مرجع کامل

JAVA

مولف

هربرت شیلد

مترجمان

الناز قاسمی

ایمان سلیمانی

«تحت نظارت و سرپرستی مهندس جواد قنبر»

نشردانشگاهی کیان

Kian Publication

سخنی با خوانندگان

«سپس، به کاتبان و نویسندگانت بنگر و بهترین آن‌ها را بر کارهای خود بگمار...
کاتبان و نویسندگانی برگزین که قدر خود را بشناسند، چون کسی که به قدر خود شناخت ندارد، دیگران را هم نمی‌شناسد.»

«برگرفته از نامه‌ی ۵۳ نهج‌البلاغه به مالک‌اشتر»

اگرچه نوشتن و پرداختن زکات علم از توصیه‌های اکید بزرگان و گواه بر کرامت اهل دانش است، اما امروزه پرداختن به انگیزه‌ها و اهداف نوشتن بیشتر جلوه می‌کند. بی‌شک این‌که چه کسی می‌نویسد مهم نیست، اما این‌که چرا و به چه پشتوانه‌ای می‌نویسد، درخور تأمل است. ما معتقدیم که چاپ روزافزون کتاب‌های به اصطلاح «زرد» که خالی از هرگونه نوآوری و بی‌توجه به استانداردهای چاپ کتاب و نیازهای مخاطبان است، حاصل تفکر بازاری مستولی بر جامعه‌ی نشر است. بی‌پرده آن‌که عنوان پر زرق و برق، دستاویز قرار دادن مضمون‌های نو با هدف فروش بالا و طویل کردن سیاهه‌ی سابقه‌ی علمی، نمی‌تواند دلیل محکمی برای چاپ و نشر کتابی باشد که خواننده‌ی مشتاق با صرف هزینه‌های نه چندان کم آن را تهیه می‌کند؛ به امید آن که چیزی از آن بیاموزد.

باید پذیرفت که انگیزه‌ی نوشتن کم از محتوای نوشته نیست و بین این دو رابطه‌ای مستقیم برقرار است. اگر انگیزه از نوشتن، تولید دانش باشد، بی‌شک نویسنده از قلم بی‌محتوا و کم‌عمق پرهیز می‌کند و اگر دغدغه‌ی دانش و فرهنگ زخم‌خورده در میان باشد، ناشر تنها به عنوان پرطمطراق بسنده نمی‌کند.

و چقدر امروزه، فرهنگ و دانش این مرزوبوم که گرفتار آفت بی‌انگیزگی و زخم هوس است، نیازمند ناشران و نویسندگانی است که نیت‌شان کمک به رشد دانش و ارتقای فرهنگ جامعه است و به راستی که التیامی بر این درد نیست مگر نویسندگانی که قدر خود و دیگران را می‌دانند و خوب می‌فهمند که کتاب، ابزار سودجویی‌های مغرضانه نیست و می‌کوشند تا خود را از هرگونه شهوت نام و رسم و ثروت تهی کنند.

انتشارات دانشگاهی کیان خود را بری از عیب و خطا نمی‌داند، اما همواره بیش از پیش می‌کوشیم تا در راستای تولید علم و نشر کتاب‌های پرمحتوا، دست نویسندگانی را که انگیزه‌ی پاک دارند فشرده و در کنارشان باشیم و از خداوند متعال می‌خواهیم که در این مسیر صعب و پرخطر در سایه‌ی لطف و عنایت خود از آن‌چه به عهده‌ی ما نهاده شده، سربلند و پیروز برآییم.

انتشارات دانشگاهی کیان

درباره‌ی نویسنده

نویسنده‌ی این کتاب، هربرت شیلد[1]، در زبان‌های جاوا، C++، C و C# مهارتی ویژه دارد. میلیون‌ها نسخه از کتاب‌های وی در زمینه‌ی برنامه‌نویسی در سراسر جهان به فروش رفته و به تمام زبان‌های اصلی دنیا ترجمه شده است. از او کتاب‌های متعددی درباره‌ی زبان برنامه‌نویسی جاوا به چاپ رسیده است از جمله جاوا، راهنمایی برای افراد مبتدی کتاب هربرت شیلد درباره‌ی زبان برنامه‌نویسی جاوا و کتاب هنر جاوا.

از بین پرتیراژترین کتاب‌های دیگر وی نیز می‌توان به کتاب‌های مرجع کامل C++، مرجع کامل C# و مرجع کامل زبان C اشاره کرد. علی‌رغم علاقه‌مندی به تمام زمینه‌های مربوط به علوم رایانه‌ای، بیشترین تمرکز وی بر روی زبان‌های برنامه‌نویسی و موارد مربوط به آن از جمله مفسرها[2]، مترجم‌ها[3] و زبان‌های کنترل رباتیک بوده است.

او همچنین گرایشی ویژه به استانداردسازی زبان‌ها دارد.

هربرت مدرک کارشناسی خود را از دانشگاه ایلی نویز[4] دریافت کرده است. وب‌سایت رسمی او www.Herbschildt.com است که از این طریق می‌توانید با وی در تماس باشید.

درباره‌ی ویراستار متنی

دکتر دنی کاوارد[5]، از سال ۱۹۹۷ در امور مربوط به توسعه‌ی پلت‌فرم جاوا مشارکت داشته است. زمانی که در شرکت Sun فعالیت داشت، یکی از اعضای اصلی گروه توسعه‌ی نسخه‌ی سازمانی جاوا (EE group java) به شمار می‌آمد. او یکی از اعضای کمیته‌ی اجرایی توسعه‌ی جاوا و مشاور در توسعه‌ی تمام نسخه‌های پلت‌فرم جاوا بوده است، از جمله Java SE، Java ME و Java EE. همچنین تیم اصلی Java FX توسط دکتر کاوارد بنیان نهاده شد.

---

1. Herbert schildt
2. Compilers
3. Interpreters
4. Illinois
5. Dr.Danny Coward

مقدمه‌ی نویسنده

جاوا یکی از مهم‌ترین و پرکاربردترین زبان‌های برنامه‌نویسی دنیاست که توانسته این تمایز و برتری را سال‌ها در تصرف خود نگاه دارد و برخلاف سایر زبان‌های برنامه‌نویسی که نفوذ و اعتبارشان با گذشت زمان رو به افول می‌رود، جاوا با گذشت زمان قوی و قوی‌تر شده است. پس از انتشار اولین نسخه از جاوا، این زبان توانست در صدر زبان‌های برنامه‌نویسی مربوط به وب قرار بگیرد. نسخه‌های منتشرشده‌ی بعدی نیز این جایگاه را مستحکم‌تر نمودند. هنوز هم جاوا اولین و بهترین گزینه برای توسعه‌ی برنامه‌های تحت وب به شمار می‌آید. باید گفت که جاوا نقش مهمی در انقلاب تلفن‌های هوشمند داشته است، چرا که کاربرد گسترده‌ای در برنامه‌نویسی اندروید یافته است. به بیان ساده بسیاری از برنامه‌های دنیای مدرن توسط کدهای جاوا اجرا می‌شوند و این موضوع نشان‌دهنده‌ی اهمیت فراوان این زبان است.

کلید اصلی موفقیت جاوا، انعطاف‌پذیری و ظرافت در طراحی آن است. از زمان انتشار نسخه‌ی 1.0 به بعد، جاوا به طور مستمر با تغییرات ایجاد شده در محیط برنامه‌نویسی و نیز تغییر در نحوه‌ی برنامه‌نویسی توسعه‌دهندگان هماهنگ شده است و البته مهم‌تر از انطباق با تغییرات این است که خود جاوا به ایجاد این تغییرات کمک کرده است. توانایی جاوا در انطباق و هماهنگی با روند سریع تغییرات در دنیای رایانه، بخش عمده‌ای از دلیل تداوم موفقیت این زبان به شمار می‌آید.

از زمان انتشار اولین نسخه‌ی این کتاب در سال ۱۹۹۶ تا کنون، ویرایش‌های مختلفی از آن ارائه شده که هر یک بازتابی از تکامل تدریجی جاوا هستند. نسخه‌ای که در حال حاضر پیش روی شما قرار دارد، ویرایش نهم آن است که به جهت تطابق با Java SE8 (JDK 8) به‌روزرسانی شده و از این رو می‌توان گفت میزان قابل توجهی از مضامین و مفاهیم جدید در آن لحاظ شده است. این موضوع بدین معنی است که چنانچه شما با نسخه‌های قبلی آشنایی داشته باشید، می‌توانید به‌راحتی نسخه‌های فعلی را مطالعه کرده و با مضامین جدید آن آشنا شوید.

کتابی برای تمامی برنامه‌نویسان

این کتاب برای تمامی برنامه‌نویسان طراحی شده است، چه افراد تازه‌کار و چه افراد باتجربه و حرفه‌ای. مباحث و مثال‌های کتاب با دقت بسیار طراحی شده و برای افراد مبتدی بسیار مفید و کارآمد است. به علاوه، پوشش گسترده‌ی آن از ویژگی‌های پیشرفته‌ی جاوا و کتابخانه‌های آن می‌تواند نظر کارشناسان مجرب و حرفه‌ای را نیز به خود جلب نماید. می‌توان گفت این کتاب برای هر دو گروه یک منبع پایدار و یک مرجع مفید به شمار می‌آید.

مطالب کتاب

کتاب حاضر راهنمای جامع زبان برنامه‌نویسی جاواست که به توضیح دستورات نحوی، واژه‌های کلیدی و اصول پایه‌ای برنامه‌نویسی در جاوا می‌پردازد. به علاوه بخش قابل توجهی از کتابخانه‌ی رابط برنامه‌ی کاربردی جاوا در آن مورد بحث و بررسی قرار گرفته است. مطالب کتاب به سه بخش اصلی تقسیم‌بندی شده که هر یک به جنبه‌ی خاصی از محیط برنامه‌نویسی جاوا تمرکز دارند.

بخش یک به آموزش پایه‌ای زبان جاوا اختصاص دارد. این بخش با معرفی مبانی زبان شامل مواردی چون انواع داده‌ای، عملگرها، عبارت‌های کنترلی و کلاس‌ها آغاز شده و در ادامه به تشریح مفهوم وراثت، بسته‌ها، واسط‌ها، مدیریت خطاها و برنامه‌نویسی چند نخی می‌پردازد. فصل‌های پایانی نیز توصیف حاشیه‌نویسی‌ها[1]، انواع شمارشی[2]، Auto Boxing و جنریک‌هاست. به علاوه ورودی/خروجی (I/O) و اپلت‌ها نیز در همین فصل معرفی می‌شوند.

در بخش دو، جنبه‌های کلیدی کتابخانه‌ی استاندارد رابطه برنامه‌ی کاربردی جاوا بررسی می‌شود که شامل مواردی چون رشته‌ها، ورودی/خروجی (I/O)، شبکه، ابزارهای استاندارد، پلت‌فرم Collections، اپلت‌ها، کنترل‌های مبتنی بر واسط کاربری گرافیکی و هم‌گام‌سازی (شامل پلت‌فرم جدید Fork/Join) است.

در پایان، در بخش سه نیز نگاهی به سه فناوری نوین جاوا خواهیم داشت که عبارتند از Java Beans، سرولت‌ها و swing.

هربرت شیلد

کدهای منبع تمام مثال‌های کتاب در آدرس زیر موجود و به صورت رایگان قابل دانلود است:
www.kianpub.com

1. Annotation
2. Enumeration

فهرست مختصر

بخش اول: زبان جاوا

فصل ۱- تاریخچه و تکامل جاوا ۲۳
فصل ۲- مروری بر جاوا ۴۱
فصل ۳- انواع داده‌ای، متغیرها و آرایه‌ها ۶۱
فصل ۴- عملگرها ۸۷
فصل ۵- عبارات کنترلی ۱۰۵
فصل ۶- معرفی کلاس‌ها ۱۳۳
فصل ۷- نگاهی دقیق‌تر به متدها و کلاس‌ها ۱۵۱
فصل ۸- وراثت ۱۷۹
فصل ۹- بسته‌ها و واسط‌ها ۲۰۳
فصل ۱۰- مدیریت خطاها (استثنائات) ۲۲۱
فصل ۱۱- برنامه‌نویسی چندنخی ۲۳۷
فصل ۱۲- فهرست‌ها، AutoBoxing و حاشیه‌نویسی (ابرداده‌ها) ۲۵۷
فصل ۱۳- ورودی/خروجی، اپلت‌ها و سایر مباحث مرتبط ۲۷۳
فصل ۱۴- جنریک‌ها ۲۹۱

بخش دوم: کتابخانه‌ی جاوا

فصل ۱۵- اداره کردن رشته (String) ۳۳۵
فصل ۱۶- بررسی java.lang ۳۵۹
فصل ۱۷- java.util قسمت اول: ساختار مجموعه‌ها ۴۱۱
فصل ۱۸- java.util قسمت دوم: سایر کلاس‌های سودمند ۴۷۵
فصل ۱۹- ورودی/خروجی (Input/Output) ۵۲۵
فصل ۲۰- بررسی NIO ۵۶۹
فصل ۲۱- شبکه ۶۰۵
فصل ۲۲- کلاس اپلت (Applet) ۶۲۳
فصل ۲۳- پردازش رویداد (Event) ۶۴۳
فصل ۲۴- معرفی AWT: کار با پنجره‌ها، گرافیک‌ها و متن ۶۶۹
فصل ۲۵- به‌کارگیری کنترل‌های AWT، مدیران لایه‌بندی و منوها ۷۰۳
فصل ۲۶- تصاویر ۷۵۱
فصل ۲۷- ابزار همزمانی ۷۷۷
فصل ۲۸- عبارت‌های منطقی و سایر پکیج‌ها ۸۲۳

بخش سوم: توسعه‌ی نرم‌افزاری با استفاده از جاوا

فصل ۲۹- Java Beans ۸۴۷
فصل ۳۰- معرفی Swing ۸۵۹
فصل ۳۱- بررسی Swing ۸۷۹
فصل ۳۲- سرولت‌ها (Servlet) ۹۰۷

فهرست مطالب

بخش اول: زبان جاوا

فصل ۱- تاریخچه و تکامل جاوا

ریشه‌ی زبان جاوا ۲۳
ظهور برنامه‌نویسی نوین: C ۲۴
گام بعدی: C++ ۲۵
زمینه برای ظهور جاوا آماده است ۲۷
تولد جاوا ۲۷
ارتباط با C# ۲۹
جاوا چگونه توانست اینترنت را تغییر دهد ۳۰
اپلت‌های جاوا ۳۰
امنیت ۳۱
قابلیت حمل ۳۱
جادوی جاوا: Bytecode ۳۱
سرولت‌ها: اجرای جاوا در سمت سرور ۳۲
ویژگی‌های جاوا ۳۳
ساده ۳۳
شیءگرا ۳۴
قدرتمند ۳۴
چندگره‌ای ۳۵
معماری خنثی ۳۵
زبان تفسیری با کارایی بالا ۳۵
توزیع شده ۳۵
پویا ۳۵
سیر تکامل جاوا ۳۶
Java SE 7 ..... ۳۸
فرهنگ نوآوری ۳۹

فصل ۲- مروری بر جاوا

برنامه‌نویسی شیءگرا ۴۱
دو نمونه ۴۱
انتزاع (تجرد) ۴۲
سه قاعده‌ی اصلی برنامه‌نویسی شیءگرا (oop) ۴۳
بسته‌بندی ۴۳
وراثت ۴۴
چندریختی ۴۶
مفاهیم چندریختی، بسته‌بندی و وراثت ۴۶
یک برنامه‌ی ساده ۴۷
درج برنامه ۴۸
کامپایل برنامه ۴۸
نگاهی دقیق‌تر به این برنامه‌ی ساده ۴۹
برنامه‌ی دوم ۵۲
دو دستور کنترلی ۵۳
دستور if ۵۳
حلقه‌ی for ۵۴
استفاده از بلوک‌های کد ۵۶
قواعد نحوی ۵۷
فضاهای خالی ۵۷
شناسه‌ها ۵۷
لیترال‌ها ۵۸
توضیحات ۵۸
جداسازها ۵۸
کلمات کلیدی جاوا ۵۹
کتابخانه‌های کلاس جاوا ۵۹

فصل ۳- انواع داده‌ای، متغیرها و آرایه‌ها

جاوا یک زبان برنامه‌نویسی قدرتمند است ۶۱
انواع داده اصلی یا اولیه ۶۱
اعداد صحیح (integer) ۶۲
byte ۶۳
short ۶۳
int ۶۳
long ۶۴
انواع داده‌ای ممیز شناور ۶۴
float ۶۵
double ۶۵
کاراکترها ۶۵
نوع داده‌ای بولی ۶۷
نگاهی دقیق‌تر به لیترال‌ها ۶۸
لیترال‌های صحیح ۶۸
لیترال‌های ممیز شناور ۶۹
لیترال‌های بولی ۷۰
لیترال‌های کاراکتری ۷۰
لیترال‌های رشته‌ای ۷۱
متغیرها ۷۲
تعریف (اعلان) یک متغیر ۷۲
مقداردهی اولیه به صورت پویا ۷۲
حوزه‌ی کاربرد و طول عمر متغیرها ۷۳
تبدیل نوع و نسبت دادن ۷۵
تبدیل خودکار در جاوا ۷۵
انتساب انواع داده‌ای ناسازگار ۷۶
ارتقای خودکار نوع در عبارات ۷۷
قواعد ارتقای نوع ۷۸
آرایه‌ها ۷۹
آرایه‌های یک‌بعدی ۷۹
آرایه‌های چندبعدی ۸۱
روش دیگری برای اعلان (تعریف) آرایه‌ها ۸۵
چند کلمه در مورد رشته‌ها ۸۵
هشدار به برنامه‌نویسان C/C++ در مورد اشاره‌گرها ۸۶

فصل ۴- عملگرها

عملگرهای حسابی (ریاضی) ۸۷
عملگرهای ریاضی اصلی ۸۸
عملگر modulus (باقیمانده) ۸۹
عملگرهای ریاضی - انتساب ۸۹
عملگرهای افزایش (Increment) و ۹۰
Bitwise - عملگر دودویی (بیتی) ۹۱
عملگرهای بیتی منطقی ۹۳
NOT بیتی ۹۳
AND بیتی ۹۳
OR بیتی ۹۳
XOR بیتی ۹۴
استفاده از عملگرهای منطقی Bitwise ۹۴
انتقال به چپ ۹۵
انتقال به راست ۹۶
انتقال به راست بی‌علامت ۹۷
ترکیب عملگرهای بیتی و انتساب ۹۹
عملگرهای رابطه‌ای ۹۹
عملگرهای منطقی بولی ۱۰۰
عملگرهای منطقی مدار کوتاه ۱۰۲
عملگر انتساب ۱۰۲
عملگر ? ۱۰۳
اولویت عملگرها ۱۰۳
استفاده از پرانتز ۱۰۴

فصل ۵- عبارات کنترلی

عبارات انتخاب در زبان جاوا ۱۰۵
عبارت کنترلی if ۱۰۵
ifهای تودرتو ۱۰۷
عبارت پلکانی if-else-if ۱۰۷
عبارت switch ۱۰۸
عبارات switch تودرتو ۱۱۲
عبارت‌های تکرار ۱۱۳
حلقه while ۱۱۳
حلقه do-while ۱۱۴
حلقه for ۱۱۷
نحوه‌ی تعریف و اعلان متغیرهای تکرار ۱۱۸
استفاده از کاما ۱۱۹
شکل‌های مختلف حلقه‌ی for ۱۲۰
نوع for-each از حلقه‌ی for ۱۲۱
تکرار روی آرایه‌های چندبعدی ۱۲۳
استفاده از حلقه‌ی for پیشرفته (for-each) ۱۲۵
حلقه‌های تودرتو ۱۲۵
عبارت‌های پرشی ۱۲۶
استفاده از break ۱۲۶
استفاده از break برای خروج از حلقه‌ها ۱۲۶
استفاده از break به عنوان صورتی از عبارت goto ۱۲۸
استفاده از continue ۱۳۰
عبارت return ۱۳۱

فصل ۶- معرفی کلاس‌ها

اصول مربوط به کلاس‌ها ۱۳۳
قالب کلی یک کلاس ۱۳۳
یک کلاس ساده ۱۳۴
اعلان اشیا ۱۳۷
نگاهی دقیق‌تر به عملگر new ۱۳۷
انتساب متغیرهای ارجاع شیء ۱۳۸
معرفی متدها ۱۳۹
افزودن یک متد به کلاس Box ۱۳۹
بازگرداندن یک مقدار ۱۴۱
افزودن یک متد پارامتردار ۱۴۲
سازنده‌ها ۱۴۴
سازنده‌های پارامتردار ۱۴۶
کلید واژه‌ی this ۱۴۷
پنهان‌سازی متغیرهای نمونه‌ای ۱۴۷
جمع‌آوری زباله‌های حافظه‌ای ۱۴۸
متد finalize( ) ۱۴۸
کلاس پشته‌ای ۱۴۸

فصل ۷- نگاهی دقیق‌تر به متدها و کلاس‌ها

تحریف (سربارگذاری متدها) ۱۵۱
تحریف سازنده‌ها ۱۵۴
استفاده از اشیا به عنوان پارامتر ۱۵۶
نگاهی دقیق‌تر به فرایند ارسال آرگومان ۱۵۸
بازگرداندن اشیا ۱۵۹
بازگشتی ۱۶۰
معرفی مفهوم کنترل میزان دسترسی ۱۶۲
درک مفهوم static ۱۶۵
معرفی final ۱۶۶
بازبینی آرایه‌ها ۱۶۷
معرفی کلاس‌های تودرتو و کلاس‌های داخلی ۱۶۸
بررسی کلاس String ۱۷۱
استفاده از آرگومان‌های خط فرمان ۱۷۳
varargs: آرگومان‌هایی با طول متغیر ۱۷۴
تحریف متدهای varargs ۱۷۶
varargsها و ابهامات مربوط به آنها ۱۷۷

فصل ۸- وراثت

مفاهیم پایه‌ای وراثت ۱۷۹
دسترسی به اعضا و وراثت ۱۸۱
یک مثال کاربردی‌تر ۱۸۲
متغیر کلاس والد می‌تواند مرجع شیء کلاس ۱۸۴
به‌کارگیری super ۱۸۴
استفاده از عبارت super برای ۱۸۵
کاربرد دوم super ۱۸۸
ایجاد یک ساختار سلسله‌مراتب چندسطحی ۱۸۹
استفاده از سازنده‌ها در وراثت ۱۹۲
تحریف توابع ۱۹۲
ارسال توابع به صورت پویا ۱۹۵
چرا متدها تحریف می‌شوند ۱۹۶
استفاده از تحریف متدها ۱۹۶
استفاده از کلاس‌های انتزاعی ۱۹۸
استفاده از final به همراه وراثت ۲۰۰
استفاده از final به منظور اجتناب از تحریف ۲۰۰

استفاده از final به منظور اجتناب از وراثت ۲۰۱
کلاس Object ۲۰۱

فصل ۹- بسته‌ها و واسط‌ها
بسته‌ها ۲۰۳
تعریف یک بسته ۲۰۴
یافتن بسته‌ها و پارامتر CLASSPATH ۲۰۴
یک مثال ساده از بسته‌ها ۲۰۵
محافظت از دسترسی ۲۰۶
یک مثال از سطح دسترسی ۲۰۶
دریافت (Importing) بسته‌ها ۲۰۹
واسط‌ها ۲۱۱
تعریف یک واسط ۲۱۱
پیاده‌سازی واسط‌ها ۲۱۲
دسترسی به پیاده‌سازی‌ها از طریق ارجاع واسط ۲۱۲
پیاده‌سازی‌های جزئی ۲۱۴
واسط‌های تودرتو ۲۱۴
به‌کارگیری واسط‌ها ۲۱۵
متغیرها در واسط‌ها ۲۱۸
واسط‌ها قابل توسعه هستند ۲۱۹

فصل ۱۰- مدیریت خطاها (استثنائات)
اصول پایه‌ای مدیریت خطا ۲۲۱
انواع استثنا ۲۲۲
خطاهای گرفتارنشده (uncaught) ۲۲۳
استفاده از عبارت try و catch ۲۲۴
نمایش شرح خطا ۲۲۵
عبارت‌های catch چندگانه ۲۲۵
کاربرد عبارت try به صورت تودرتو ۲۲۶
عبارت throw ۲۲۸
عبارت throws ۲۲۹
عبارت finally ۲۳۰
خطاهای داخلی جاوا ۲۳۲
ایجاد زیرکلاس‌های شخصی Exception ۲۳۳
استثنائات زنجیره‌ای ۲۳۴
سه ویژگی جدید JDK7 برای خطاها ۲۳۵
استفاده از خطاها (استثنائات) ۲۳۶

فصل ۱۱- برنامه‌نویسی چندنخی
مدل نخی جاوا (The Java Thread Model) ۲۳۸
اولویت نخ‌ها ۲۳۸
همگام‌سازی ۲۳۸
انتقال پیام ۲۳۹
کلاس Thread و واسط Runnable ۲۳۹
نخ اصلی (Main Thread) ۲۳۹
ایجاد یک نخ ۲۴۰
پیاده‌سازی واسط Runnable ۲۴۰
توسعه‌ی نخ‌ها ۲۴۱
ایجاد نخ‌های چندگانه ۲۴۲
استفاده از isAlive( ) و join( ) ۲۴۴
همگام‌سازی ۲۴۶
استفاده از متدهای همگام‌سازی‌شده ۲۴۶
دستور همگام‌شده ۲۴۷
ارتباطات میان نخ‌ها ۲۴۸
بن‌بست (Deadlock) ۲۵۲
معلق کردن، توقف یا ازسرگیری نخ‌ها ۲۵۳
استفاده از قابلیت چندنخی ۲۵۵

فصل ۱۲- فهرست‌ها، AutoBoxing و حاشیه‌نویسی (ابرداده‌ها)
فهرست‌ها (Enumeration) ۲۵۷
مبانی فهرست‌ها ۲۵۷
متدهای values( ) و valueOf( ) ۲۵۹
فهرست‌های جاوا نوعی کلاس هستند ۲۶۰
فهرست‌ها از نوع Enum ارث‌بری می‌کنند ۲۶۲
مثال دیگری از فهرست‌ها ۲۶۳
پوشش‌های نوع ۲۶۴
کاراکتر ۲۶۴
نوع بولی ۲۶۵
پوشش‌های نوع عددی ۲۶۵
Autoboxing ۲۶۶
ارتباط Autoboxing و متدها ۲۶۶
انواع Autoboxing/Unboxing در عبارت‌ها ... ۲۶۷
Autoboxing/Unboxing مقادیر بولی یا کاراکتری ۲۶۸
Autoboxing/Unboxing به جلوگیری از وقوع ... ۲۶۹
حاشیه‌نویسی (ابر داده‌ها) ۲۶۹
تعیین سیاست‌های حفظ و نگهداری ۲۶۹
دسترسی به حاشیه‌نویسی‌ها در زمان اجرا ... ۲۷۰
یک مثال دیگر از مفهوم انعکاس ۲۷۰
به‌دست آوردن تمام حاشیه‌نویسی‌ها ۲۷۱
واسط AnnotatedElement ۲۷۱
استفاده از مقادیر پیش‌فرض ۲۷۱
حاشیه‌های نشانگر ۲۷۱
حاشیه‌نویسی‌های تک‌عنصری ۲۷۲

فصل ۱۳- ورودی/خروجی، اپلت‌ها و سایر مباحث مرتبط
مبانی ورودی/خروجی ۲۷۳
جریان‌ها ۲۷۴
جریان‌های بایتی و جریان‌های کاراکتری ۲۷۴
کلاس‌های مربوط به جریان‌های بایت ۲۷۴
کلاس‌های مربوط به جریان‌های کاراکتری ۲۷۵
جریان‌های ازپیش‌تعریف‌شده ۲۷۶
نحوه‌ی خواندن ورودی کنسول ۲۷۶
نحوه‌ی خواندن کاراکترها ۲۷۶
نحوه‌ی خواندن رشته‌ها ۲۷۷
نحوه‌ی نوشتن خروجی کنسول ۲۷۸
معرفی کلاس PrintWriter ۲۷۹
خواندن و نوشتن فایل‌ها ۲۸۰
بستن یک فایل به صورت خودکار ۲۸۵
مبانی اپلت‌ها ۲۸۶

مودی‌فایرهای transient و volatile ۲۸۷
استفاده از instanceof ۲۸۷
به‌کارگیری assert ۲۸۹

فصل ۱۴- جنریک‌ها

جنریک چیست؟ ۲۹۲
یک مثال ساده از جنریک‌ها ۲۹۳
جنریک‌ها تنها با انواع ارجاعی کار می‌کنند ۲۹۶
انوع جنریک براساس آرگومان‌های ... ۲۹۶
انواع داده‌ای جنریک چگونه ... ۲۹۷
یک کلاس جنریک با دو پارامتر نوع ۲۹۹
قالب کلی یک کلاس جنریک ۳۰۰
انواع داده‌ای کران‌دار (محدود) ۳۰۱
استفاده از آرگومان‌های Wildcard ۳۰۳
Wildcardهای کران‌دار (محدود) ۳۰۶
تعریف یک متد جنریک ۳۱۰
سازنده‌های جنریک ۳۱۲
واسط‌های جنریک ۳۱۳
انواع داده‌ای خام و کدهای قدیمی ۳۱۵
ساختار سلسله‌مراتبی کلاس‌های جنریک ۳۱۷
استفاده از یک ابرکلاس جنریک ۳۱۸
یک زیرکلاس جنریک ۳۲۰
مقایسه‌های زمان اجرای نوع در یک ساختار ... ۳۲۱
انتساب نوع ۳۲۴
سربارگذاری متدها در کلاس‌های جنریک ۳۲۴
استنتاج نوع در جنریک‌ها ۳۲۵
Erasure ۳۲۷
متدهای bridge ۳۲۷
خطاهای مبهم ۳۲۹
معرفی برخی از محدودیت‌های جنریک‌ها ۳۳۰
از پارامترهای نوع نمی‌توان نمونه ایجاد کرد ۳۳۰
محدودیت‌های مربوط به آرایه ۳۳۱

بخش دوم: کتابخانه‌ی جاوا

فصل ۱۵- اداره کردن رشته (String)

سازنده‌های کلاس String ۳۳۶
طول رشته ۳۳۸
عملیات مخصوص رشته‌ها ۳۳۸
String Literals (لیترال‌های رشته‌ای) ۳۳۸
الحاق رشته ۳۳۸
الحاق رشته با دیگر انواع داده‌ای ۳۳۹
تبدیل رشته و متد toString() ۳۳۹
استخراج کاراکتر ۳۴۰
charAt() ۳۴۱
getChars() ۳۴۱
getBytes() ۳۴۱
toCharArray() ۳۴۲
مقایسه‌ی رشته ۳۴۲
equalsIgnoreCase() و equals() ۳۴۲
regionMatches() ۳۴۳
endsWith() و startsWith() ۳۴۳
equals در برابر == ۳۴۳
compareto() ۳۴۴
جست‌وجوی رشته‌ها ۳۴۵
اصلاح کردن یک رشته ۳۴۶
Substring() ۳۴۶
concat() ۳۴۷
replace() ۳۴۸
trim() ۳۴۸
تبدیل اطلاعات با استفاده از valueOf() ۳۴۹
تغییر دادن نوع کاراکترها ... ۳۴۹
سایر متدهای String ۳۵۰
StringBuffer ۳۵۱
سازنده‌های StringBuffer ۳۵۱
length() و capacity() ۳۵۱
ensureCapacity() ۳۵۲
setLength() ۳۵۲
CharAt() و setCharAt() ۳۵۲
getChars() ۳۵۳
append() ۳۵۳
insert() ۳۵۴
reverse() ۳۵۴
delete() و deleteCharAt() ۳۵۴
replace() ۳۵۵
substring ۳۵۶
سایر متدهای StringBuffer ۳۵۶
StringBuilder ۳۵۷

فصل ۱۶- بررسی java.lang

پوشاننده‌های انواع داده‌ای اولیه ۳۶۰
Number ۳۶۰
Double و Float ۳۶۱
آشنایی با متدهای isInfinite() و isNaN() ۳۶۴
Byte، Short، Integer و Long ۳۶۴
تبدیل اعداد به رشته‌ها و برعکس ۳۷۰
Character (کاراکتر) ۳۷۱
افزودنی‌ها به Character برای پشتیبانی از ... ۳۷۴
Boolean ۳۷۴
Void ۳۷۶
Process ۳۷۶
Runtime ۳۷۷
مدیریت حافظه ۳۷۸
اجرای سایر برنامه‌ها ۳۷۹
ProcessBuilder ۳۸۰
System ۳۸۳
به‌کارگیری currentTimeMillis() برای ... ۳۸۴
استفاده از arraycopy() ۳۸۵
ویژگی‌های محیط ۳۸۶
Object ۳۸۶
به‌کارگیری متد clone() و واسط Cloneable ۳۸۷
Class ۳۸۹

ClassLoader ۳۹۲
Math ۳۹۲
عملیات مثلثاتی ۳۹۲
توابع نمایی ۳۹۳
توابع گرد کردن ۳۹۳
متدهای گوناگون Math ۳۹۵
StrictMath ۳۹۶
Compiler (کامپایلر) ۳۹۶
Thread، ThreadGroup و Runnable ۳۹۶
واسط Runnable ۳۹۶
Thread (نخ) ۳۹۶
ThreadGroup ۳۹۹
InheritableThreadLocal و ThreadLocal ۴۰۲
Package ۴۰۳
RuntimePremission ۴۰۴
Throwable ۴۰۴
SecurityManager ۴۰۴
StackTraceElement ۴۰۴
Enum ۴۰۵
ClassValue ۴۰۶
واسط CharSequence ۴۰۶
واسط Comparable ۴۰۶
واسط Appendable ۴۰۷
واسط Iterable ۴۰۷
واسط Readable ۴۰۷
واسط AutoCloseable ۴۰۷
واسط Thread.UncaughtExceptionHandler ۴۰۸
زیرپکیج‌های java.lang (پکیج‌های جانبی) ۴۰۸
java.lang.annotation ۴۰۸
java.lang.instrument ۴۰۸
java.lang.invoke ۴۰۸
java.lang.management ۴۰۸
java.lang.ref ۴۰۸
java.lang.reflect ۴۰۹

فصل ۱۷- java.util قسمت اول: ساختار مجموعه‌ها

اصول پایه‌ای مربوط به Collections ۴۱۳
JDK5، ساختار Collections را تغییر داده است ۴۱۴
پیدایش جنریک‌ها تغییراتی اساسی در پلت‌فرم ... ۴۱۴
AutoBoxing، استفاده از نوع داده‌ای اولیه ... ۴۱۴
نسخه‌ی For-each از حلقه‌ی for ۴۱۴
واسط‌های Collection ۴۱۵
واسط Collection ۴۱۵
واسط List ۴۱۷
واسط Set ۴۱۸
واسط SortedSet ۴۱۹
واسط NavigableSet ۴۱۹
واسط Queue ۴۲۱
واسط Deque ۴۲۱
کلاس‌های Collection ۴۲۳
کلاس ArrayList ۴۲۴
به دست آوردن یک آرایه از ArrayList ۴۲۵
کلاس LinkedList ۴۲۷
کلاس HashSet ۴۲۸
کلاس LinkedHashSet ۴۲۹
کلاس TreeSet ۴۳۰
کلاس PriorityQueue ۴۳۱
کلاس ArrayDeque ۴۳۱
کلاس EnumSet ۴۳۲
دسترسی به Collection از طریق یک تکرارکننده ۴۳۳
به‌کارگیری تکرارکننده (Iterator) ۴۳۴
For-Each جانشینی برای ... ۴۳۶
ذخیره‌ی کلاس‌های User-Defined در ... ۴۳۷
واسط RandomAccess ۴۳۸
کار با Mapها (نقشه‌ها) ۴۳۸
واسط‌های نقشه ۴۳۸
واسط Map ۴۳۹
واسط SortedMap ۴۴۰
واسط NavigableMap ۴۴۱
واسط Map.Entry ۴۴۲
کلاس‌های Map ۴۴۲
کلاس HashMap ۴۴۳
کلاس TreeMap ۴۴۴
کلاس LinkedHashMap ۴۴۵
کلاس IdentityHashMap ۴۴۶
کلاس EnumMap ۴۴۶
مقایسه‌کننده‌ها (comparators) ۴۴۷
به‌کارگیری یک مقایسه‌کننده ۴۴۷
الگوریتم‌های Collection ۴۵۰
آرایه‌ها ۴۵۴
کلاس‌ها و واسط‌های ارثی ۴۵۸
واسط Enumeration ۴۵۹
Vector (بردار) ۴۵۹
Stack (پشته) ۴۶۲
Dictionary (فرهنگ لغت) ۴۶۴
جدول هش (Hashtable) ۴۶۵
Properties (ویژگی‌ها) ۴۶۸
به‌کارگیری ( )store و ( )load ۴۷۱
حرف آخر درباره‌ی Collectionها ۴۷۳

فصل ۱۸- java.utilقسمت دوم: سایر کلاس‌های سودمند

StringTokenizer ۴۷۵
BitSet ۴۷۷
Date (تاریخ) ۴۷۹
Calendar (تقویم) ۴۸۱
GregorianCalendar (تقویم جورجی) ۴۸۳
TimeZone (منطقه‌ی زمانی) ۴۸۵
SimpleTimeZone ۴۸۶
Locale ۴۸۶
Random ۴۸۸
Observable ۴۸۹
واسط Observer ۴۹۰
مثالی از Observer ۴۹۰

Timer و TimerTask ۴۹۳
Currency (واحد پول) ۴۹۵
Formatter ۴۹۶
تابع‌های سازنده‌ی Formatter ۴۹۶
متدهای Formatter ۴۹۷
اصول قالب‌بندی (Formatting) ۴۹۷
قالب‌بندی رشته‌ها و کاراکترها ۴۹۹
فرمت‌دهی اعداد ۴۹۹
قالب‌بندی زمان و تاریخ ۵۰۰
مشخص‌کننده‌های n% و % % ۵۰۲
مشخص کردن حداقل پنهای فیلد ۵۰۲
مشخص کردن ۵۰۴
به‌کارگیری نشانه‌های فرمت (format flags) ۵۰۴
ترازبندی خروجی ۵۰۵
نشانه‌های space (فاصله)، + ، 0 و) ۵۰۵
نشانه‌ی کاما (ویرگول) ۵۰۶
نشانه # ۵۰۷
حروف بزرگ ۵۰۷
به‌کارگیری اندیس آرگومان ۵۰۷
بستن یک Formatter ۵۰۸
ارتباط ( )Java Printf ۵۰۹
Scanner ۵۰۹
تابع‌های سازنده Scanner ۵۰۹
اصول اسکن کردن ۵۱۰
مثال‌هایی از Scanner ۵۱۳
تنظیم جداکننده‌ها (Delimiters) ۵۱۷
سایر ویژگی‌های Scanner ۵۱۸
کلاس‌های ResourceBundle، ۵۱۹
واسط‌ها و کلاس‌های گوناگون کاربردی ۵۲۳
زیرپکیج‌های java.util ۵۲۳
java.util.concurrent ۵۲۴
java.util.jar ۵۲۴
java.util.logging ۵۲۴
java.util.prefs ۵۲۴
java.util.regen ۵۲۴
java.util.spi ۵۲۴
java.util.zip ۵۲۴

فصل ۱۹- ورودی/خروجی (Input/Output)

بررسی java.io ۵۲۵
کلاس‌های واسط I/O ۵۲۶
File ۵۲۶
Directorها (راهنماها) ۵۲۹
به‌کارگیری FilenameFilter ۵۳۰
جایگزین ( )listFiles ۵۳۰
ساخت Directory ۵۳۱
واسط‌های AutoCloseable، Closeable و ۵۳۱
استثناهای I/O ۵۳۲
دو راه برای بستن یک استریم ۵۳۲
کلاس‌های Stream ۵۳۳
استریم‌های بایتی ۵۳۳
InputStream ۵۳۳
OutputStream ۵۳۴
FileInputStream ۵۳۵
FileOutputStream ۵۳۶
ByteArrayInputStream ۵۳۹
ByteArrayOutputStream ۵۴۰
استریم‌های بایت فیلتر شده ۵۴۱
استریم‌های بایت بافر شده ۵۴۱
BufferedInputStream ۵۴۱
BufferedOutputStream ۵۴۳
PushbakInputStream ۵۴۳
SequenceInputStream ۵۴۵
PrintStream ۵۴۶
DataOutputStream و DataInputStream ۵۴۸
RandomAccessFile ۵۵۰
استریم‌های کاراکتری ۵۵۱
Reader ۵۵۱
Writer ۵۵۱
FileReader ۵۵۳
FileWriter ۵۵۳
CharArrayReader ۵۵۴
CharArrayWriter ۵۵۵
BufferedReader ۵۵۶
BufferedWriter ۵۵۸
PushbackReader ۵۵۸
PrintWriter ۵۵۹
کلاس کنسول (console) ۵۶۰
Serialization (سریال‌سازی) ۵۶۲
Serializable ۵۶۲
Externalizable ۵۶۲
ObjectOutput ۵۶۳
ObjectOutputStream ۵۶۳
ObjectInput ۵۶۴
ObjectInputStream ۵۶۵
مثالی از سریال‌سازی ۵۶۶
مزایای استریم ۵۶۷

فصل ۲۰- بررسی NIO

کلاس‌های NIO ۵۶۹
اصول NIO ۵۷۰
بافرها ۵۷۰
کانال‌ها ۵۷۱
مجموعه‌ی کاراکترها و انتخاب کدها ۵۷۳
تقویت‌های NIO توسط JDK7 ..... ۵۷۳
واسط Path ۵۷۳
کلاس Files ۵۷۴
کلاس Paths ۵۷۷
واسط‌های File Atrribute ۵۷۷
کلاس‌های FileSystem، FileSystems و ۵۷۹
به‌کارگیری سیستم NIO ۵۷۹
استفاده از NIO برای I/O کانال محور ۵۸۰

خواندن یک فایل از طریق یک کانال ۵۸۰
نوشتن در فایل از طریق کانال ۵۸۴
کپی کردن فایل با استفاده از NIO ۵۸۸
به‌کارگیری NIO برای I/O ۵۸۹
به‌کارگیری NIO برای عملیات Path و ۵۹۱
به دست آوردن اطلاعات مربوط به ۵۹۱
محتویات یک راهنما را فهرست کنید ۵۹۳
به کارگیری walkFileTree( ) برای ۵۹۶
مثال‌های کانال محور قبل از JDK7 ..... ۵۹۸
خواندن یک فایل قبل از JDK7 ..... ۵۹۹
نوشتن در فایل، قبل از JDK7 ..... ۶۰۲

فصل ۲۱- شبکه

اصول شبکه ۶۰۵
کلاس‌ها و واسط‌های شبکه ۶۰۶
InetAddress ۶۰۷
متدهای کارخانه‌ای ۶۰۷
متدهای نمونه (موردی) ۶۰۸
Inet4Address و Inet6Address ۶۰۸
سوکت‌های مشتری ICP/IP ۶۰۹
URL ۶۱۲
URLConnection ۶۱۳
HttpURLConnection ۶۱۶
کلاس URI ۶۱۷
کوکی‌ها ۶۱۸
سوکت‌های سرور TCP/IP ۶۱۸
نمودارهای داده‌ای (Datagrams) ۶۱۸
DatagramSocket ۶۱۹
DatagramPacket ۶۲۰
مثالی از نمودار داده ۶۲۰

فصل ۲۲- کلاس اپلت (Applet)

انواع اپلت‌ها ۶۲۳
اصول اپلت ۶۲۴
کلاس Applet ۶۲۵
معماری اپلت ۶۲۶
ساختار اپلت ۶۲۷
آغاز و پایان اپلت ۶۲۸
بازنویسی update( ) ۶۲۹
روش‌های نمایش ساده‌ی اپلت ۶۲۹
درخواست طراحی دوباره (repainting) ۶۳۱
یک اپلت بنر ساده ۶۳۲
به‌کارگیری پنجره‌ی وضعیت (status) ۶۳۳
برچسب HTML APPLET ۶۳۴
وارد کردن پارامترها به اپلت‌ها ۶۳۵
ارتقای اپلت بنر ۶۳۷
getDocumentBase( ) و getCodeBase( ) ۶۳۸
ShowDocument( ) و AppletContext ۶۳۹
واسط AudioClip ۶۴۱
واسط AppletStub ۶۴۱
خروجی گرفتن برای کنسول ۶۴۱

فصل ۲۳- پردازش رویداد (Event)

دو مکانیسم پردازش رویداد ۶۴۳
مدل رویداد نیابتی (Delegation Event) ۶۴۴
رویدادها ۶۴۴
منابع رویداد ۶۴۴
شنونده‌های رویداد ۶۴۵
کلاس‌های رویداد ۶۴۵
کلاس ActionEvent ۶۴۶
کلاس AdjustmentEvent ۶۴۷
کلاس ComponentEvent ۶۴۸
کلاس ContainerEvent ۶۴۸
کلاس FocusEvent ۶۴۸
کلاس InputEvent ۶۴۹
کلاس ItemEvent ۶۵۰
کلاس KeyEvent ۶۵۰
کلاس MouseEvent ۶۵۱
کلاس MouseWheelEvent ۶۵۲
کلاس TextEvent ۶۵۳
کلاس WindowEvent ۶۵۳
منابع رویدادها ۶۵۴
واسط‌های شنونده‌ی رویداد ۶۵۵
واسط ActionalListener ۶۵۶
واسط AdjusmentListener ۶۵۶
واسط ComponentListener ۶۵۶
واسط ContainerListener ۶۵۶
واسط FocusListener ۶۵۶
واسط ItemListener ۶۵۶
واسط KeyListener ۶۵۶
واسط MouseListener ۶۵۷
واسط MouseMotionListener ۶۵۷
واسط MouseWheelListener ۶۵۷
واسط TextListener ۶۵۷
واسط windowFocusListener ۶۵۷
واسط WindowListener ۶۵۷
به‌کارگیری مدل رویداد نیابتی ۶۵۸
پردازش رویدادهای ماوس ۶۵۸
پردازش رویدادهای صفحه‌کلید ۶۶۰
کلاس‌های سازگارکننده (Adapter) ۶۶۳
کلاس‌های درونی (Inner) ۶۶۵
کلاس‌های درونی بدون نام (Anonymous) ۶۶۷

فصل ۲۴- معرفی AWT: کار با پنجره‌ها، گرافیک‌ها و متن

کلاس‌های AWT ۶۷۰
اصول پنجره ۶۷۲
Component ۶۷۲
Container ۶۷۲
Panel ۶۷۲
Window ۶۷۳
Frame ۶۷۳
Canvas ۶۷۳
کار کردن با پنجره‌های فریم ۶۷۳

تنظیم ابعاد پنجره ۶۷۳
مخفی کردن و نشان دادن یک پنجره ۶۷۴
تنظیم عنوان پنجره ۶۷۴
بستن یک پنجره‌ی فریم ۶۷۴
ساخت یک پنجره‌ی فریم در یک اپلت ۶۷۴
پردازش رویدادها در یک پنجره‌ی فریم ۶۷۶
ساخت یک برنامه‌ی پنجره‌ای ۶۸۰
نمایش اطلاعات در یک پنجره ۶۸۲
معرفی گرافیک ۶۸۲
طراحی خط ۶۸۳
طراحی مستطیل ۶۸۳
طراحی بیضی و دایره ۶۸۳
ترسیم کمان (قوس) ۶۸۴
طراحی چندضلعی ۶۸۴
اندازه‌گیری گرافیک ۶۸۴
کار با رنگ‌ها ۶۸۵
متدهای Color ۶۸۶
به‌کارگیری سایه (Hue)، اشباع (Saturation) ... ۶۸۶
getBlue( )، getGreen( ) و getRed( ) ۶۸۶
getRGB( ) ۶۸۷
تنظیم رنگ فعلی گرافیک ۶۸۷
اپلت نمونه رنگ (color) ۶۸۷
تنظیم حالت نقاشی (Paint Mode) ۶۸۸
کار با قلم‌ها (Font) ۶۸۹
تعیین قلم‌های موجود ۶۹۰
ساخت و انتخاب یک قلم ۶۹۱
به دست آوردن اطلاعات قلم ۶۹۳
مدیریت خروجی متن با ... ۶۹۴
نمایش چند سطر از متن ۶۹۵
در مرکز قرار دادن متن ۶۹۷
ترازبندی متن چندسطری ۶۹۸

فصل ۲۵- به‌کارگیری کنترل‌های AWT، مدیران لایه‌بندی و منوها

اصول کنترل ۷۰۳
افزودن و حذف کنترل‌ها ۷۰۴
پاسخ‌گویی به کنترل‌ها ۷۰۴
HeadlessException ۷۰۴
برچسب‌ها ۷۰۴
به‌کارگیری دکمه‌ها ۷۰۶
پردازش دکمه‌ها ۷۰۶
به‌کار بردن چک‌باکس‌ها ۷۰۸
پردازش چک‌باکس‌ها ۷۰۹
CheckboxGroup ۷۱۱
کنترل‌های Choice ۷۱۲
پردازش فهرست‌های Choice ۷۱۳
به‌کارگیری فهرست‌ها ۷۱۴
پردازش فهرست‌ها ۷۱۵
مدیریت نوارهای حرکت (Scroll Bars) ۷۱۷
پردازش نوارهای حرکت ۷۱۸
به‌کارگیری TextField ۷۲۰
پردازش TextField ۷۲۱
به‌کارگیری TextArea ۷۲۲
آشنایی با مدیران لی‌اوت ۷۲۴
FlowLayout ۷۲۵
BorderLayout ۷۲۶
به‌کارگیری ضمیمه‌ها (Inset) ۷۲۸
GridLayout ۷۲۹
CardLayout ۷۳۰
GridBagLayout ۷۳۳
نوارهای منو و منوها ۷۳۸
جعبه‌های گفت‌وگو (DialogBox) ۷۴۳
FileDialog ۷۴۷
چند کلمه در مورد بازنویسی paint( ) ۷۴۹

فصل ۲۶- تصاویر

فرمت‌های فایل ۷۵۱
اصول تصویر: ساخت، بارگذاری و نمایش ۷۵۲
ساخت یک شیء Image ۷۵۲
بارگذاری یک تصویر ۷۵۲
نمایش تصویر ۷۵۳
Image Observer (مشاهده‌گر تصویر) ۷۵۴
بافر دوبل ۷۵۵
MediaTracker ۷۵۷
ImageProducer ۷۶۰
MemoryImageSource ۷۶۰
ImageConsumer ۷۶۲
PixelGrabber ۷۶۲
ImageFilter ۷۶۴
CropImageFilter ۷۶۴
RGBImageFilter ۷۶۶
ImageFilterDemo.java ۷۶۶
Plug.InFilter.java ۷۶۸
LoadedImage.java ۷۶۹
Grayscale.java ۷۶۹
Invert.java ۷۷۰
Contrast.java ۷۷۱
Convolver.java ۷۷۱
Blur.java ۷۷۳
Sharpen.java ۷۷۵
سایر کلاس‌های تصویرسازی ۷۷۶

فصل ۲۷- ابزار همزمانی

پکیج‌های همزمانی API ۷۷۸
java.util.concurrent ۷۷۸
java.util.concurrent.atomic ۷۷۹
java.util.concurrent.locks ۷۷۹
به‌کارگیری اشیای همگام‌سازی ۷۷۹
Semaphore ۷۷۹
CountDownLatch ۷۸۵
CyclicBarrier ۷۸۶
Exchanger ۷۸۹

phaser ۷۹۰
به‌کارگیری اجراکننده ۷۹۶
مثالی از اجراکننده ساده ۷۹۷
به‌کارگیری Callable و Future ۷۹۹
شمارش TimeUnit ۸۰۲
Collectionهای همزمان ۸۰۳
Locks (قفل‌ها) ۸۰۳
عملیات اتمی (Atomic) ۸۰۶
برنامه‌نویسی موازی از ... ۸۰۷
کلاس‌های اصلی Fork/Join ۸۰۸
ForkJoinTask<V> ۸۰۸
RecursiveAction ۸۰۹
RecursiveTask<V> ۸۰۹
ForkJoinPool ۸۱۰
استراتژی Divide – and – Conquer ۸۱۱
یک مثال ساده از Fork/Join ۸۱۱
درک تأثیر سطح موازی‌سازی ۸۱۴
مثالی از به کارگیری RecursiveTask<V> ۸۱۶
اجرای یک وظیفه به صورت غیرهمزمان ۸۱۹
لغو کردن یک وظیفه ۸۱۹
تعیین وضعیت کامل بودن یک وظیفه ۸۱۹
آغاز دوباره‌ی یک وظیفه (restart) ۸۲۰
نکات مهم ۸۲۰
نمونه‌ای سایر خصوصیات ForkJoinTask ۸۲۰
نمونه‌ای از سایر خصوصیات ForkJoinTask ۸۲۱
نکاتی در مورد Fork/Join ۸۲۱
ابزار همزمانی در مقابل روش سنتی جاوا ۸۲۲

فصل ۲۸- عبارت‌های منطقی و سایر پکیج‌ها

پکیج‌های اصلی Java API ۸۲۳
پردازش عبارت منطقی ۸۲۵
Pattern ۸۲۵
Matcher ۸۲۶
قواعد عبارت منطقی ۸۲۶
ارایه‌ای از الگوی همانندسازی ۸۲۷
به‌کارگیری کاراکترهای عمومی و معیارها ... ۸۲۹
کار با کلاس‌های کاراکترها ۸۳۰
به‌کارگیری repalceAll( ) ۸۳۱
به‌کارگیری spilt( ) ۸۳۱
دو امکان الگو – تطبیق ۸۳۲
بررسی عبارت‌های منطقی ۸۳۲
Reflection (بازتابش) ۸۳۳
فراخوان روش راه دور (RMI) ۸۳۶
برنامه‌ی کاربردی کارخواه ... ۸۳۶
مرحله‌ی اول: کد منبع را وارد و کامپایل کنید ۸۳۶
مرحله‌ی دوم: تولید دستی یک stub ... ۸۳۸
مرحله‌ی سوم: نصب فایل‌ها روی ماشین‌های ... ۸۳۸
مرحله‌ی چهارم: آغاز رجسیتری RMI روی ۸۳۹
مرحله‌ی پنجم: سرور را آغاز کنید ۸۳۹
مرحله‌ی ششم: کلاینت را آغاز کنید ۸۳۹
قالب‌بندی تاریخ و ساعت با java.text ۸۳۹
کلاس DateFormat ۸۳۹
کلاس simpleDateFormat ۸۴۱

بخش سوم: توسعه‌ی نرم‌افزاری با استفاده از جاوا

فصل ۲۹- Java Beans

Java Beans چیست؟ ۸۴۷
مزایای Java Beans ۸۴۸
خودکاوی (introspection) ۸۴۸
الگوهای طراحی ویژگی‌ها ۸۴۸
ویژگی‌های ساده ۸۴۸
ویژگی‌های اندیس‌گذاری شده ۸۴۹
الگوهای طراحی رویدادها ۸۴۹
متدهای و الگوهای طراحی ۸۵۰
به‌کارگیری واسط BeanInfo ۸۵۰
ویژگی‌های Bound (محدود) ... ۸۵۰
Persistence (استمرار) ۸۵۱
Customizerها ۸۵۱
Java Beans API ۸۵۲
Introspector ۸۵۳
PropertyDescriptor ۸۵۳
EventSetDescriptor ۸۵۴
MethodDescriptor ۸۵۴
مثالی از Bean ۸۵۴

فصل ۳۰- معرفی Swing

ظهور Swing ۸۵۹
Swing براساس AWT ساخته شد ۸۶۰
دو ویژگی کلیدی Swing ۸۶۰
مولفه‌های Swing، سبک هستند ۸۶۰
Swing از pluggable look and feel ... ۸۶۰
ارتباط MVC ۸۶۱
مولفه‌ها و ظروف ۸۶۲
مولفه‌ها (Components) ۸۶۲
ظروف (Containers) ۸۶۳
ردیف‌هایی از ظرف سطح بالا ۸۶۳
پکیج‌های Swing ۸۶۳
یک برنامه‌ی ساده‌ی Swing ۸۶۴
پردازش رویداد ۸۶۸
ساخت اپلت Swing ۸۷۱
طراحی (Painting) در Swing ۸۷۳
اصول طراحی ۸۷۳
محاسبه‌ی ناحیه‌ی قابل طراحی ۸۷۴
مثالی از طراحی (paint) ۸۷۵

فصل ۳۱- بررسی Swing

JLable و ImageIcon ۸۷۹
JTextField ۸۸۱
دکمه‌های Swing ۸۸۳
JButton ۸۸۳
JToggleButton ۸۸۵
چک‌باکس‌ها ۸۸۸

دکمه‌های رادیویی ۸۸۹
JTabbedPane ۸۹۱
JScrollPane ۸۹۴
JList ۸۹۶
JComboBox ۸۹۹
Tree (درخت‌ها) ۹۰۱
JTable ۹۰۴

فصل ۳۲- سرولت‌ها (Servlet)

پیشینه ۹۰۷
چرخه‌ی حیات سرولت ۹۰۸
امکانات توسعه‌ی سرولت ۹۰۸
به‌کارگیری Tomcat ۹۰۹
یک سرولت ساده ۹۱۰
Tomcat را آغاز کنید ۹۱۱
یک مرورگر وب را آغاز کرده و ... ۹۱۱
Servlet API ۹۱۲
پکیج javax.servlet ۹۱۲
رابط Servlet ۹۱۲
واسط ServletConfig ۹۱۳
واسط ServletContext ۹۱۳
واسط ServletRequest ۹۱۳
واسط ServletResponse ۹۱۴
کلاس GenericServlet ۹۱۵
کلاس ServletInputStream ۹۱۵
کلاس ServletOutputStream ۹۱۵
کلاس‌های ServletException ۹۱۶
خواندن پارامترهای سرولت ۹۱۶
پکیج javax.servlet.http ۹۱۷
واسط HttpServletRequest ۹۱۸
واسط HttpServletResponse ۹۱۹
واسط HttpSession ۹۱۹
واسط HttpSessionBindingListener ۹۱۹
کلاس Cookie ۹۲۰
کلاس HttpServlet ۹۲۱
پردازش درخواست‌ها و پاسخ‌های HTTP ۹۲۲
پردازش درخواست‌های HTTP GET ۹۲۲
مدیریت درخواست‌های HTTP POST ۹۲۴
به‌کارگیری کوکی‌ها ۹۲۵
پیگیری Session ۹۲۷

بخش اول

زبان جاوا

فصل ۱

تاریخچه و تکامل جاوا

برای درک بهتر جاوا باید دلایل ایجاد و لزوم شکل‌گیری آن را مورد مطالعه قرار داد. جاوا نیز همانند سایر زبان‌های برنامه‌نویسی موفقی که پیش از آن ایجاد شده‌اند، ترکیبی از مولفه‌های قوی و مضامین نوینی است که جهت نیل به هدف و رسالت منحصر به فرد خود به آنها نیاز دارد.

این فصل بیان می‌دارد که جاوا چگونه و به چه دلیل پدید آمده است، چه ویژگی‌هایی آن را ممتاز کرده و چگونه توانسته است در این مدت قابلیت‌های خود را بهبود بخشد. در حالی‌که سایر فصل‌های کتاب بیشتر به جنبه‌های کاربردی آن نظیر دستورهای نحوی[۱]، کتابخانه‌های کلیدی و برنامه‌های کاربردی می‌پردازند.

هرچند این زبان در حال حاضر بخشی جدایی‌ناپذیر از محیط آنلاین اینترنت شده و ارتباطی تنگاتنگ با آن دارد، اما نباید این نکته را فراموش کرد که جاوا در درجه‌ی نخست یک زبان برنامه‌نویسی است.

پیدایش و توسعه‌ی زبان‌های برنامه‌نویسی رایانه‌ای دو دلیل عمده داشته است:

- انطباق با کاربردها و محیط‌های در حال تغییر؛
- انجام اصلاحات و تقویت مهارت برنامه‌نویسی.

در ادامه درخواهید یافت که هر دو عامل ذکرشده، به‌طور تقریبی به یک اندازه در توسعه‌ی جاوا سهم داشته‌اند.

ریشه‌ی زبان جاوا

جاوا ارتباط نزدیکی با زبان C++ دارد که خود از نوادگان مستقیم زبان C به شمار می‌آید. به این ترتیب می‌توان گفت که جاوا بسیاری از خصوصیات و ویژگی‌های اصلی خود را از این دو زبان به ارث برده است. به عنوان مثال، دستورات نحوی جاوا برگرفته از زبان C و بسیاری از ویژگی‌های شیءگرایی آن نیز تاثیر گرفته از مفاهیم شیءگرایی در زبان C++ است. در حقیقت جاوا، بسیاری از خصوصیات خود را از این دو زبان به ارث برده است. به علاوه، پیدایش زبان جاوا اساسا مربوط به فرایندهای پالایش و تطابقی است که در طی چند دهه‌ی اخیر در زبان‌های برنامه‌نویسی رایانه‌ای رخ داده است.

---

۱. Syntax به معنای ساختار دستوری یک زبان برنامه‌نویسی است که چون در متن کتاب، همراه کلمه‌ی Statement (دستور) به کار گرفته شده، از کلمه نحوی برای ترجمه آن استفاده شده است.

به همین دلیل، در این فصل نگاهی به مجموعه‌ای از رویدادها و نیازهایی خواهیم داشت که منجر به پیدایش زبان جاوا شده‌اند. در ادامه خواهید دید که هرگونه نوآوری در طراحی این زبان به دلیل نیاز به حل مسائل بنیادی و خاصی بوده که زبان‌های قبلی قادر به حل آن مسائل و ارایه‌ی راهکارهای مناسب برای آنها نبوده‌اند. البته زبان جاوا نیز عاری از عیب نیست.

ظهور برنامه‌نویسی نوین: C

ظهور زبان C، دنیای رایانه را به لرزه درآورد. به هیچ وجه نباید تاثیرات این زبان را دست کم گرفت و ناچیز انگاشت، چرا که زبان C روش‌هایی را که تا آن زمان برای برنامه‌نویسی و تحلیل استفاده می‌شد به طور کلی تغییر داد.

پیدایش این زبان نتیجه‌ی مستقیم نیاز به یک زبان ساختاریافته، کارآمد و سطح بالا بود که بتواند به هنگام ایجاد برنامه‌های سیستمی، کدهای نزدیک به زبان کاربر را جایگزین کدهای اسمبلی نماید.

همان‌طور که می‌دانید، به هنگام طراحی یک زبان برنامه‌نویسی باید یک سری مزایا و معایب را مد نظر گرفته و بررسی نمود؛ مواردی از قبیل:

- استفاده‌ی آسان در مقابل قدرتمندی زبان؛
- امنیت در مقابل کارایی؛
- یکپارچگی در برابر توسعه‌پذیری.

پیش از پیدایش زبان C برنامه‌نویسان مجبور بودند برای نوشتن برنامه‌های خود، زبان‌های موجود و قابلیت‌های آنها را بررسی نموده و از بین آنها زبانی را انتخاب کنند که نسبت به سایر زبان‌ها ویژگی‌های برجسته‌تری داشت. به عنوان مثال اگرچه زبان فرترن زبان نسبتا مناسبی برای نوشتن برنامه‌های علمی است، اما برای برنامه‌های سیستمی گزینه‌ی مناسبی نیست و یا اگرچه فراگیری زبان بیسیک بسیار آسان است، اما از لحاظ قدرت و کارایی، زبان ضعیفی به شمار می‌آید. همچنین عدم ساخت‌یافتگی این زبان، کارایی آن برای برنامه‌های بزرگ را زیر سؤال برده است.

زبان اسمبلی برای ایجاد برنامه‌هایی با کارایی بالا بسیار مناسب است، اما یادگیری آن دشوار بوده و استفاده از آن نیز چندان ساده به نظر نمی‌آید. به علاوه، اشکال‌زدایی برنامه‌هایی که به زبان اسمبلی نوشته می‌شوند هم بسیار دشوار است.

مسئله‌ی بغرنج دیگر در مورد زبان‌های برنامه‌نویسی اولیه مثل بیسیک، کوبول و فرترن آن است که این زبان‌ها براساس مفاهیم ساخت یافتگی طراحی نشده‌اند. در عوض طراحی آنها مبتنی بر عبارت کنترلی GOTO است که یکی از ابزارهای اولیه برای کنترل برنامه‌ها به شمار می‌آید. از این رو اغلب به برنامه‌هایی که با این دسته از زبان‌های برنامه‌نویسی نوشته می‌شوند، "کدهای اسپاگتی[1]" گفته می‌شود؛ یک توده‌ی درهم از عبارت‌های شرطی و پرشی که در نهایت فهم منطق برنامه را برای کاربران و دیگر برنامه‌نویسان غیرممکن می‌سازد. اگرچه برخی از زبان‌های برنامه‌نویسی نظیر پاسکال ساخت‌یافته هستند، اما به گونه‌ای طراحی نشده‌اند که کارایی بالایی داشته باشند و اساسا ویژگی‌های لازم برای نوشتن طیف گسترده‌ای از برنامه‌ها را ندارند (به ویژه، با توجه به نسخه‌های خاصی از زبان پاسکال که در حال حاضر موجود است، به نظر نمی‌رسد استفاده از این زبان برای برنامه‌های سیستمی مناسب باشد).

بنابراین پیش از پیدایش زبان C، هیچ زبانی نتوانسته بود ویژگی‌های متنوعی که از یک زبان برنامه‌نویسی انتظار می‌رود را به طور کامل پوشش دهد و نیاز به وجود چنین زبانی کاملا احساس می‌شد.

در اوایل دهه‌ی ۱۹۷۰ بود که انقلاب رایانه‌ای به وقوع پیوست. در این شرایط نیاز به نرم‌افزارهای جدید به شدت افزایش یافت، به گونه‌ای که برنامه‌نویسان قادر نبودند تقاضای بازار را در تولید نرم‌افزارهای جدید برآورده سازند. از این رو در محیط‌های آکادمیک تلاش‌های گسترده‌ای به منظور ایجاد یک زبان برنامه‌نویسی قدرتمند انجام گرفت.

---

۱. spaghetti code: کد منبع دارای ساختارهای کنترلی پیچیده.

اما در همین زمان مسئله‌ی دیگری نیز وجود داشت که به مراتب مهم‌تر از انقلاب نرم‌افزاری به شمار می‌آمد. سخت‌افزارهای رایانه‌ای در حجم وسیع تولید شده و رواج یافته و به راحتی در دسترس عموم قرار گرفته بودند. به گونه‌ای که تجهیزات رایانه‌ای دیگر در انحصار افراد خاص نبوده و به قول معروف پشت درهای بسته نگهداری نمی‌شد. در همین زمان بود که برنامه‌نویسان توانستند برای اولین بار به طور مجازی و به صورت نامحدود به رایانه‌ها دسترسی داشته باشند. این امر آزادی عمل بیشتری را برای برنامه‌نویسان جهت تست و آزمایش برنامه‌ها فراهم کرده و به آنها این اجازه را داد که بتوانند خودشان ابزارهای مناسب را تهیه نمایند و بدین ترتیب در آستانه‌ی پیدایش زبان C، شرایط برای جهش رو به جلوی زبان‌های برنامه‌نویسی به خوبی مهیا گردید.

زبان C برای اولین بار توسط دنیس ریچی[1] بر روی ماشین DEC PDP-11 و سیستم‌عامل یونیکس پیاده‌سازی گردید، اما باید گفت که این زبان نتیجه‌ی فرایند توسعه‌ی زبان قدیمی‌تری به نام BCPL است که توسط مارتین ریچاردز[2] طراحی و توسعه داده شده است.

BCPL با تغییر در زبان B که توسط کم تامسون[3] ابداع شده بود، توانست در دهه‌ی ۱۹۷۰، زبان C را به دنیا معرفی کند.

در دسامبر سال ۱۹۸۹، زبان C به طور رسمی از سوی مؤسسه‌ی استاندارد ملی آمریکا (ANSI) به عنوان یک زبان استاندارد به تصویب رسید.

به عقیده بسیاری، پیدایش زبان C را می‌توان آغازگر عصری نوین در زبان‌های برنامه‌نویسی دانست. این زبان به خوبی توانست ویژگی‌های ناسازگار یا حتی متناقض زبان‌های برنامه‌نویسی دیگر را به صورت کاملا یکپارچه و کارآمد گردآوری نماید. در نتیجه، زبانی قدرتمند، کارا و ساخت‌یافته پدید آمد که یادگیری آن نیز بسیار ساده بود. در کنار همه‌ی این موارد، پیدایش زبان C مزیت دیگری نیز به همراه داشت که شاید چندان ملموس نباشد: زبان C زبانی اجرایی برای برنامه‌نویسان بود!

پیش از پیدایش زبان C، زبان‌های برنامه‌نویسی عموما برای استفاده از محیط‌های آکادمیک یا کمیته‌های اداری طراحی شده و توسعه می‌یافتند. اما C، با همه‌ی زبان‌های پیشین تفاوت داشت. این زبان به طور کامل توسط برنامه‌نویسان حرفه‌ای طراحی، پیاده‌سازی و توسعه داده شده است و در حقیقت بازتابی از مهارت برنامه‌نویسی آنها می‌باشد. ویژگی‌های این زبان به طور کامل توسط افراد مجرب مورد بررسی و آزمایش قرار گرفت. به علاوه پس از معرفی زبان C، از بازخورد نظرات افرادی که از آن استفاده کرده بودند نیز در بازبینی ویژگی‌های آن بهره گرفته شد. نتیجه‌ی تمام این تلاش‌ها زبانی بود که اغلب برنامه‌نویسان تمایل داشتند از آن استفاده کنند. به این ترتیب زبان C توانست به سرعت در میان برنامه‌نویسان محبوب شده و افراد زیادی را طرفدار سرسخت خود سازد.

در مجموع، زبان C زبانی است که برای برنامه‌نویسان طراحی و توسعه یافته است. در ادامه خواهید دید که جاوا نیز به همین هدف توسعه پیدا کرده است.

گام بعدی: C++

در سال‌های پایانی دهه‌ی ۱۹۷۰ و اوایل دهه‌ی ۱۹۸۰، زبان C محبوب‌ترین و رایج‌ترین زبان برنامه‌نویسی به شمار می‌آمد که البته محبوبیت خود را تا زمان حال نیز حفظ کرده است. ممکن است این سؤال به ذهنتان برسد که اگر C زبانی مفید، موفق و قدرتمند است، چه دلیلی برای طراحی و توسعه‌ی زبان‌های دیگر وجود دارد؟ پاسخ این سؤال تنوع و پیچیدگی نیاز کاربران است. در طول تاریخ برنامه‌نویسی افزایش پیچیدگی برنامه‌ها موجب شده تا همواره به دنبال روش‌های کامل‌تری جهت مدیریت پیچیدگی در برنامه‌ها باشیم و زبان C++، پاسخی به این نیاز بوده است.

1. Dennis Ritchie
2. Martin Richards
3. Kem Tompson

برای آن که دریابید چرا مدیریت پیچیدگی برنامه‌ها یکی از دلایل اصلی پیدایش زبان C++ بوده است، به بخش زیر توجه کنید.

از همان آغاز اختراع رایانه، زبان‌های برنامه‌نویسی به طور چشمگیر دستخوش تغییرات فراوانی قرار گرفته‌اند. به عنوان مثال، زمانی که رایانه‌ها برای اولین بار مورد استفاده قرار گرفتند، کدنویسی توسط دستورالعمل‌های دودویی انجام می‌گرفت. تا زمانی که تعداد دستورالعمل‌های موجود در برنامه‌ها از چند صد دستور تجاوز نمی‌کرد، این روش جوابگوی نیازهای برنامه‌نویسان بود. اما رفته رفته با بزرگ‌تر شدن برنامه‌ها، زبان اسمبلی به برنامه‌نویسان معرفی شد. بدین ترتیب برنامه‌نویسان قادر بودند به کمک مجموعه‌ای از دستورالعمل‌های نمادین زبان ماشین، برنامه‌های بزرگ‌تر و پیچیده‌تری بنویسند. پس از زبان اسمبلی، زبان‌های سطح بالا به دنیای برنامه‌نویسی معرفی شدند تا بتوان به کمک آنها، پیچیدگی برنامه‌های بزرگ را به نحوی کنترل و مدیریت نمود.

آنچه مسلم است، اولین زبانی که به طور گسترده در همه جا مورد استفاده قرار گرفت، زبان فرترن بود. اگرچه این زبان به عنوان اولین گام مؤثر در دنیای برنامه‌نویسی رایانه‌ای شناخته شده، اما باید گفت که فهم برنامه‌های نوشته شده به زبان فرترن دشوار بود و از این رو به راحتی نمی‌توان منطق برنامه را درک کرد.

دهه‌ی ۱۹۶۰ را آغاز برنامه‌نویسی ساخت‌یافته می‌دانند. برنامه‌نویسی ساخت‌یافته روش تقریبا نوین برای برنامه‌نویسی است که زبان‌هایی مثل زبان C از آن پشتیبانی می‌کنند.

استفاده از زبان‌های برنامه‌نویسی ساخت‌یافته برای اولین بار این امکان را برای برنامه‌نویسان فراهم کرد که بتوانند به کمک ساخت‌یافتگی در کدنویسی، برنامه‌های طولانی و نسبتا پیچیده را به راحتی مدیریت کنند. با این حال، زمانی که اندازه‌ی یک پروژه به یک حد نهایی برسد، حتی با استفاده از روش‌های ساخت‌یافته نیز نمی‌توان پیچیدگی برنامه را تحت کنترل درآورد. در اوایل دهه‌ی ۱۹۸۰ بود که این محدودیت در روش‌های ساخت‌یافته باعث شد که در بسیاری از پروژه‌ها از روش‌ها و زبان‌های ساخت‌یافته استفاده نگردد. به منظور حل این مشکل روش جدیدی در برنامه‌نویسی رایانه‌ای ابداع شد که به برنامه‌نویسی شیءگرا [1](OOP) مشهور شد. در بخش‌های بعدی کتاب به طور مفصل به معرفی مفهوم برنامه‌نویسی شیءگرا خواهیم پرداخت و در این‌جا صرفا به ارایه‌ی یک توصیف خلاصه از آن بسنده می‌کنیم: شیءگرایی یک شیوه در برنامه‌نویسی رایانه‌ای است که با استفاده از مفاهیم وراثت[2]، دسته‌بندی[3] و چندریختی[4] می‌تواند برنامه‌نویسان را در سازمان‌دهی و مدیریت برنامه‌های بزرگ و پیچیده یاری نماید.

به طور خلاصه باید گفت که اگرچه زبان C یکی از بزرگ‌ترین زبان‌های برنامه‌نویسی دنیا به شمار می‌آید، اما در زمینه‌ی مدیریت برنامه‌های بزرگ و پیچیده نواقص و محدودیت‌هایی دارد. زمانی که اندازه‌ی برنامه‌ای از یک مقدار معین تجاوز کند، پیچیدگی آن به حدی خواهد رسید که درک کلیت آن بسیار دشوار می‌شود. این حد آستانه به طور دقیق قابل پیش‌بینی نیست و به ماهیت برنامه و مهارت شخص برنامه‌نویس بستگی دارد، اما همیشه حد آستانه‌ای وجود دارد که از آن به بعد عملا مدیریت برنامه غیرممکن خواهد شد. قابلیتی به زبان C++ افزوده شد که با استفاده از آن می‌توان این حد آستانه را شکست و به این ترتیب برنامه‌نویسان قادر خواهند بود که برنامه‌های بزرگ‌تر و پیچیده‌تر را به‌راحتی درک کرده و مدیریت نمایند.

زبان C++ توسط بارنه[5] در سال ۱۹۷۹ ابداع شد؛ زمانی‌که وی در آزمایشگاه بل در Murray Hill واقع در نیوجرسی مشغول به کار بود.

1. Object Oriented Programming
2. Inheritance
3. Encapsulation
4. Polymorphysm
5. Bjarne stroustrup

استراس‌تروپ در ابتدای امر زبان جدید را "زبان C به همراه کلاس‌ها" نامید، اما در سال ۱۹۸۳ این نام به C++ تغییر پیدا کرد. C++ در حقیقت زبان C را با افزودن مفاهیم شیءگرایی توسعه داده است. از آنجا که C++ براساس زبان C و بر پایه‌ی مفاهیم آن ایجاد شده است، بنابراین شامل همه‌ی ویژگی‌ها، خصایص و مزایای آن نیز می‌باشد. دلیل اصلی موفقیت و محبوبیت زبان C++ نیز همین می‌باشد. طراحان این زبان قصد نداشتند که یک زبان کاملا جدید ایجاد کنند، بلکه هدف از تولید زبان C++ بهینه‌سازی یکی از زبان‌های برنامه‌نویسی موفق گذشته بوده است.

زمینه برای ظهور جاوا آماده است

در اواخر دهه‌ی ۱۹۸۰ و اوایل دهه‌ی ۱۹۹۰، برنامه‌نویسی شیءگرا با استفاده از زبان C++ رواج پیدا کرد. در حقیقت این طور به نظر می‌رسید که برنامه‌نویسان توانسته‌اند پس از مدت‌ها یک زبان برنامه‌نویسی کامل و بی‌عیب و نقص داشته باشند. از آنجا که C++ کارایی بالای زبان C و عناصر آن را با مفاهیم شیءگرایی ترکیب نموده، می‌توانست برای تولید طیف گسترده‌ای از برنامه‌های پیچیده و بزرگ به کار رود. اما با این حال هنوز هم نیازمندی‌هایی وجود داشت که باعث می‌شد روند تکاملی زبان‌های برنامه‌نویسی همانند گذشته ادامه پیدا کند. در عرض چند سال شبکه‌ی جهانی وب و اینترنت رواج یافته و به یک نیاز جدی تبدیل شدند و این موضوع نیز روند تکاملی زبان‌های برنامه‌نویسی را تسریع بخشید.

تولد جاوا

زبان جاوا توسط جیمز گسلینگ[1]، پاتریک ناوتون[2]، کریس وارث[3]، اِد فرانک[4] و مایک شریدان[5] در سال ۱۹۹۱ و در شرکت سان میکروسیستم[6] تولید شده است. طراحی و تولید اولین نسخه‌ی کاربردی این زبان، ۱۸ ماه به طول انجامید. این زبان در ابتدا "oak" نام داشت که سرانجام در سال ۱۹۹۵ به جاوا تغییر نام پیدا کرد.

از پیاده‌سازی اولیه‌ی oak در پاییز سال ۱۹۹۲ تا انتشار عمومی جاوا در بهار سال ۱۹۹۵، افراد بسیاری در طراحی و توسعه‌ی این زبان همکاری و مشارکت داشته‌اند. بیل جوی، آرتور وَن هاف، جاناتان پِین، فرانک یلین و تیم لایندهولم از جمله‌ی مهم‌ترین افرادی هستند که در فرایند تکمیل نسخه‌ی ابتدایی زبان جاوا مشارکت داشته‌اند.

شاید اندکی شگفت‌آور باشد اگر بگوییم که هدف اصلی از تولید زبان جاوا به‌کارگیری آن در اینترنت نبود، بلکه انگیزه‌ی اصلی سازندگان جاوا تولید زبانی مستقل از پلت‌فرم (معماری خنثی) بود که بتوان از آن برای تولید نرم‌افزارهایی استفاده کرد که به صورت توکار[7] در ابزارهای الکترونیکی مختلفی چون اجاق‌های ماکروویو و ابزارهای کنترل از راه دور به کار گرفته شوند. همان‌گونه که احتمالا می‌دانید، انواع مختلفی از پردازشگرها به عنوان کنترلر مورد استفاده قرار می‌گیرند و مشکلی که در این زمینه با C، C++ و سایر زبان‌هایی از این دست وجود دارد آن است که این دسته از زبان‌ها اساسا به هدف خاص و برای انجام وظایف خاصی طراحی و کامپایل شده‌اند. اگرچه این امکان وجود دارد که بتوان برنامه‌های نوشته شده به زبان C++ را در هر نوع پردازشگری کامپایل کرد، اما برای انجام این کار لازم است که دسترسی به نسخه‌ی کامل کامپایلر C++ برای پردازشگر مورد نظر وجود داشته باشد. مسئله‌ی اساسی این‌جاست که کامپایلرها اصولا گران بوده و طراحی و تولید آنها نیز زمان‌بر است. بنابراین به یک روش بهینه‌تر به لحاظ زمان و هزینه نیاز بود. به منظور یافتن چنین روشی، گسلینگ و دیگران تلاش کردند تا یک زبان سبک و مستقل از پلت‌فرم تولید کنند، به گونه‌ای که بتوان از آن در تولید کدهای قابل اجرا روی پردازشگرهای مختلف و در محیط‌های مختلف استفاده کرد. این تلاش‌ها در نهایت منجر به تولید زبان جاوا گردید.

1. James Gosling
2. Patrick Naughton
3. Chris warth
4. Ed Frank
5. Mike sheridan
6. Sun Microsystem
7. Embeded

زمانی که سازندگان جاوا در حال بررسی جزئیات آن بودند، عامل مهم دیگری که در آینده‌ی این زبان نقشی حیاتی ایفا می‌کرد پدیدار شد. بله، این عامل مهم همان شبکه‌ی جهانی وب است. اگر شبکه جهانی وب در همان زمان ظهور و تکمیل جاوا، شکل نمی‌گرفت احتمالا جاوا هنوز هم یک زبان برنامه‌نویسی مخصوص تجهیزات الکترونیکی باقی مانده بود. با این حال با پیدایش شبکه‌ی جهانی وب، جاوا گوی سبقت را از سایر زبان‌های برنامه‌نویسی در زمینه‌ی طراحی برنامه‌های اینترنتی ربود؛ چرا که شبکه‌ی جهانی وب بنا بر ماهیت خود به برنامه‌های سبک و حمل‌پذیر نیاز داشته و دارد.

خیلی زود بسیاری از برنامه‌نویسان دریافتند که برنامه‌های سبک و حمل‌پذیر در حرفه‌ی آنها بسیار کارآمدتر از برنامه‌های بزرگ و حجیم هستند. تلاش برای تولید برنامه‌هایی کارا و قابل حمل (مستقل از پلت‌فرم) قدمتی برابر با نظام برنامه‌نویسی دارد و البته این موضوع نسبت به سایر مسائل مربوط به برنامه‌نویسی اهمیت بیشتری داشته است. البته چون در ابتدا، بیشتر رایانه‌های موجود به یکی از سه گروه رقیب اینتل، مکینتاش و یونیکس تعلق داشتند، برنامه‌نویسان به خط قرمزها و محدودیت‌های این گروه‌های سازنده پای‌بند بودند و همین امر نیاز به وجود برنامه‌های قابل حمل را از اهمیت انداخته بود.

با ظهور اینترنت و شبکه‌ی جهانی وب، این نیاز قدیمی بیشتر مور توجه قرار گرفت. به علاوه، با در نظر داشتن این موضوع که اینترنت دربردارنده‌ی مضامین مختلفی است که در سرتاسر جهان منتشر شده و با سیستم‌عامل‌ها و پردازنده‌های مختلفی در ارتباط است، حتی اگر دستگاه‌هایی با پلت‌فرم‌های کاملا متفاوت به اینترنت متصل شوند، همگی باید بتوانند برنامه‌های یکسانی را به اجرا درآورند. بنابراین مسئله‌ی آزار دهنده‌ای که در ابتدا اولویت چندانی نداشت به یک نیاز اساسی تبدیل شد.

در سال ۱۹۹۳ این امر برای اعضای تیم طراحی جاوا مسلم شد که مشکلات مربوط به قابلیت حمل‌پذیری کدهای کنترلرهای توکار، در مورد کدهای اینترنتی نیز وجود دارند. در حقیقت همان مسائلی که جاوا در ابتدا برای حل آن‌ها طراحی شده بود، اینک در مقیاس بزرگ‌تر در مورد کدهای اینترنتی مشاهده می‌شد. این حقیقت باعث شد که جاوا به جای تمرکز بر روی برنامه‌های مربوط به ابزارهای مصرفی الکترونیکی، بر روی برنامه‌های اینترنتی متمرکز گردد. بنابراین اگرچه جرقه اولیه، علاقه به داشتن یک زبان برنامه‌نویسی با معماری خنثی بود، اما اینترنت بود که در نهایت باعث موفقیت نهایی جاوا گردید.

همان‌گونه که در مباحث پیشین نیز اشاره کردیم، جاوا بسیاری از ویژگی‌های خود را از زبان‌های C و C++ به ارث برده است. این ارث‌بری به صورت آگاهانه و هدفمند انجام شده است.

طراحان جاوا بر این مهم واقف بودند که اگر از دستورات نحوی C و C++ که برای کاربران آشناست استفاده کرده و مفاهیم شیءگرایی C++ را در این زبان جدید انعکاس دهند، آنگاه می‌توانند نظر بسیاری از افرادی را که با این زبان‌ها برنامه‌نویسی کار می‌کرده‌اند، به جاوا جلب نمایند.

جاوا علاوه بر شباهت‌های ظاهری، تعدادی از ویژگی‌هایی که زبان‌های C و C++ را به زبان‌هایی موفق تبدیل کرده بود، در خود جای داده است. ابتدا طراحی جاوا انجام گرفت، سپس طراحی، آزمایش و تصحیح جاوا توسط برنامه‌نویسان مجرب و حرفه‌ای انجام شد. در حقیقت جاوا زبانی است که بر پایه‌ی نیازها و تجربیات افراد سازنده‌ی آن ایجاد شده و از این رو گفته می‌شود جاوا زبانی برای برنامه‌نویسان است.

از سوی دیگر، جاوا زبانی منسجم و به لحاظ منطقی سازگار است. به علاوه صرف‌نظر از محدودیت‌هایی که محیط اینترنت ایجاد می‌کند، جاوا برای کاربران خود (برنامه‌نویسان) امکان کنترل کامل را فراهم می‌کند. یعنی اگر شما بتوانید خوب برنامه بنویسید، برنامه‌ی نوشته شده این ویژگی را به خوبی انعکاس خواهد داد. همین موضوع در مورد

برنامه‌های ضعیف نیز صادق است. به بیان دیگر، جاوا یک زبان آموزشی نیست بلکه برای برنامه‌نویسان حرفه‌ای طراحی شده است.

به دلیل شباهت‌های بسیاری که بین جاوا و C++ وجود دارد می‌توان جاوا را نسخه‌ی اینترنتی C++ دانست. اما بیان چنین امری قطعا اشتباه است، چون جاوا به لحاظ منطقی و عملیاتی نسبت به C++ برتری دارد. علی‌رغم اینکه جاوا به واقع از زبان C++ تاثیر گرفته اما نباید آن را نسخه‌ی بهینه‌سازی شده‌ی C++ دانست. به عنوان مثال اگرچه جاوا به هیچ وجه با زبان C++ قابل مقایسه نبوده و با آن همخوانی ندارد، اما به دلیل وجود شباهت‌های بسیار، اگر شما برنامه‌نویس حرفه‌ای C++ باشید به‌راحتی می‌توانید با زبان جاوا نیز کدنویسی کنید. نکته‌ی حائز اهمیت دیگر آن است که جاوا طراحی نشده تا جایگزین C++ گردد، بلکه طراحی این زبان به این منظور بوده که در حل مسائل خاصی از آن استفاده شود که C++ برای آنها کارآمد نمی‌باشد.

همان‌گونه که در ابتدای این فصل بدان اشاره شد، زبان‌های برنامه‌نویسی به دو دلیل عمده ایجاد و تکامل یافتند: برای تطبیق با محیط‌های در حال تغییر رایانه‌ای و نیز جهت به‌کارگیری پیشرفت‌های ایجاد شده در هنر برنامه‌نویسی. در حقیقت، تغییرات محیطی یکی از دلایل اصلی تولید برنامه‌های مستقل از پلت‌فرمی هستند که برای انتشار در اینترنت ایجاد می‌شوند. جاوا از تغییراتی که افراد در نوشتن برنامه‌ها ایجاد می‌کنند نیز پشتیبانی می‌کند. به عنوان مثال، جاوا الگوهای شیءگرای C++ را بهینه‌سازی و اصلاح کرده، پشتیبانی یکپارچه از قابلیت چند گره‌ای[1] را فراهم نموده و همچنین کتابخانه‌ای ایجاد کرده که دسترسی به اینترنت را ساده‌تر می‌نماید. در نهایت باید گفت که صرفا این ویژگی‌های منحصر به فرد نیست که توجه افراد بسیاری را به جاوا جلب کرده، بلکه خود زبان جاوا در کل توانسته این محبوبیت را به دست آورد. در حقیقت جاوا پاسخ مناسبی به نیازها و مطالبات دنیای جدید و وسیع برنامه‌نویسی رایانه‌ای است. این زبان برای برنامه‌نویسی اینترنتی طراحی شده بود و این در حالی است که زبان C برای برنامه‌نویسی سیستمی مورد استفاده قرار می‌گرفت. در حقیقت این برنامه‌نویسی اینترنتی بود که به عنوان یک حرکت انقلابی، دنیای برنامه‌نویسی رایانه‌ای را تحت تاثیر خود قرار داد.

ارتباط با C#

همان‌طور که گفته شد زبان قدرتمند و غنی جاوا توانست دنیای در حال توسعه‌ی زبان‌های برنامه‌نویسی را تحت تاثیر ویژگی‌های منحصر به فرد خود قرار دهد. به گونه‌ای که بسیاری از خصوصیات، ساختارها و مفاهیم نوین و ابتکاری جاوا، تبدیل به بخش‌های اساسی و پایه‌ای زبان برنامه‌نویسی جدیدتر شدند. موفقیت جاوا و دلایل این موفقیت به قدری حائز اهمیت است که نمی‌توان به سادگی از آن عبور کرد.

شاید زبان C# بهترین مثال برای تاثیر جاوا بر روی زبان‌های برنامه‌نویسی باشد. این زبان که توسط شرکت مایکروسافت برای پشتیبانی از پلت‌فرم Net. ایجاد شده، شباهت‌های بسیاری به زبان جاوا دارد. به عنوان مثال، هر دوی این زبان‌ها از دستورات نحوی یکسانی استفاده کرده، برنامه‌نویسی توزیع شده را پشتیبانی نموده و مدل شیءگرای مشابهی را به کار می‌برند. البته تفاوت‌هایی نیز میان این دو زبان وجود دارد، اما مفاهیم بنیادی و اساس کار هر دو زبان بسیار شبیه یکدیگر است. بسیاری از افراد، تولید زبان C# را "گرده‌افشانی[2]" جاوا تلقی نموده‌اند که همین امر گواه روشنی است بر این ادعا که جاوا روش تفکر و نیز نحوه‌ی استفاده از زبان‌های برنامه‌نویسی را تغییر داده و براساس ویژگی‌های خود اصلاح نموده است.

1. Cross pollination
2. Multithreading

جاوا چگونه توانست اینترنت را تغییر دهد

اینترنت سکوی پرتاب جاوا به صدر جدول زبان‌های برنامه‌نویسی بوده و جاوا نیز به نوبه‌ی خود تاثیر بسزایی در پیشرفت و تکامل اینترنت داشته است. جاوا علاوه بر تسهیل در برنامه‌نویسی تحت وب، توانست روش نوینی در برنامه‌نویسی تحت شبکه با عنوان "اپلت[1]" ابداع کند که شیوه‌ی تولید محتوای آنلاین را به طور کلی تغییر داد. به علاوه جاوا توانست دو مورد از مهم‌ترین مسائل مربوط به دنیای اینترنت را برطرف نماید: امنیت و قابلیت حمل‌پذیری. بیایید کمی دقیق‌تر به این دو مسئله بپردازیم.

اپلت‌های جاوا

یک اپلت، در حقیقت نوع خاصی از برنامه‌های جاواست که به هدف انتقال در محیط اینترنت ایجاد شده و می‌تواند به صورت خودکار توسط مرورگرهای اینترنتی سازگار با جاوا اجرا شود.

همچنین اپلت‌ها بدون آنکه نیاز باشد کاربر عملیات خاصی را انجام دهد، به راحتی قابل دانلود هستند. یعنی اگر کاربر بر روی لینکی که دربردارنده‌ی یک اپلت است کلیک کند، اپلت به صورت خودکار بارگیری (دانلود) شده و در مرورگر اینترنتی اجرا می‌شود.

هدف اصلی در طراحی اپلت‌ها، کوچک و قابل حمل بودن برنامه‌ها می‌باشد. از آنها معمولا برای نمایش داده‌های سرور، مدیریت ورودی‌های کاربران یا تولید توابع کوچکی مثل یک ماشین‌حساب ساده که به جای اجرا روی رایانه‌ی میزبان (سرور) به صورت محلی اجرا می‌گردد، استفاده می‌شود. در حقیقت، اپلت‌ها این امکان را فراهم می‌آورند که بتوان برخی از قابلیت‌ها و توابع را از رایانه‌ی میزبان (سرور) به رایانه‌ی سرویس گیرنده[2] منتقل کرد.

تولید اپلت‌ها دنیای اینترنت را دستخوش تغییرات اساسی نمود، زیرا اپلت‌ها تعداد و تنوع اشیای قابل انتقال در فضای مجازی را توسعه دادند.

به طور کلی دو گروه از اشیا می‌توانند در اینترنت بین سرویس گیرنده و سرور منتقل شوند:

۱. اطلاعات منفعل و مجهول[3]؛

۲. برنامه‌های فعال[4] و پویا.

به عنوان مثال وقتی شما ایمیلی را مطالعه می‌کنید، در حقیقت در حال مشاهده‌ی داده‌های منفعل هستید. یا حتی وقتی برنامه‌ای را دانلود می‌کنید، کد برنامه‌ی مورد نظر تا زمانی که اجرا نشود به صورت داده‌ی منفعل باقی خواهد ماند. در مقابل، اپلت‌ها برنامه‌های پویایی هستند که قابلیت خود اجرایی[5] دارند. چنین برنامه‌ای به عنوان یک عامل فعال بر روی رایانه‌ی سرویس گیرنده اجرا می‌شود، اما در ابتدای امر بر روی رایانه‌ی میزبان قرار داشته است.

هرچند پویا بودن، یک نقطه قوت برای برنامه‌های تحت شبکه محسوب می‌شود، اما در عین حال با مشکلات جدی‌ای در عرصه‌ی امنیت و قابل حمل بودن نیز مواجه هستند. واضح است که وقتی برنامه‌ای از اینترنت دانلود شده و به صورت خودکار بر روی یک رایانه‌ی سرویس گیرنده اجرا می‌گردد، باید به لحاظ امنیتی قابل اطمینان باشد، به طوری که نتواند هیچ تاثیر مخربی بر روی رایانه‌ی سرویس گیرنده داشته باشد. به علاوه چنین برنامه‌ای باید بتواند در محیط‌ها و سیستم‌عامل‌های مختلف نیز اجرا شود. در ادامه درخواهید یافت که جاوا توانسته است این مسائل را با استفاده از روش‌های ظریف و مؤثری به خوبی برطرف نماید. اجازه دهید نگاه عمیق‌تری به هر یک از این مسائل داشته باشیم.

---

1. Applet
2. Client
3. Passive Data
4. Active programs
5. Self executing