**فصل 1**

**مقدمه**

**اَبستِرَکشن**

**آشناسازی**

ابسترکشن به معنای پنهان سازی جزئیات و فقط آشکارسازی ویژگی های ضروری مفاهیمی خاص یا اشیا می باشد. دانشمندان علوم کامپیوتر از ابسترکشن برای فهم و حل مشکلات و مرتبط کردن پاسخ های خود با کامپیوتر در بعضی از زبان های کامپیوتری خاص استفاده می کنند. ما این فرایند را با امتحان کردن حل مشکل زیر با استفاده از زبان برنامه نویسی جاوا نمایش می دهیم.

**مشکل:** به یک مستطیل 4.5 فوت عرض و 7.2 فوت طول داده شده است، مساحت آن را محاسبه کنید.

می دانیم که مساحت مستطیل برابر است با عرض آن ضرب در طول آن. پس تمام کاری که باید انجام دهیم تا مسئله بالا را حل کنیم، این است که 4.5 را در 7.2 ضرب کرده و جواب را به دست آوریم. سوال اینجاست که چطور راه حل بالا را در جاوا بیان کنیم، که کامپیوتر بتواند عملیات محاسبه را اجرا کند.

**دیتا ابسترکشن**

حاصل ضرب 4.5 در 7.2 در جاوا به شکل 4.5 \* 7.2 بیان می شود. در این عبارت علامت \* عملگر ضرب را نشان می دهد. 4.5 و 7.2 حروف عددی نامیده می شوند. با استفاده از دکتر جاوا، ما می توانیم مستقیماً در پنجره تعاملات عبارت 4.5 \* 7.2 را تایپ کرده و جواب را مشاهده کنیم.

حالا فرض کنیم ما مسئله را به مساحت مستطیلی به عرض 3.6 و طول 9.3 تغییر دادیم.در کل آیا مسئله اصلی واقعاً تغییری کرده است؟ به عبارت دیگر، ذات مشکل اصلی تغییر کرد؟ با این وجود، فرمول محاسبه جواب همچنان همان است. تمام کاری که باید کرد، وارد کردن 3.6 \* 9.3 است. چه چیزی هست که تغییر نکرده است ( ثابت )؟ و چه چیزی تغییر کرده است ( متغیر )؟

**تایپ ابسترکشن**

مسئله ای که هنوز در آن تغییری نکرده است این است که همچنان با همان شکل هندسی سر و کار دارد، یک مستطیل، توصیف شده در شرایط همان ابعاد، عرض آن و طول آن. چیزی که به سادگی تغییر کرده مقادیر عرض و طول است. فرمول محاسبه مساحت مستطیل با عرض و طول معلوم تغییری نکرده است: (طول \* عرض)

width \* height

اصلا به اینکه مقادیر مشخص واقعی طول و عرض چه هستند اهمیتی نمی دهد. چیزی که به آن اهمیت می دهد این است که مقادیر طول و عرض باید طوری باشند که عملیات ضرب قابل فهم باشد. چطور ما ثابت های بالا را در جاوا بروز دهیم؟

ما فقط می خواهیم فکر کنیم طول و عرض داده شدۀ مستطیل به عنوان عناصری از مجموعۀ اعداد حقیقی هستند. در محاسبه، ما گروهی از مقادیر با خصوصیات یکسان را مجموعه در نظر میگیریم و به آن نام تایپ یا نوع میدهیم. در جاوا انواع دابل یا اعشاری مجموعه ای از اعداد حقیقی هستند که در کامپیوتر به روش مخصوصی اجرا می شوند. جزئیات این ارائۀ داخلی برای هدف ما بی اهمیت است و بنابراین می توان از آن چشم پوشی کرد. به علاوه نوع دابل، جاوا بسیاری از نوع داده های از پیش ساخته شده مثل نوع صحیح را تهیه کرده تا نمایندۀ مجموعه اعداد صحیح باشد و کاراکتر که نماینده مجموعه ای از کاراکترها باشد. ما آن ها را در حوزه مورد نیاز خودشان با مثال های بعدی آزمایش و استفاده خواهیم کرد. برای مسئله ما، ما فقط نیاز داریم تا خودمان را به نوع داده دابل محدود کنیم. ما می توانیم طول و عرض مستطیل را به نوع دابل مثل ادامه در جاوا تعریف کنیم.

double width;

double height;

دو دستور بالا، تعریف متغیر نامیده می شوند که به طول و عرض، نام متغیر گفته می شود. در جاوا، یک متغیر نمایندگی بخشی از مکان حافظه را در کامپیوتر بر عهده دارد. ما اول متغیر را با معین کردن نوع آن تعریف میکنیم، بعد از نوع، نام متغیر می آید، و تعریف را با یک سمی کالن خاتمه می دهیم. این یک قائده سینتکس یا نحو جاوا است. نقض کردن قانون سینتکس سبب ارور می شود. زمانی که در این روش ما متغیر تعریف می کنیم، محتوای حافظۀ مربوط به متغیر با یک مقدار مشخص پیش فرض که توسط جاوا تعیین می شود مقدار دهی اولیه می شود. برای متغیرهای از نوع دابل، مقدار پیش فرض صفر است.

**توجه:** از بخش تعاملات دکترجاوا برای سنجیدن طول و عرض و اینکه مشخص شود مقادیر آن ها صفر ذخیره شده است، استفاده کنید.

هنگامی که ما متغیر های طول و عرض را تعریف کردیم، می توانیم مسئله را با نوشتن عبارتی که مساحت مستطیل وابسته به شرایط طول و عرض را مانند ادامه محاسبه کند.

width \* height

مشاهده کنید که دو تعریف متغیر با هم و عبارت ضرب برای محاسبۀ محیط که در بالا ارائه شده، مستقیماً توضیح مسئلۀ دو عدد حقیقی که توسط طول و عرض یک مستطیل نمایندگی می شود ( و سطح بالا فکر کردن اینکه پاسخ مسئله چه باید باشد ) مساحت برابر طول ضرب در عرض معنی می شود. ما تازه ثابت های مسئله و پاسخ آن را بیان کردیم. حالا چطور ما طول و عرض را در جاوا تغییر دهیم؟ ما چیزی که عملگر انتساب نامیده می شود را استفاده می کنیم. برای نسبت دادن مقدار 4.5 به متغیر عرض و مقدار 7.2 به متغیر طول، ما دستور انتساب بعدی را در جاوا می نویسیم.

width = 4.5;

height = 7.2;

قائدۀ سینتکس برای جاوا به این شکل است: اول نام متغیر را بنویسید، سپس با علامت تساوی کد را ادامه دهید، بعد از علامت تساوی یک عبارت جاوا را بنویسید، و با یک سمی کالن خاتمه دهید. معنای چنین عبارتی این است : عبارت سمت راست تساوی را محاسبه کن، و مقدار نتیجه را به مکان حافظه ای که نمایندۀ نام متغیر سمت چپ تساوی است، نسبت بده. اگر نوع عبارت سمت راست زیر مجموعه ای از نوع متغیر سمت چپ نباشد ارور رخ می دهد. حالا اگر دوباره طول \* عرض (به انگلیسی) را محاسبه کنیم، ( با استفاده از پنجره تعاملات دکتر جاوا)، ما باید جواب مطلوب را بگیریم. تا اینجا قابل فهم است، اما اینجا کمی مشکل وجود دارد: ما باید عبارت طول \* عرض را هر بار که بخواهیم مساحت مستطیلی با طول و عرض معلوم را محاسبه کنیم، تایپ کنیم. این شاید برای فرمولی به این سادگی قابل انجام باشد، اما اگر فرمول چیزی بسیار پیچیده تر باشد، مثل محاسبه اندازه قطر یک مستطیل چه باید کرد؟ هر بار دوباره تایپ کردن فرمول شانسی را برای بروز خطا های آینده فراهم می کند. آیا راهی وجود دارد که سبب شود کامپیوتر فرمول را به حافظه بسپارد و پشت صحنه محاسبه را اجرا کند که ما خودمان مجبور نباشیم آن را حفظ کنیم و دوباره آن را بنویسیم کنیم؟ پاسخ مثبت است، و این کمی زحمت بیشتر می طلبد تا به این هدف در جاوا برسیم.

چیزی که می خواهیم انجام دهیم این است که مشابه یک کادر مشکی ایجاد کنیم که به عنوان ورودی با استفاده از یک دکمه دو عدد حقیقی می پذیرد ( نوع دابل را یادآوری کنید ). وقتی ما دو عدد را قرار می دهیم و دکمه را فشار می دهیم، کادر مشکی با محاسبه حاصل ضرب دو عدد ورودی و نشان دادن جواب، جادو میکند، که ما آن را به عنوان جواب مسئله تفسیر میکنیم که طول و عرض آن توسط دو عدد ورودی داده شده است. این کادر مشکی در ذات یک ماشین حساب مخصوص می باشد که فقط قابلیت محاسبۀ یک چیز را دارد: مساحت یک مستطیل با طول و عرض داده شده. برای ایجاد این کادر در جاوا، ما از ساختاری به نام کلاس استفاده می کنیم، که به مثال ادامه شبیه است.

class AreaCalc {

double rectangle(double width, double height) {

return width \* height;

}

}

این کد جاوا چنین چیزی معنی می شود: نام کلاس طرحی از ماشین محاسبه مخصوصی است که توانمند است دو عدد دابل را بپذیرد، یکی را با برچسب طول و دیگری را با برچسب عرض بشناسد، حاصل ضرب آن ها را محاسبه کند و نتیجه را برگرداند. به این محاسبه نامی داده شده است: مستطیل. در نگارش جاوا این یک متد برای کلاس محاسبه مساحت نام دارد.

اینجا مثالی آورده شده که ما چطور از کلاس محاسبه مساحت استفاده کنیم تا مساحت مستطیلی به طول 7.2 و عرض 4.5 را به دست آوریم. در بخش تعاملات دکتر جاوا، خطوط کد پایین را وارد کنید.

AreaCalc calc = new AreaCalc();

calc.rectangle(4.5, 7.2)

اولین خط کد کلمه "محاسبه" را به عنوان متغیری از نوع محاسبه مساحت تعریف می کند و آن را به نمونه ای از کلاس محاسبه مساحت نسبت می دهد. کلمه "جدید" کلمه ای کلیدی در جاوا می باشد. این یک مثال از این است که چه چیزی عملگر کلاس نامیده می شود. عملیاتی را روی کلاس انجام می دهد و نمونه ای از کلاس داده شده را می سازد ( شئ هم نامیده می شود ). خط دوم کد صدا زدن شئ محاسبه است که وظیفۀ مستطیل را اجرا کند که عرض با مقدار 4.5 و طول با مقدار 7.2 منسوب شده است. برای به دست آوردن مساحت یک مستطیل 8.4 در 5.6 ما دوباره به سادگی از همان ماشین حسابِ "محاسبه" استفاده میکنیم:

calc.rectangle(5.6, 8.4);

پس به جای حل کردن فقط یک مسئله ( طول یک مستطیل 7.2 و عرض آن 4.5 داده شده است، مساحت آن را بیابید )

ما ماشینی را ساخته ایم که می تواند مساحت هر مستطیل داده شده را محاسبه کند. اما راجع به محاسبه کردن مساحت مثلث قائم الزاویه با ارتفاع 5 و قائده 4 چه؟ ما به سادگی نمی توانیم از این ماشین حساب استفاده کنیم. ما یک ماشین حساب مخصوص دیگر نیاز داریم، نوعی که بتواند مساحت یک دایره را محاسبه کند.

حداقل دو نوع طراحی مختلف برای چنین ماشین حسابی وجود دارد.

* کلاسی جدید تحت عنوان "محاسبه مساحت 2" با یک متد به نام مثلث قائم الزاویه با دو پارامتر ورودی از نوع دابل ایجاد کنید. این روش با اینکه یک ماشین حسابِ مساحتِ متفاوت با یک دکمه به اسم مثلث قائم با دو درگاه برای ورودی طراحی کنیم، یکی است.
* به کلاس ماشین حسابِ مساحت، متدی تحت عنوان مثلث قائم با دو پارامتر ورودی از نوع دابل اضافه کنیم. این روش با اینکه یک ماشین حساب مساحت با دو دکمه ایجاد کنیم یکی است: یکی برچسبِ مستطیل خورده که دو درگاه ورودی دارد، و دیگری مثلث قائم، که آن هم دو درگاه ورودی دارد.

در هر دو طراحی، این مسئولیت کاربر ماشین حساب است که ماشین حساب مناسب را انتخاب کند یا دکمه مناسب را روی ماشین حساب فشار دهد تا به طرز صحیح مساحت شکل هندسی داده شده را به دست بیاورد. از آن جایی که دو محاسبه، پارامتر های ورودی مشابه از دقیقاً یک نوع داده رادارند، کاربر ماشین حساب باید مراقب باشد که سردرگم نشود. اگر فقط دو نوع شکل برای انتخاب وجود داشته باشد، شاید خیلی مایۀ ناراحتی نباشد: مستطیل و مثلث قائم. اما اگر کاربر مجبور باشد میان صد ها شکل مختلف انتخاب کند چه؟ یا بهتر، تعداد بی شماری از اشکال؟ چطور ما به عنوان برنامه نویس، ماشین حسابی بسازیم که بتواند از پس بی نهایت تعداد از اشکال برآید؟ پاسخ ابسترکشن است. برای انگیزه پیدا کردن راجع به مفهومی کردن مسئله، اجازه دهید گریزی بزنیم و راجع به رفتار یک کودک بیندیشیم!

**مدل سازی یک شخص**

برای چندین سال ابتدایی زندگی، پیتر هیچ سرنخی نداشت که تاریخ تولدها کی هستند، چه برسد به تاریخ تولد خودش. او به پاسخ دادن به پرسش شما راجع به تاریخ تولد خودش ناتوان بود. این والدین او بودند که به زحمت جشن های تولد او را ماه ها قبل برنامه ریزی می کردند. پس ما می توانیم پیتر را کاملاً یک نادان فرض کنیم که دارای هوش بسیار پایین و توانایی کم است. حالا پیتر یک دانشجو است. بخشی از مغر او هست که تاریخ تولد خودش را در آن ذخیره کرده است. تاریخ تولد، 12 سپتامبر 1985 است! پیتر حالا از باهوش هم باهوش تر است. او می تواند بفهمد چند ماه دیگر تا تولد بعدی خودش مانده و لیست آرزوهایش را دو ماه قبل از تولدش ایمیل کند. چطور ما شخصی باهوش مثل پیتر را مدل سازی کنیم؟ مدل سازی چنین شخصی شامل مدل سازی های زیر می باشد:

* یک تاریخ تولد
* محاسبه تعداد ماه های مانده تا تولد بعدی، با ماه فعلی معلوم

یک تاریخ تولد شامل یک ماه، یک روز و یک سال است. هر کدام از این داده ها می تواند توسط یک عدد صحیح نمایش داده شود، که در جاوا عددی از نوع اینت نام دارد. مثل محاسبه مساحت یک مستطیل، محاسبه تعداد ماه ها تا تولد بعدی که دادۀ مسئله ماه فعلی است، میتواند به عنوان یک متد از کلاسی نمایش داده شود. در این وضعیت کاری که ما انجام خواهیم داد با ماشین حساب مساحت متفاوت است. ما هر دو داده را جمع خواهیم کرد ( منظور تاریخ تولد می باشد ) و محاسبه شامل تاریخ تولد در یک کلاس است. به گروه ساختن از داده ها و محاسبات روی داده در یک کلاس، اِنکَپسولیشن می گویند. کد جاوای پایین، شخصی باهوش را مدل سازی می کند که می داند چگونه تعداد ماه های مانده تا تولدش را محاسبه کند. شمارۀ خطوط کد نمایش داده شده، برای مراجعۀ آسان آن جا قرار داده شده اند و بخشی از کد نیستند.

1 public class Person {

2 /\*\*

3 \* All data fields are private in order to prevent code outside of this

4 \* class to access them.

5 \*/

6 private int \_bDay; // birth day

7 private int \_bMonth; // birth month; for example, 3 means March.

8 private int \_bYear; // birth year

9 /\*\*

10 \* Constructor: a special code used to initialize the fields of the class.

11 \* The only way to instantiate a Person object is to call new on the constructor.

12 \* For example: new Person(28, 2, 1945) will create a Person object with

13 \* birth date February 28, 1945.

14 \*/

15 public Person(int day, int month, int year) {

16 \_bDay = day;

17 \_bMonth = month;

18 \_bYear = year;

19 }

20 /\*\*

21 \* Uses "modulo" arithmetic to compute the number of months till the next

22 \* birth day given the current month.

23 \* @param currentMonth an int representing the current month.

24 \*/

25 public int nMonthTillBD(int currentMonth) {

26 return (\_bMonth - currentMonth + 12) % 12;

27 }

28 }

* خط اول، کلاسی به نام "شخص" تعریف میکند. آکولاد باز شده در انتهای خط، و جفتِ بسته شدۀ آن در خط 28 محتوای کلاس شخص را محدود می کند. کلمه کلیدی پابلیک یا عمومی تعیین کنندۀ دسترسی نام دارد و یعنی همۀ کد های جاوا در سیستم می توانند به آن مراجعه کنند.
* خطوط 2 الی 5 کامنت هستند. هر چیزی که میان \*/ و /\* بیاید، توسط کامپایلر نادیده گرفته می شود.
* خطوط 6 الی 8 تعداد 3 متغیر از نوع صحیح را تعریف میکند. این متغیر ها فیلد های کلاس نام دارند. کلمه کلیدی پرایوت یا خصوصی نیز تعیین کنندۀ دسترسی دیگری هست که از دسترسی کدی خارج از کلاس، جلوگیری می کند. تنها کد داخل خود کلاس می تواند به آن دسترسی داشته باشد. هر فیلد با یک کامنت توسط // در انتهای خط کد محدود شده. پس دو راه برای کامنت کردن کد در جاوا وجود دارد: با \*/ شروع کنید و با /\* خاتمه دهید یا با // شروع کنید و با انتهای خط کد خاتمه دهید.
* خطوط 9 الی 14 کامنت هستند.
* خطوط 15 تا 19 چیزی که سازنده نام دارد را تشکیل می دهند. این برای مقدار دهی اولیه کردن فیلد های کلاس با بعضی از مقادیر خاص به کار می رود. نام سازنده باید دقیقاً از نظر املایی برابر نام کلاس باشد. سازنده اینجا پابلیک است، به این معنی که می تواند توسط کد خارج از کلاس "شخص" با استفاده از عملگر "جدید" صدا زده شود. برای مثال :

new Person(28, 2, 1945)

نمونه ای از کلاس شخص با روز تولد 28، ماه تولد 2، و سال تواد 1945 خواهد ساخت.

* خطوط 20 الی 24 کامنت هستند.
* خط 23 یک قالب خاص برای ثبت کردن پارامتر های یک متد است. این قالب، فرمتِ جاواداک نام دارد. ما در ماژولی دیگر بیشتر دربارۀ جاواداک یاد خواهیم گرفت.
* خطوط 25 الی 27 تعریف یک متد در کلاس شخص را تشکیل می دهند.
* خط 26 فرمول محاسبه تعداد ماه های قبل از تولد بعدی با استفاده از عملگر باقی مانده % است. باقی ماندۀ متغیر ایکس به متغیر وای، باقی ماندۀ تقسیم صحیح بین مقسومِ ایکس و مقسوم علیه وای را می دهد.

**اشیا و کلاس ها**

**اشیا**

در جهان واقعی، موجودیت اشیا هست که باعث تشکیل شدن جهان می شود. هر چیزی که در جهان اتفاق میفتد به تعاملات میان اشیا در جهان ارتباط دارد. مثلاً اتم ها، که خودشان شئ هستند، ترکیب شده اند تا مولکول ها و اشیا بزرگتر را شکل دهند، تعاملات موجودیت ها در جهان می تواند به عنوان تعاملات بین هر دو شئ مفرد ("اتمی") یا به صورت ترکیب ("ترکیب شده") در نظر گرفته شود. جهان واقعی شامل تعداد خیلی زیادی از اشیا هست که از راه های مختلف بسیاری با هم در تعامل هستند. در حالی که هر شی ممکن است خیلی پیچیده نباشد، هزاران تعاملات آن ها سراسر پیچیدگی دنیای طبیعی را می سازد. این پیچیدگی است که ما می خواهیم در سیستم نرم افزاری خود به دست آوریم. در یک سیستم نرم افزاری شئ گرا، اشیا موجودیت هایی هستند که استفاده شده تا یک مدل یا بخشی خاص از سیستم را نمایش دهد.

اشیا واحد های اصلی استفاده شده برای ساختن مدل های مجازی هستند.

چندین طرز فکر برای برنامه نویسی شئ گرا وجود دارد، که اندکی روی دیدگاه نسبت به اشیا تفاوت دارند. اینجا ما موضع "رفتارگرایی" (شرط خودمان) را خواهیم گرفت:

**یک شئ فقط توسط رفتارهایش توصیف می شود.**

این گونه ذاتاً یک شئ را بر اساس چگونگی روشی که با دنیایش در حال تعامل است، تعریف می کند. یک شئ که با چیز دیگری تعامل ندارد، به طور موثر وجود ندارد. دسترسی به داده ذخیره شدۀ داخلی ضرورتاً توسط برخی از رفتار های تعریف شدۀ شئ صورت می پذیرد. به هر حال این برای یک موجودیت خارجی غیر ممکن است که حقیقتاً "بداند" یک قسمت از داده داخل یک شئ دیگر در حال ذخیره شدن است.