نام کتاب: دنیای شی گرا

نویسنده: امیررضا اندیشمند

سال انتشار: 1392

نوع انتشار: الکترونیکی

ایمیل: amirrezaandishmand2020@gmail.com

فهرست مطالب

مقدمه:

این کتاب برای درک بهتر مفاهیم شی گرایی نوشته شده است و در آن از کد های زبان برنامه نویسی صحبت نمی شود پس به این معنی است که پس از خواندن کتاب باید نحوه پیاده سازی مطالب شی گرایی را در زبان مورد نظر خود یاد بگیرید.

اگر با مفاهیم پایه برنامه نویسی آشنایی ندارید اکیداً توصیه می کنم قبل از خواندن کتاب با مباحث پایه زبان برنامه نویسی مورد نظرتان آشنا شوید زیرا در درک شما از

این مباحث تاثیر بسزایی دارد.

از دلایلی که من شروع به نوشتن این کتابچه کردم این بود که خود من هم در یادگیری این مسائل مشکل داشتم. سایت های بسیاری را مطالعه کردم تا بتوانم به یک مفهوم ساده و قابل درک از مباحث شی گرایی برسم و تمام آنچه از شی گرایی درک کردم را در این کتاب وارد کردم.

امیدوارم قدم کوچکی در رشد شما داشته باشم.

سعی من بر این بوده به صورت مفهومی مطالب شی گرایی را تا جای ممکن ساده بیان کنم و تا حد ممکن از مثال های تصویری برای ارائه بهتر مفهوم استفاده کنم. البته این مطلب در گرو تلاش شما برای یادگیری هر بخش در زبانی که با آن برنامه نویسی

می کنید نیز هست.

جالب است قبل از شروع با تاریخچه پیدایش شی گرایی آشنا شویم.

تاریخچه برنامه نویسی شی گرا:

اصطلاح شناختی "اشیاء" و "گرا" در معنای مدرن برنامه نویسی شیء گرا اولین بار در MIT در اواخر دهه ۱۹۵۰ و اوایل دهه ۱۹۶۰ ظاهر شد. در محیط گروه هوش مصنوعی، در اوایل سال ۱۹۶۰، مفهوم "شی" می تواند به اقلام مشخص شده (اتم LISP) با خواص (attributes) اشاره کند. مطرح شد.

Sutherland مفهوم "شی" و "نمونه" (با مفهوم کلاس تحت عنوان "استاد" یا "تعریف") را تعریف می کند، هرچند این تعاریف او متمرکز بر تعامل گرافیکی است.

برنامه نویسی شی گرا اولین بار در سال ۱۹۶۰ با زبان سیمولا اجرا شد. که مفاهیم مهمی را که امروزه بخشی اساسی از برنامه نویسی شی گرا هستند را معرفی کرد. و مفاهیمی مانند کلاس و شی، ارث و اتصال پویا ارائه شد، سیمولا همچنین برای برنامه نویسی و امنیت داده طراحی شده است.

در دهه ۱۹۷۰ اولین نسخه زبان برنامه نویسی Smalltalk در آلبوم Xerox PARC توسط آلن کای، دن انگلز و آدل گلدبرگ که از برنامه نویسی شی گرا پشتیبانی می کرد . ساخته شد. Smaltalk-71 شامل یک محیط برنامه نویسی بود و به صورت پویا تایپ می شد و در ابتدا تفسیر می شد. Smalltalk برای استفاده ، از جهت گیری شیء در سطح زبان و محیط توسعه گرافیکی آن اشاره کرد. Smalltalk از طریق نسخه های مختلفتوسعه یافت و علاقه به این زبان افزایش یافت. در حالی که Smalltalk تحت تاثیر ایده های معرفی شده در Simula 67 بود، آن را به عنوان یک سیستم کاملا پویا طراحی شده بود که در آن کلاس ها می توانند ایجاد و تغییر پویا شوند.

در سال ۱۹۸۱، گلدبرگ برنامه های Smalltalk و برنامه های شیء گرا را به مخاطبان گسترده تری معرفی کرد. در سال ۱۹۸۶ انجمن ماشین آلات رایانه ، اولین کنفرانس برنامه نویسی مبتنی بر Object Oriented (شی گرایی)، سیستم ها، زبان ها و برنامه های کاربردی (OOPSLA) را سازماندهی کرد. که به طور غیر منتظره با حضور ۱۰۰۰ نفر استقبال شد.

از زبان هایی که از شی گرایی پشتیبانی می کنند می توان به , c# , java , lisp ++c و python اشاره کرد.

برنامه نویسی شی گرا یا OOP چیست؟

Oop مخفف object oriented programming (برنامه نویسی شی گرا) است

Oop یک شیوه برنامه نویسی است که به ما در درک راحت تر برنامه نویسی کمک می کند و همچنین باعث می شود ما با تطابق اشیا در دنیای واقعی بتوانیم به شکل راحت تری کد نویسی کنیم. این مسئله این سبک برنامه نویسی را برای مدت زیادی به محبوب ترین شیوه برنامه نویسی تبدیل کرده.

Oop یک شیوه برنامه نویسی است که مفاهیمی که با آن سر و کار داریم اشیا هستند.

تعریف بالا یک تعریف کوتاه و خلاصه از مفهوم oop است

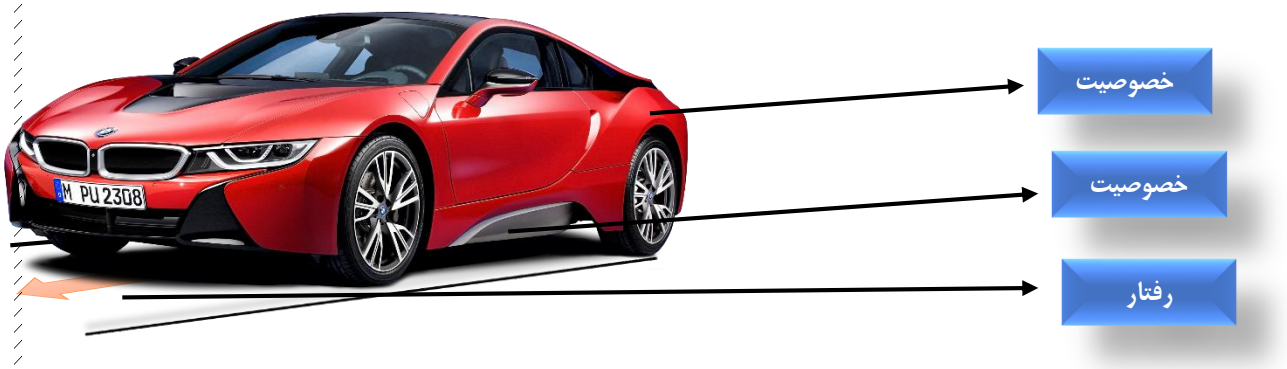
شی گرایی از 7 مفهوم تشکیل شده که در ادامه به بررسی این مفاهیم می پردازیم.

شی، کلاس، ارث بری، کپسوله سازی، انتزاع و چند ریختی از مباحث شی گرایی هستن

لازم به ذکر است 4 مفهوم اصلی شی گرایی متشکل از چند ریختی، ارث بری، کپسوله سازی و انتزاع می باشد.

لازم است قبل از شروع مباحث شی گرایی با دو مفهوم مهم آشنا شوید.

ماشین رو به رو دارای یکسری خصوصیات و رفتار است که در برنامه نویسی آن را به شکل زیر بیان می کنیم



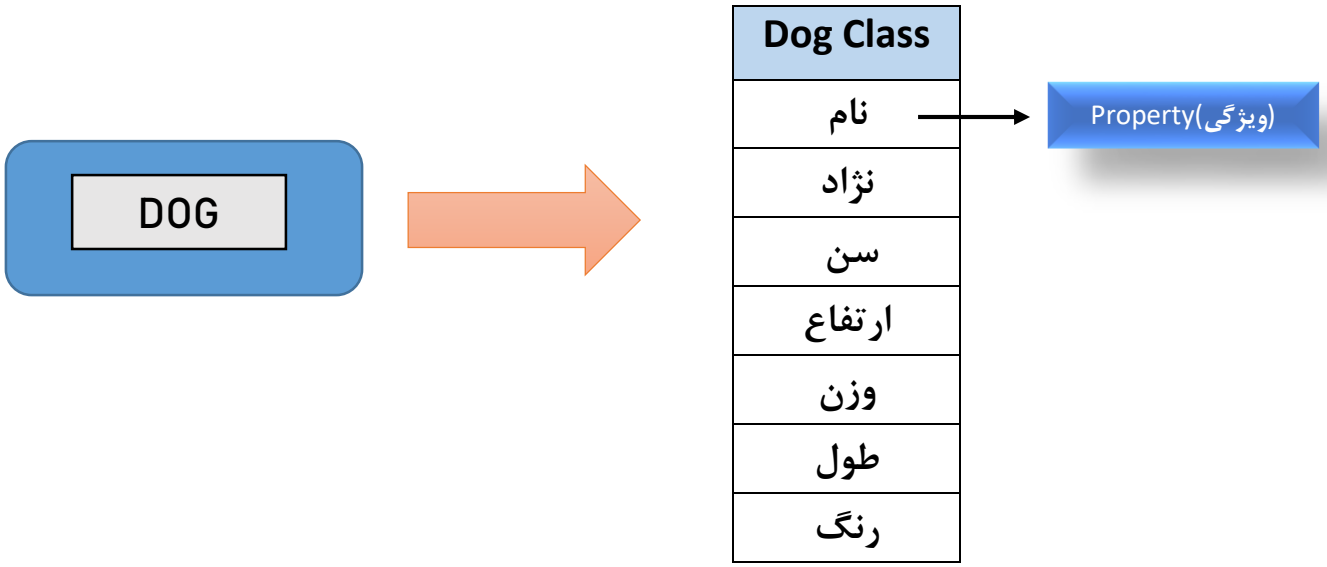
Properties (خصوصیت): رنگ – مدل – سرعت – شرکت سازنده

Behavior (رفتار): حرکت کردن – ترمز – بوق زدن و...

Class (کلاس):

ماهیت شی ما را مشخص می کند

در مثال پایین ما کلاس Dog را داریم این کلاس دارای ویژگی های مختلف از جمله نام، نژاد و... است



در کلاس **Dog** ما 7 ویژگی یا **(property)** را برای سگ عنوان کردیم.

تعداد این ویژگی ها بستگی به نوع برنامه، جزئیات پیاده سازی دارد

مثلا می خواهید برای دیجیکالا سایت طراحی کنید یا برای یک فروشگاه اینترنتی معمولی. بسته به آن شما به ویژگی های بیشتر و یا کمتری نیاز دارید.

متد:

قطعه کدی است که یک بار آن را می نویسیم و می توانیم هر بار آن را صدا بزنیم تا برای ما عملیاتی که برای آن تعیین کردیم را انجام دهد

مثلا فرض کنید برنامه ایی نوشته ایم و می خواهیم هر جا که برنامه خطایی داد یک پیغام به کاربر نمایش دهیم

```
Print(“ we have an error “)
```

ممکن است در قسمت های مختلفی از برنامه نیاز به نمایش پیغام باشد به جای اینکه هر با این خط کد را بنویسیم یک متد تعریف می کنیم و هر جا که نیاز بود آن را فراخوانی می کنیم

از مزایای نوشتن متد می توان به کاهش حجم کد نویسی و کاهش پیچیدگی کد اشاره کرد.

اگر با نحوه نوشتن متد آشنایی ندارید به زبان برنامه نویسی مورد نظرتان رجوع کنید و نحوه نوشتن آن را یاد بگیرید.

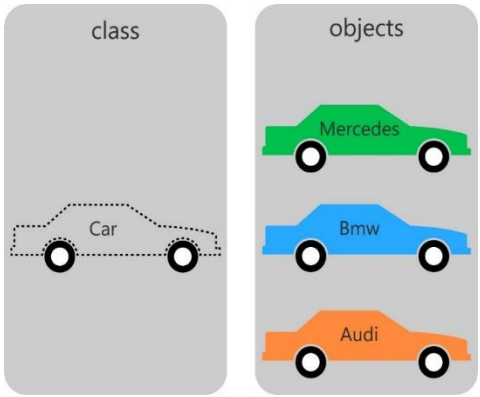
Object (شیء):

نمونه ایی از کلاس است

نام	آلفرد
نژاد	ژرمن شپرد
سن	2 Y
ارتفاع	70 cm
وزن	45 Kg
طول	110 cm
رنگ	مشکی

اگر مثال بالا را در نظر بگیریم می توانیم خصوصیات دیگری مانند سرعت، نوع کارایی، قدرت آرواره، محدوده بویایی، مکان تولد نژاد، محدوده شنوایی، حداکثر عمر، محدوده بینایی،رنگ و... را برای سگ در نظر بگیریم. ولی باز هم اشاره می کنم خصوصیاتی که بیان می کنید می تواند بسته به کسب و کاری که می خواهید برای آن برنامه بنویسید متفاوت باشد.

تمرین: یک شی زنده را نظر بگیرید و خصوصیات و رفتار هایی که می تواند داشته باشد را درون کاغذ بنویسید.



Abstraction (انتزاع):

به مفاهیم انتزاعی که یک طرح یا الگو هستند انتزاع می گویند.

به عنوان مثال حیوان یک الگوی انتزاعی و ببر نمونه ایی از آن الگو است.

ما در زبان های شی گرا از دو مفهوم abstract class و interface

برای پیاده سازی انتزاع استفاده می کنیم.

Abstract Class:

کلاس های ابسترکت قابل نمونه سازی نیستند و باید به ارث برده شوند.

Abstract method:

همچنین ما متد های ابسترکت نیز داریم که نمی توانند بدنه داشته باشند

و باید در کلاس مشتق شده (به ارث برده شده) بدنه آن را بنوسیم.

Interface (واسط):

یک راه دیگر برای پیاده سازی ابسترکشن استفاده از واسط است

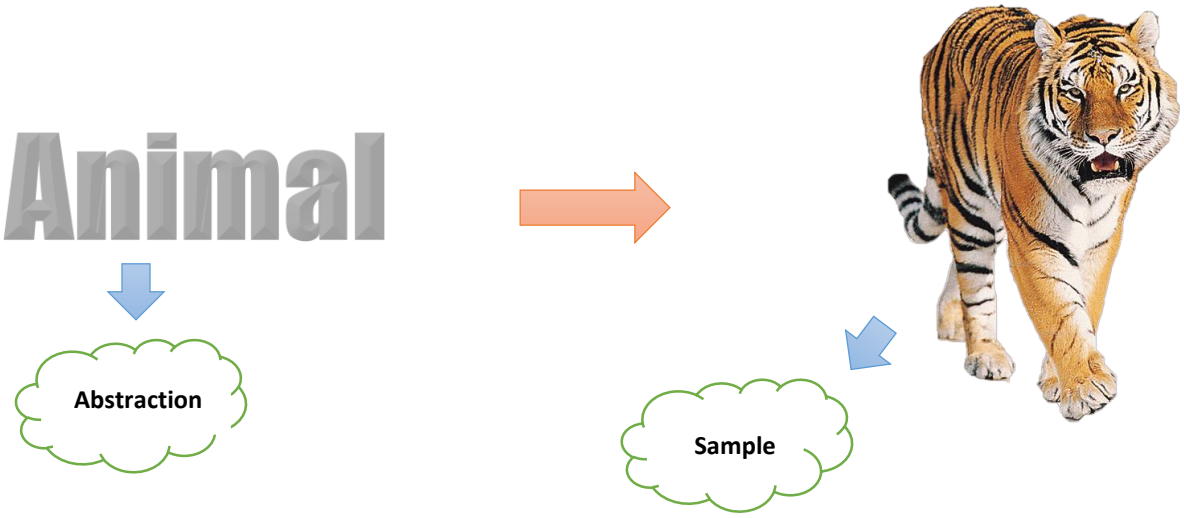
اینترفیس ها نمی توانند پراپرتی داشته باشند.

متد هایشان نمی توانند بدنه داشته باشند

و باید توسط یه کلاس و یا یک واسط دیگر پیاده شوند

ویژگی های بالا در زبان های مختلف متفاوت است و کمی نحوه پیاده سازی آن ها متفاوت است اما قوانین یکی است و در زبان های مختلف تفاوتی ندارد .

مثلا ما در زبان جاوا برای تشخیص متد ساده از متد ابسترکت به صراحت کلمه abstract را در ایجاد متد به کار می بریم



Inheritance (ارث بری):

همانگونه که در دنیای واقعی فرزند از والدین ارث می برد در دنیای برنامه نویسی هم می توان کلاسی یا اینترفیسی را به ارث برد

عملی که در آن یک شی تمام ویژگی ها و رفتارهای شی والد را به ارث می برد.

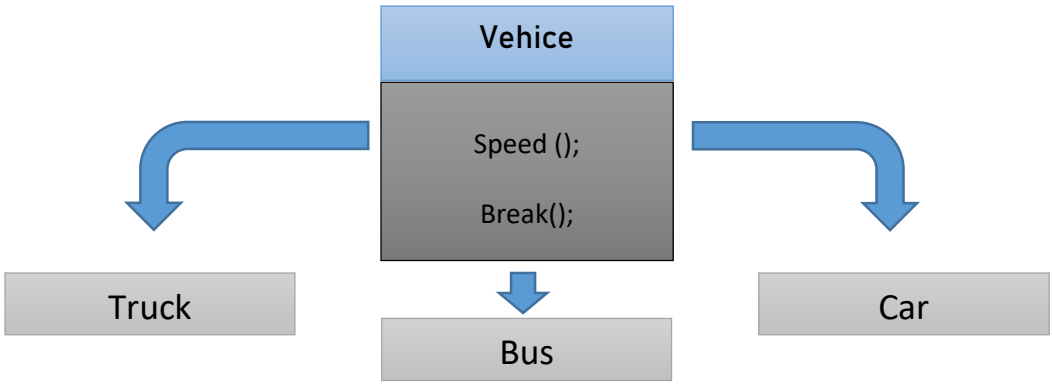
مزایا:

قابلیت استفاده مجدد از کد

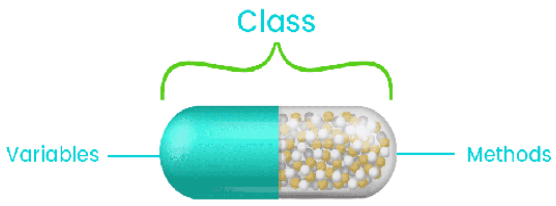
برای MethodOverriding استفاده می شود (در ادامه توضیح می دهم)

مثال:

ما کلاسی با نام Vehicle (وسیله نقلیه) داریم این کلاس دارای 3 زیر کلاس دیگر به نام های Bus (اتوبوس),Car (ماشین),Truck (کامیون) است. می توان متد هایی که کلاس های فرزند (اتوبوس ، ماشین و کامیون) به آن نیاز دارند را در کلاس وسیله نقلیه نوشت و آن را در کلاس های فرزند به ارث برد.



Encapsulation (کپسوله سازی):



اطمینان حاصل می کنیم که

داده های حساس از دید کاربران پنهان هستند.

بگذارید با یک مثال ساده توضیح دهم.

حداکثر میزان سرعتی که در سرعت سنج ماشین بنز گذاشته شده

340km بر ساعت است.

خب اما آیا این به این معنی است که موتور ماشین نمی تواند بیشتر از

340km حرکت کند؟

جواب این سوال بله است. موتور بنز می تواند بیشتر از این سرعت نیز حرکت کند اما ماشین برای بالاتر از این سرعت طراحی نشده و امکان دارد ماشین و یا سرنشینان دچار مشکل شوند پس در اینجا داده ی حساس سرعت اصتلاحا کپسوله شده و کاربر که راننده باشد نمی تواند با سرعت بیشتر از آن حرکت کند.

Max Speed = 360km;

Speed Now =360km;

Gas For the mor speed;

System Message:

You can't go any faster than that

مزایا:

می توانیم یک مقدار را فقط خواندنی و یا فقط نوشتنی تعریف کنیم

می توانیم داده های ورودی را کنترل کنیم

در زبان های مختلف کپسوله سازی به شکل های مختلفی پیاده می شود

مثلا در زبان جاوا ما با متد های Getter و Setter کپسوله سازی را انجام می دهیم

پس از خواندن هر مفهوم به گوگل مراجعه کنید و نحوه پیاده سازی آن بخش را در زبان خود یاد بگیرید

Polymorphism (چند ریختی):

بر خلاف اسم پیچیده آن مفهوم بسیار ساده ایی دارد.

چند ریختی یعنی پیاده سازی یک عمل به چند شکل مختلف.

چند شکلی از دوکلمه یونانی گرفته شده است poly به معنای چند و morphism به معنای شکلی. پس پلی مورفیسم یعنی چند شکلی

دو نوع چند شکلی وجود دارد.

فرض کنید می خواهید به خانه دوستتان بروید چند راه برای رسیدن وجود دارد اما در نهایت شما می خواهید به خانه دوستتان بروید.

ما دو نوع پلی مورفیسم داریم

Runtime Polymorphism (چند شکلی زمان اجرا):

فرایندی است که در آن فراخوانی یک متد نادیده گرفته می شود و در زمان اجرا به آن پرداخته می شود .

ما با استفاده از overriding چند شکلی زمان اجرا را پیاده می کنیم.

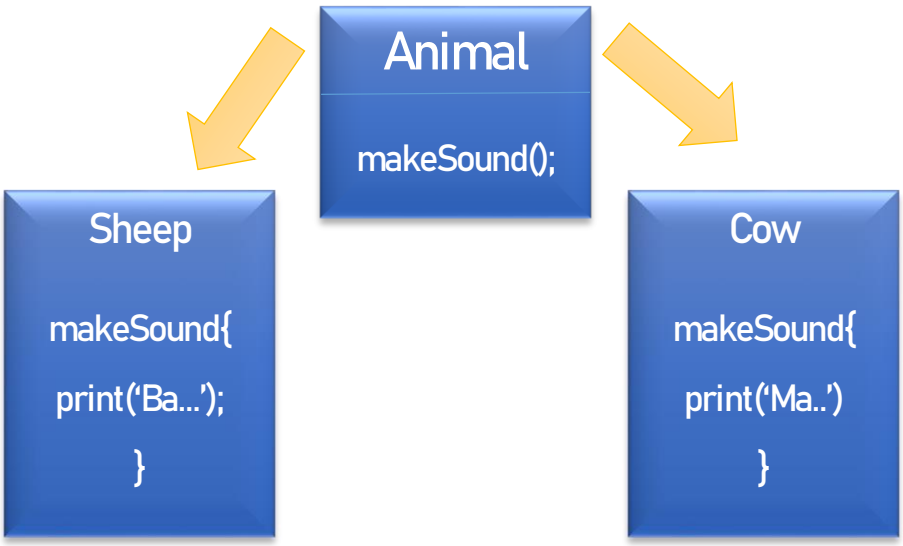
به مثال زیر توجه کنید.

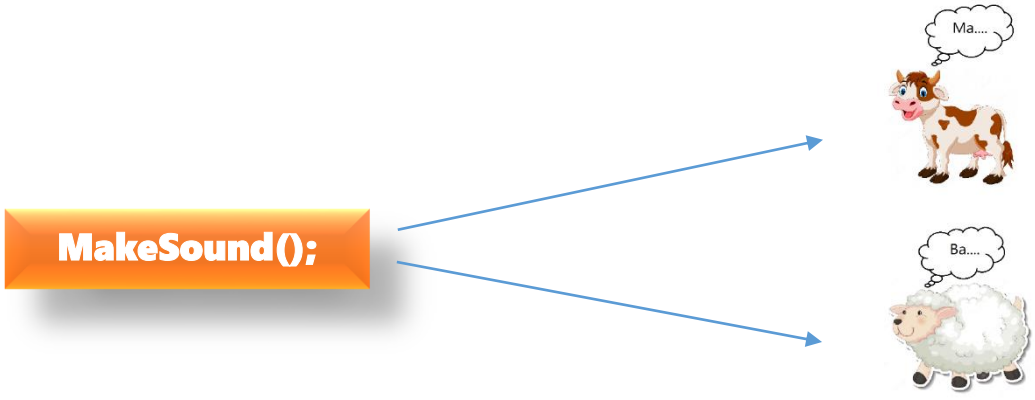
فرض کنید یک کلاس انتزاعی با نام **animal** داریم و این کلاس متدی با نام **()makeSound** دارد. ما بدنه این کلاس را نمی نویسم و قصد داریم در کلاس های مشتق شده از آن بدنه آن را بنوسیم.

حال 2 کلاس دیگر با نام های **cow** و **sheep** داریم و حال برای هر کدام از آن ها بدنه ای جداگانه می نویسیم زیرا شکل تولید صدای گوسفند با گاو فرق می کند.

به بازنویسی متد اصتلاحا **override** می گوییم.

ما متد **makeSound()** در کلاس **animal** را در زیر کلاس های آن بازنویسی می کنیم





Compile Polymorphism (چند شکلی زمان کامپایل):

فرایندی است که در زمان کامپایل بررسی می شود.

کامپایلر بررسی می کند تا بتواند تشخیص دهد کدام متد را در زمان ساخت باید فراخوانی کند.

از آن با عنوان چند شکلی استاتیک یا اضافه بار نیز یاد می کنند.

به مثال زیر توجه کنید.

در قوانین زبان های برنامه نویسی ما نمی توانیم دو متد با یک نام در یک کلاس داشته باشیم مگر اینکه پارامتر های ورودی آنها متفاوت باشد.

در کلاس رو به رو ما یک کلاس با نام **calculator** داریم که دو متد برای جمع دارد

یکی از متد ها جمع دو عدد ورودی را

بر می گرداند و متد دیگر جمع نام و فامیلی را بر می گرداند.

به این کار **Method Overloading** یا سربارگذاری متد می گویند.

```
Class Calculator
{
  Int Sum( int num1,int num 2)
  {
  return num1 + num2;
  }
  String Sum (str Name,str family)
  {
  return name + family;
  }

}
```

نحوه انجام کار بسته به زبان برنامه نویسی شما ممکن است متغیر باشد زیرا هر زبان روش مخصوص به خود برای انجام کار را دارد پیشنهاد می کن بعد از خواندن کتاب به نحوه پیاده سازی این مفاهیم در زبان خود بپردازید.

اگر دانش در ثریاّ باشد، بی گمان مردانی از فارس بدان دست یابند

حضرت محمد (صلی اللّه علیه و آله)