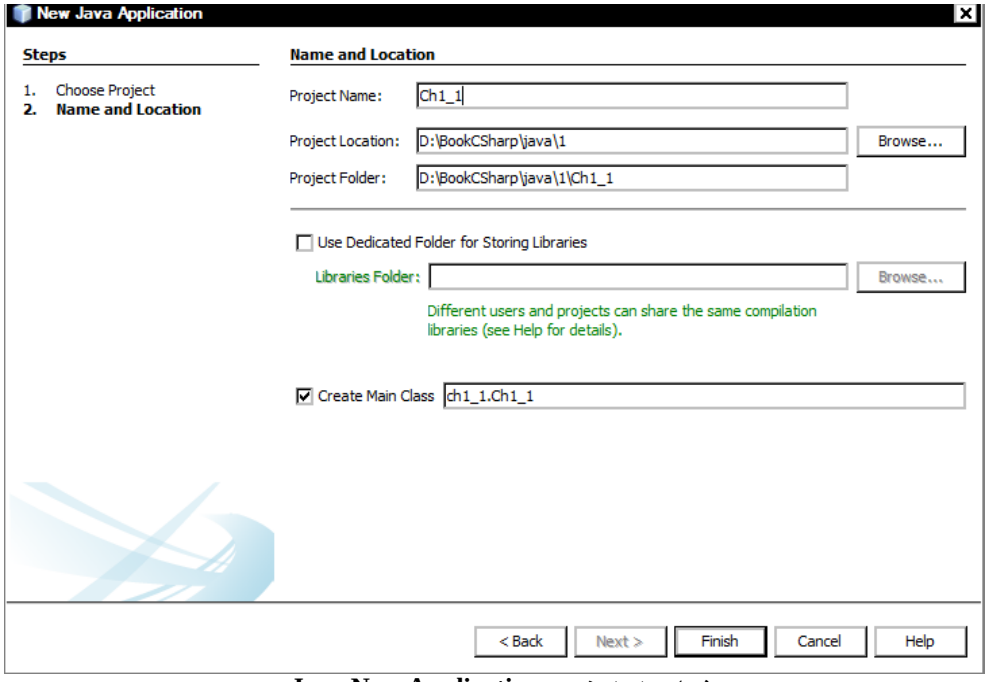
شکل ۹-۱ New Project.

در سمت چپ (بخش Categories)، نوع برنامه را انتخاب کنید. در این برنامه، Java را انتخاب نمایید.

- در پنجره ســـمت راســـت، نوع برنامه را انتخاب نمایید. در این برنامه نوع Java Application را انتخاب کنید.

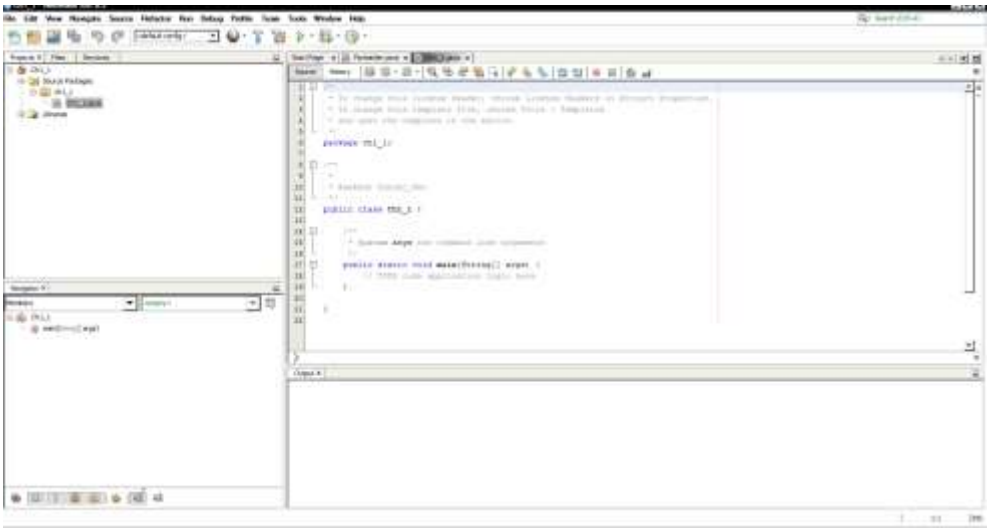
۱. دکمه Next را کلیک کنید تا شکل ۱۰-۱ ظاهر شود.

- در جلوی Name، نام پروژه را وارد کنید. در این برنامه Ch1-1 انتخاب‌شده است.
- در بخش Location مکان ذخیره برنامه را انتخاب کنید. در این برنامه D:\BookCSharp\Java\1 انتخاب گردید.



شکل ۱۰-۱ پنجره Java New Application

۴. دکمه Finsh را کلیک کنید تا اولین برنامه جاوا ایجاد شود(شکل ۱۱- ۱).



شکل ۱۱–۱ اولین برنامه جاوا.

۵. دستورات این برنامه به‌صورت زیر می‌باشند:

```
/*
   * To change this license header, choose License Headers in
     Project Properties.
   * To change this template file, choose Tools | Templates
   * and open the template in the editor.
 */
package ch1_1;
 /**
  *
  * @author Fansno_Net
  */
public class Ch1_1 {
    /**
     * @param args the command line arguments
     */
   public static void main(String[] args) {
        // TODO code application logic here
    }
}
```

این برنامه دارای ۵ بخش اصلی است:

۱. بخش اضافه کردن بسته موردنیاز برنامه، این بخش بسته‌های موردنیاز برنامه را معرفی می‌کند. این دستورات با دستور import شروع می‌شوند (دستورات زیر):

```
/*
   * To change this license header, choose License Headers in
     Project Properties.
   * To change this template file, choose Tools | Templates
   * and open the template in the editor.
 */
```

۲. تعریف بسته پروژه، هر برنامه جدیدی که ایجاد می‌کنید، بسته جدیدی به نام پروژه ایجاد می شود. در این برنامه این بسته با دستور زیر ایجاد گردید:

```
package ch1_1;
```

۳. تعریف کلاس برنامه، هر برنامه جدیدی که ایجاد می‌کنید یک کلاس به نام پروژه ایجاد می شود. از طریق این کلاس می‌توان **فیلدها، متدها، واسط‌ها** [1] را ایجاد کرد. با مفهوم کلاس در فصل‌های ۵ و ۶ بیش‌تر آشنا خواهید شد.کلاس Ch1_1 با دستور زیر ایجاد گردید:

```
public class Ch1_1 {
```

۴. متد main()، یکی از مهم‌ترین متدهای کلاس برنامه می‌باشد (وجود متد main() در تمام برنامه‌های اجرایی ضروری است). این متد به‌صورت زیر تعریف‌شده است:

```
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
    }
```

همان‌طور که در این دستور مشاهده می‌کنید، متد main() با کلمه کلیدی static تعریف شده است. این کلمه Modifier نام دارد. کلمه کلیدی static بیان می‌کند که این متد فقط در همین کلاس قابل‌اجرا می‌باشد و از طریق هیچ **نمونه** یا آبجکت [2] دیگری قابل‌اجرا نمی‌باشد (در فصل‌های ۵ و ۶ با متدهای static بیش‌تر آشنا خواهید شد). کلمه void در این متد تعیین می‌کند که این متد هیچ مقداری را برگشت نمی‌دهد. با تعریف متدها در فصل سوم بیش‌تر آشنا خواهید شد. کلمه args، آرگومان‌هایی را تعیین می‌کنند که از طریق خط فرمان می‌توان برای متد main() ارسال کرد.

۵. دستورات متد main()، در این برنامه دستورات متد main() به‌صورت زیر می‌باشند:

```
{
}
```

چون این برنامه هیچ عملی را انجام نمی‌دهد، بنابراین هیچ دستوری ندارد.

۶. پروژه را ذخیره کنید. برای این منظور، گزینه File/Save All را اجرا نمایید.

۷. پروژه را اجرا کنید. برای این منظور، یکی از اعمال زیر را انجام دهید:

- کلید F6 را فشار دهید.
- گزینه Run/Run Project را اجرا نمایید.
- بر روی آیکن پروژه کلیک راست کنید و از منویی که ظاهر می‌شود، گزینه Run را اجرا کنید.
- دکمه ▶ را کلیک کنید.

درهرصورت پروژه اجراشده، خروجی زیر نمایش داده می‌شود (شکل زیر):

[1].Interpace [2].Instance

۱۳–۱. دستورات خروجی

کنسول ویندوز، خط فرمان ساده ویندوز است که اجازه می‌دهد یک برنامه متنی را نمایش داده و داده‌ها را از صفحه‌کلید دریافت کند. برای انجام این کار در جاوا کلاسی به نام Console (در بسته System) وجود دارد که شامل متدهای برای ورودی و خروجی داده در یک کنسول ویندوز است. این کلاس دارای متدهای مختلفی است که متدهای ورودی و خروجی را در ادامه می‌بینید.

متدهای خروجی

کلاس out دارای دو متد برای نمایش داده در خروجی است. این دو متد عبارت‌اند از:

جدول ۳- ۱ کاراکترهای کنترلی.

کاراکتر	کد اسکی	هدف
\’	0x0027	کاراکتر تک کتیشن (‘) را تعیین می‌کند.
\”	0x0022	کاراکتر جفت کتیشن (“) را تعیین می‌کند.
\\	0x005C	کاراکتر بک‌اسلش (Backslash) را تعیین می‌کند.
\0	0x0000	کاراکتر تهی (null) را تعیین می‌کند.
\a	0x0007	بوق سیستم را به صدا در می‌آورد.
\b	0x0008	کاراکتر Backspace را تعیین می‌کند.
\f	0x000C	مکان‌نما را به صفحه بعدی انتقال می‌دهد.
\n	0x000A	مکان‌نما را به سطر بعدی انتقال می‌دهد.
\r	0x000D	کاراکتر Carrage return را تعیین می‌کند.
\t	0x0009	مکان‌نما را به Tab افقی بعدی انتقال می‌دهد.
\v	0x000B	مکان‌نما را به Tab عمودی بعدی انتقال می‌دهد.

متد print()، برای نمایش داده در خروجی به کار می‌رود. این متد به‌صورت زیر استفاده می‌شود:

System.out.print(مقدار قابل چاپ)

همان‌طور که در این متد می‌بینید، متد print چند پارامتر را می‌پذیرد.

متد println()، همانند متد print() است. با این تفاوت که پس از نمایش اطلاعات در خروجی مکان‌نما را به سطر بعد خروجی انتقال می‌دهد. به‌عنوان‌مثال، دستورات زیر را ببینید:

```
Sustem.out.print("One ");
Sustem.out.print("Two ");
```

این دستورات خروجی مقابل را نمایش می‌دهند:

```
One  Two
```

اکنون دستورات زیر را ببینید:

```
Sustem.out.println("One ");
Sustem.out.println("Two ");
```

این دستورات خروجی زیر را نمایش می‌دهند:

```
One
Two
```

مثال ۲–۱. برنامه‌ای که عبارت First output in Java را نمایش می‌دهد.

مراحل طراحی و اجرا

۱. پروژه جدیدی به نام Ch1_2 ایجاد کنید. برای این منظور، گزینه File/New Project را اجرا نمایید تا پنجره New Project ظاهر شود. در این پنجره، نوع برنامه را Java Application انتخاب کرده، دکمه Next را کلیک نمایید و در صفحه بعدی نام پروژه را Ch1_2 انتخاب کرده، دکمه Finsh را کلیک کنید.

۲. دستورات پروژه را به‌صورت زیر تغییر دهید:

```
package ch1_2;
public class Ch1_2 {
    public static void main(String[] args) {
       System.out.println("Frist output in Java");
    }
 }
```

دستور داخل متد main() عبارت First output in Java را نمایش می‌دهد.

۳. پروژه را ذخیره و اجرا کنید تا خروجی را به صورت زیر مشاهده نمایید:

```
run:
Frist output in Java
BUILD SUCCESSFUL (total time: 0 seconds)
```

توجه کنید که در خروجی مثال‌های بعدی سطر اول (run:) و سطر آخر(BUILD SUCCESSFUL (total time: 0 seconds) را حذف می‌کنیم.

مثال ۳–۱. برنامه‌ای که کاربرد کاراکترهای کنترلی را در متدهای print() println() نمایش می‌دهد.

مراحل طراحی و اجرا

۱. پروژه جدیدی به نام Ch1_3 ایجاد کنید. برای این منظور، گزینه File/New Project را اجرا نمایید تا پنجره New Project ظاهر شود. در این پنجره، نوع برنامه را Java Application انتخاب کرده، دکمه Next را کلیک نمایید و در صفحه بعدی نام پروژه را Ch1_3 انتخاب کرده، دکمه Finsh را کلیک کنید.

۲. دستورات پروژه را به‌صورت زیر تغییر دهید:

```
package ch1_3;
public class Ch1_3 {
  public static void main(String[] args) {
       System.out.println("One\t" + 1 + "\nTwo\t" + 2 + "\t");
       System.out.println("Fanavarienovin\twww.fanavarienovin.net
          \nKetabrah\twwww.ketabrah.ir\\");
       System.out.print("Three\t" + 3 + "\tFour\t" + 4 +"\'\n");
    }
}
```

اولین دستور تابع main() متد print() کلاس out، ابتدا عبارت One را می‌نویسد و با کاراکتر کنترلی '\t' به تب بعدی می‌رود، سپس مقدار ۱ را نمایش می‌دهد، در ادامه با کاراکتر '\n' کنترل چاپ را به سطر بعدی انتقال می‌دهد و کلمه Two را نمایش می‌دهد. در پایان با کاراکتر '\t' کنترل چاپ به تب بعدی می‌رود و ۲ را نمایش می‌دهد. کنترل چاپ به تب بعدی انتقال می‌یابد. دستور دوم، ابتدا کلمه Fanavarienovin را نمایش می‌دهد، کنترل چاپ را به تب بعدی انتقال می‌دهد و مقدار www.fanavarienovin.net را نمایش می‌دهد، سپس کنترل چاپ را با کاراکتر '\n' به سطر بعدی انتقال می‌دهد، کلمه ketabrah را نمایش داده، با کاراکتر '\t' کنترل چاپ را به تب بعدی منتقل کرده و عبارت www.ketabrah.ir را نمایش می‌دهد، در پایان، کاراکتر \ را نمایش داده (کاراکتر \\ برای چاپ کاراکتر \ به کار می‌رود) و کنترل چاپ را به سطر بعدی می‌برد (چون، از

متد println() استفاده شده است). دستور سوم، ابتدا کلمه Three را نمایش داده و کنترل چاپ را به تب بعد انتقال می‌دهد، سپس مقدار ۳ را نمایش می‌دهد و کنترل چاپ را به تب بعدی انتقال داده، Four را نمایش می‌دهد، در پایان، کنترل چاپ را به تب بعدی انتقال داده، مقدار ۴ را نمایش می‌دهد و بالاخره ' (کاراکتر '\ کوتیشن) را نمایش می‌دهد و به سطر بعدی می‌رود.

۳. پروژه را ذخیره و اجرا کنید تا خروجی را به صورت زیر مشاهده نمایید:

```
One   1
Two   2       Fanavarienovin   www.fanavarienovin.net
Ketabrah      wwww.ketabrah.ir\
Three 3       Four    4'
```

۱۴-۱. متغیر

برای نگه‌داری هر چیزی لازم است که از یک ظرف متناسب با آن استفاده نمود. به‌عنوان‌مثال، در خانه برای نگه‌داری مواد غذایی، ظروف مختلفی وجود دارند که هرکدام برای نگه‌داری مواد خاصی به کار می‌روند. مثلاً، بطری برای نگه‌داری آب. به همین ترتیب در برنامه‌نویسی، برای نگه‌داری مقادیر از ظروف مخصوصی استفاده می شود. ظرف نگه‌داری داده در زبان‌های برنامه‌نویسی **متغیر** نام دارد. **بنابراین، متغیر نامی است برای یک مکان از حافظه که ممکن است که در طول اجرای برنامه مقدار آن تغییر کند. ولی، در یک‌لحظه فقط یک مقدار را دارد.**

برای استفاده از متغیرها سه عمل باید انجام شود که عبارت‌اند از:

۱. نام‌گذاری متغیرها

همان طور که می‌دانید، بعدازاین که یک بچه به دنیا آمد، برای شناسایی او نامی انتخاب می‌کنید. جهت مراجعه به متغیرها نیز از نام آن‌ها استفاده می‌شود. برای نام‌گذاری بچه‌ها ثبت‌احوال از قوانینی خاصی پیروی می‌کند، به‌عنوان‌مثال، اجازه نمی‌دهد نام بچه را **"رضا 1"** انتخاب کنید. در زبان جاوا نیز برای نام‌گذاری متغیرها قوانین نام گذاری شناسه پیروی می‌کند. برخی از نام های مجاز برای متغیر عبارت‌اند از: area، sum، sum1، pr_1 و ... . اما، نام‌های زیر برای متغیر مجاز نیستند:

- **نام 2test :** نام متغیر نمی‌تواند با ارقام 0 تا 9 شروع شود.
- **نام test$** : در نام متغیر نمی‌توان از $ استفاده کرد.
- **نام store 2** : در نام متغیر نمی‌توان از کاراکتر فاصله[1] استفاده کرد.
- **نام .jpg :** نام متغیر نمی‌تواند با کاراکتر نقطه (.) شروع شود.

۲. معرفی متغیرها

همان‌طور که بیان گردید، هر ظرفی برای نگه‌داری نوعی غذا به کار می‌رود. بنابراین، متغیرها نیز باید دارای نوع باشند تا بتوانند انواع داده‌ها را ذخیره کنند. چون داده‌ها دارای انواع مختلف هستند، بنابراین، متغیرها که داده‌ها را نگه‌داری می‌کنند، باید دارای نوع باشند. نوع متغیر تعیین می‌کند اولاً چه نوع داده‌ای می‌تواند در آن متغیر قرار گیرد و ثانیاً، این متغیر به چند بایت از حافظه نیاز دارد. تعیین نوع متغیر به‌صورت زیر می‌باشد:

نوع داده‌ای لیست متغیرها ;

---

[1] .blank(space)