مقدمه ای بر شئ گرایی

انتزاع (Abstraction)

- پیچیدگی مسائلی که شما قادر به حل آن ها هستید ، ارتباط مستقیم با نوع و کیفیت انتزاع (چکیده) زبان برنامه نویسی شما دارد.
- برنامه نویس همیشه مجبور است بین **مدل ماشین** (فضای راهکار) و **مدل مسئله** (فضای مشکل) ارتباط برقرار کند.

روش برنامه نویسی شئ گرایی

- روش شئ گرایی ، ابزاری است برای برنامه نویس تا بتواند عناصر مورد نظرش را در فضای مشکل تعریف کند.
- به عناصری که در فضای مشکل تعریف میشوند و در فضای راهکار نمود پیدا میکنند ، شئ یا Object اطلاق میشود.

روش برنامه نویسی شئ گرایی

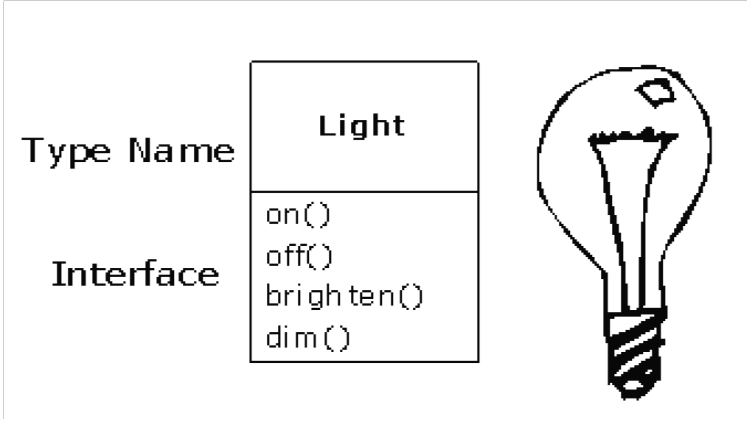
- این روش به شما اجازه میدهد که مسئله ی مورد نظر را در قالب مشکل تعریف کنید به جای اینکه آن را در شرایط راهکار ببینید.
- هر شئ شبیه به یک نرم افزار کوچک میماند که دارای شرایط (حالات) و عملکرد (Operations) خاص خود میباشد و شما میتوانید از آنها استفاده نمایید.

خصوصیات زبانهای شئ گرایی

- هر چیزی یک شئ است
- یک برنامه شامل تعدادی اشیاء است که در کنار هم تعریف شده و با هم دیگر از طریق ارسال پیغام ارتباط برقرار میکنند.
- هر شئ یک نوع (type) دارد.
- همه ی اشیائی که از یک نوع هستند میتواند پیغام های مشابه دریافت کنند

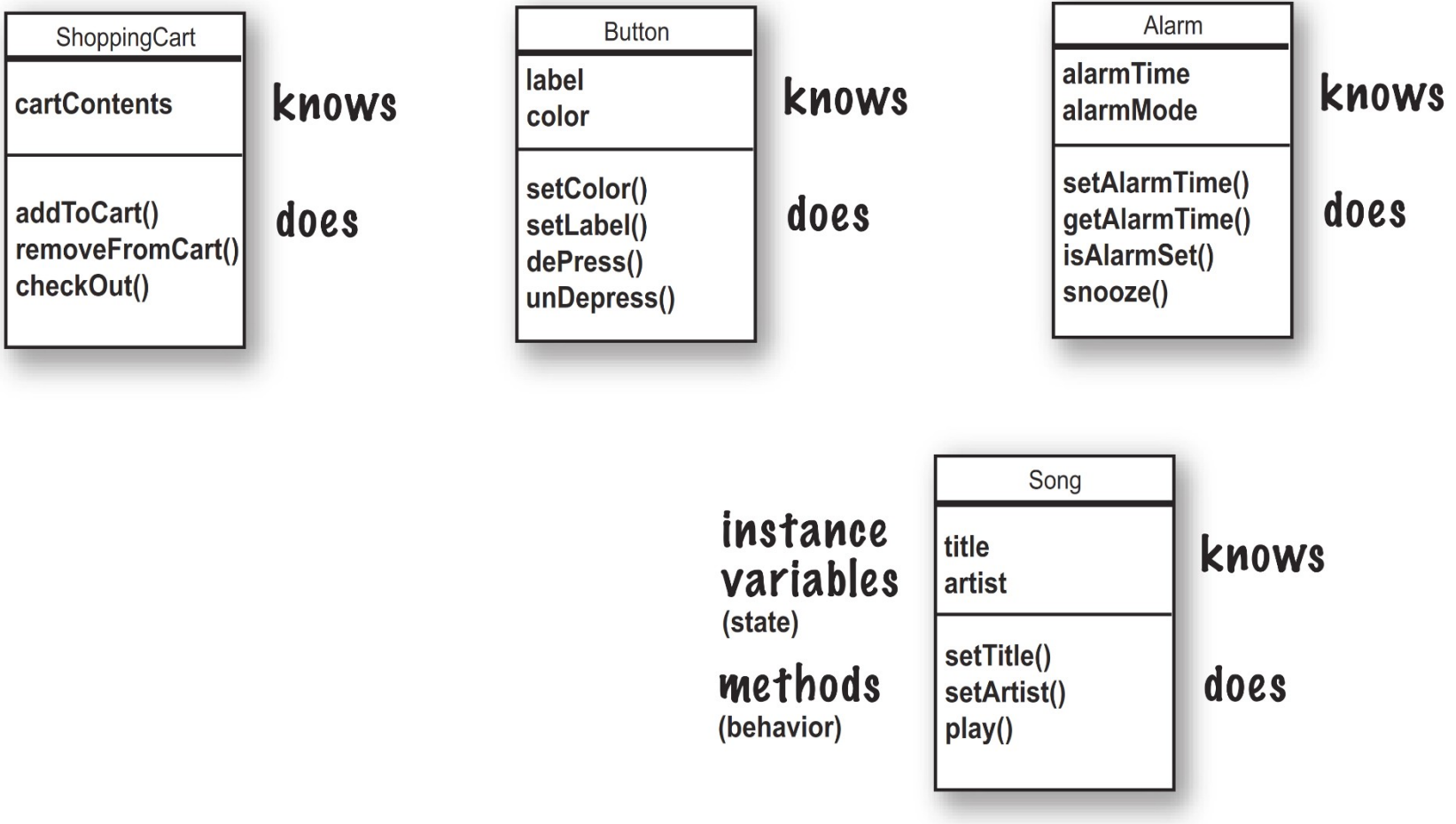
هر شئ یک نوع دارد

- اشیائی که از یک جنس یکتا هستند (به جز حالاتی که در طی اجرای برنامه دارند) در یک گروه خاص به نام «کلاس» تقسیم بندی میشوند و کلید واژه ی class از همینجا گرفته شده است.
- برنامه نویس برای رفع مسائل خود ، میتواند به جای اینکه از کلاس های موجود استفاده کند کلاس های جدیدی تعریف کند




```
Light lt = new Light();
lt.on();
```

هر شئ یک نوع دارد



یک شئ یک سرویس است

- بهترین روش برای اینکه تشخیص دهید چگونه یک برنامه طراحی کنید این است که به اشیاء به شکل یک «سرویس دهنده» نگاه کنید.
- همچنین این نوع دیدگاه کمک میکند انسجام بین اشیاء را از هم تمیز دهید.
- انسجام بالا یا High Cohesion یکی از اجزای مهم کیفیت نرم افزار میباشد.

مخفی سازی پیاده سازی (کپسوله سازی -Encapsulation)

- برنامه نویسان را میتوان به دوگروه تقسیم کرد ، کلاس نویس ها و برنامه نویسان client
- مخفی سازی پیاده سازی در اینجا اهمیت پیدا میکند
  - برای اینکه به برنامه نویسان client اجازه ندهیم به بخش های ضروری دسترسی داشته باشند
  - برای اینکه به کلاس نویس ها اجازه بدهیم بدون نگرانی از اینکه پیاده سازی داخلی شان دستخوش تغییر شود ، کلاس ها را توسعه دهند.

مخفی سازی پیاده سازی (کپسوله سازی -Encapsulation)

- در کد نویسی از چند کلید واژه برای ایجاد مرزبندی استفاده میکنیم:

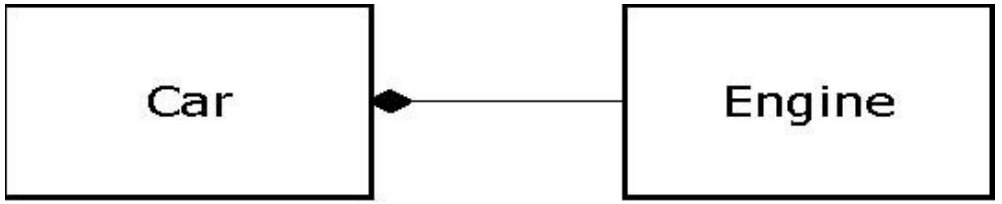
- Private
- Protected
- Public
- Default (Package access)

استفاده مجدد از کلاس ها

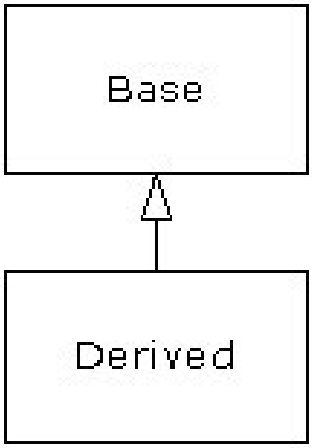
- استفاده ی مجدد از کلاس ها یکی از بهترین قابلیت های برنامه های شئ گرا است.
- یکی از ساده ترین راه های استفاده مجدد از کلاس ها این است که به راحتی یک شئ را در یک کلاس استفاده کنید یا اینکه آن شئ را داخل یک کلاس قرار دهید.

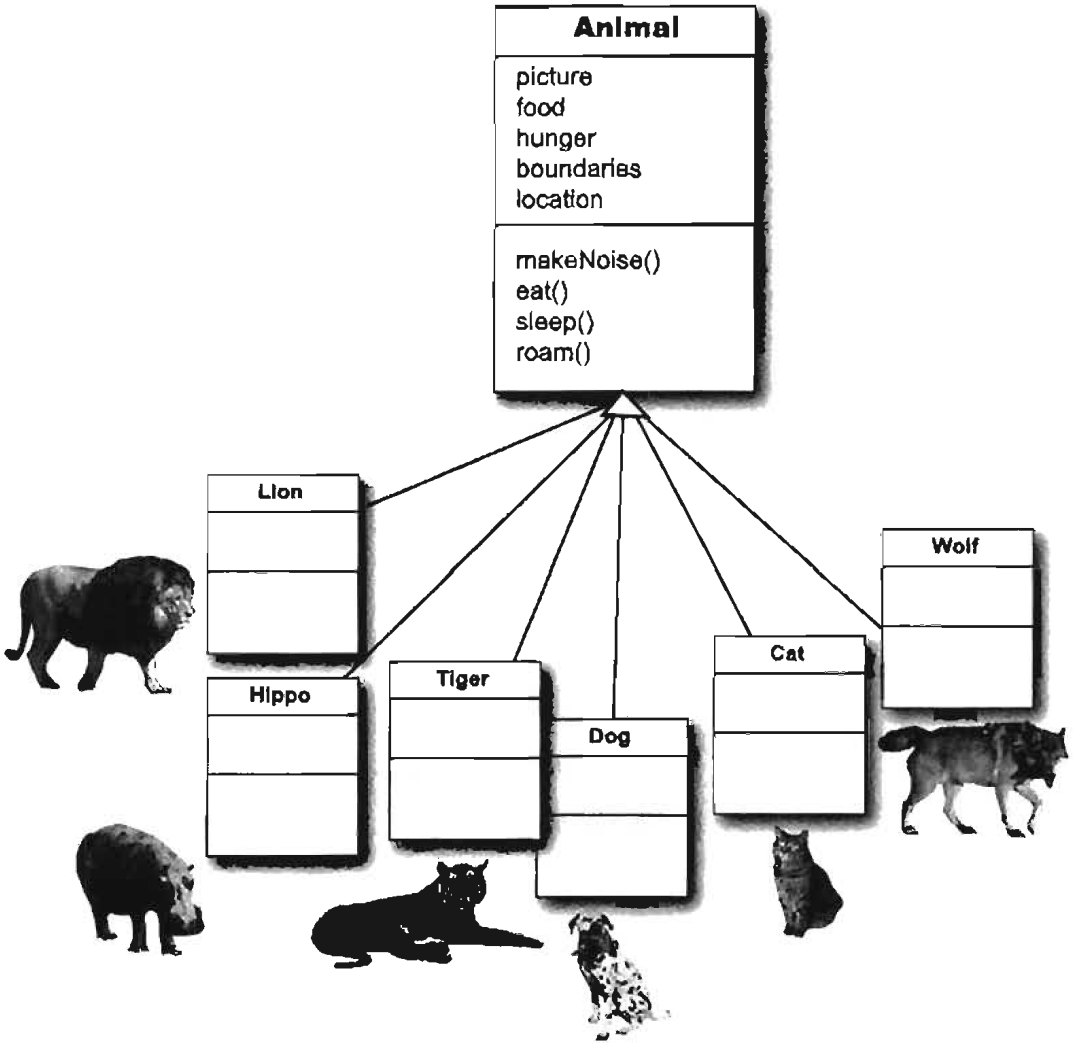
استفاده مجدد از کلاس ها

- به این عمل «ساخت یک **عضو** از شئ» نامیده میشود، برای اینکه شما یک کلاس را در یک کلاس جدید ترکیب میکنید. به این مفهوم Composition یا Aggregation میگویند.
- به این نوع رابطه ، رابطه ی has-a میگویند:
  - A Car Has an Engine



ارث بری

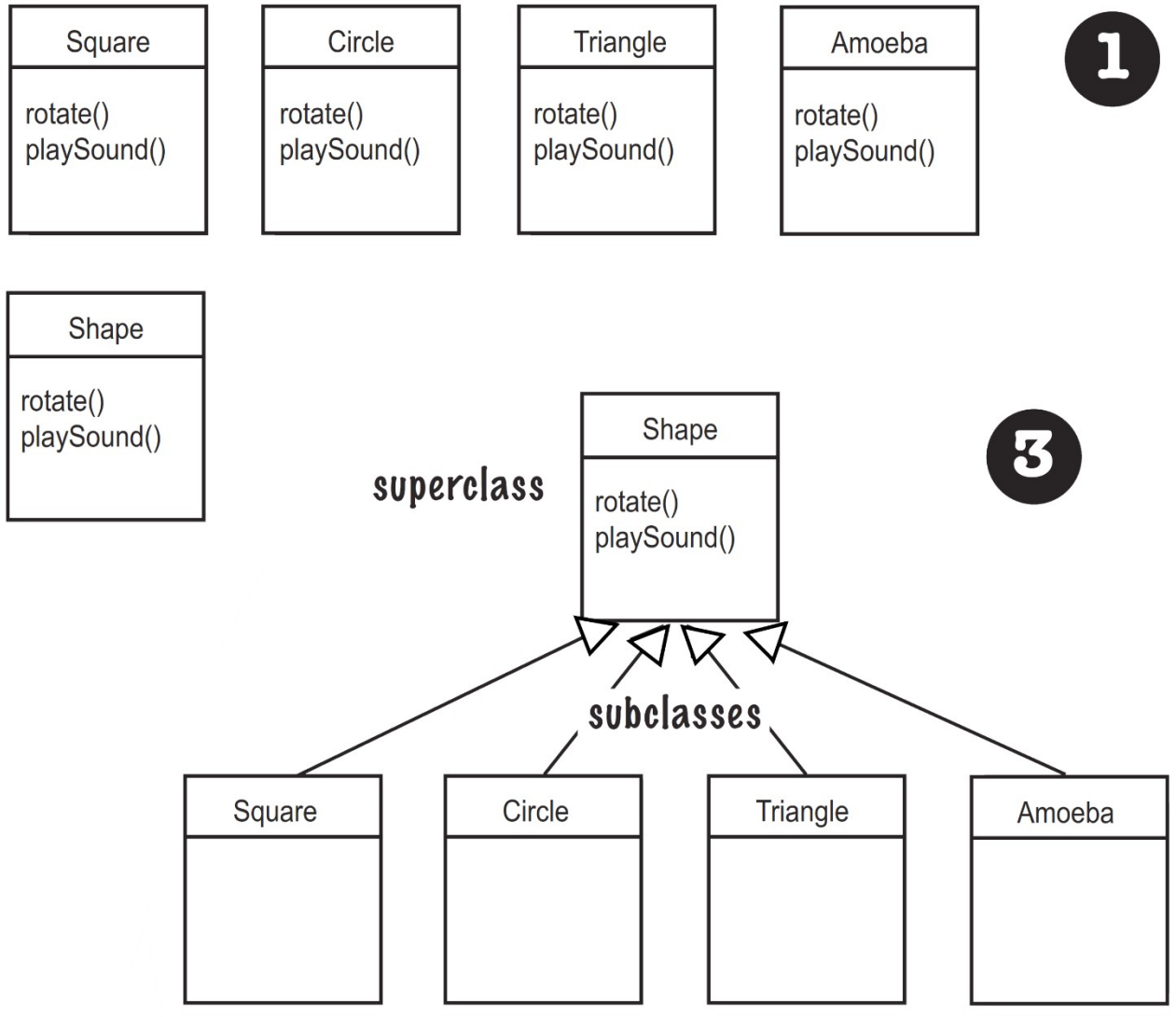

Base
Derived


Animal
picture
food
hunger
boundaries
location
makeNoise()
eat()
sleep()
roam()
Lion
Hippo
Tiger
Dog
Cat
Wolf

ارث بری

- مواردی پیش می آید که شما قصد تعریف کلاسی دارید که اتفاقاً به کلاس دیگری (از لحاظ رفتار) شبیه است. خیلی بهتر است که در این موارد از کلاس موجود ،کلاس جدید را تکثیر کنیم و بعد رفتار های آن را تغییر یا رفتار جدیدی به آن اضافه کنیم.
- اگر کلاس اصلی (کلاس پدر ، parent class، superclass، base class) تغییر کند ، کلاس تکثیر یافته (کلاس فرزند ، inheritted، derived class ،child class، sublcass، class) نیز تغییر خواهد کرد.

چگونه ارث بری را تشخیص دهیم؟



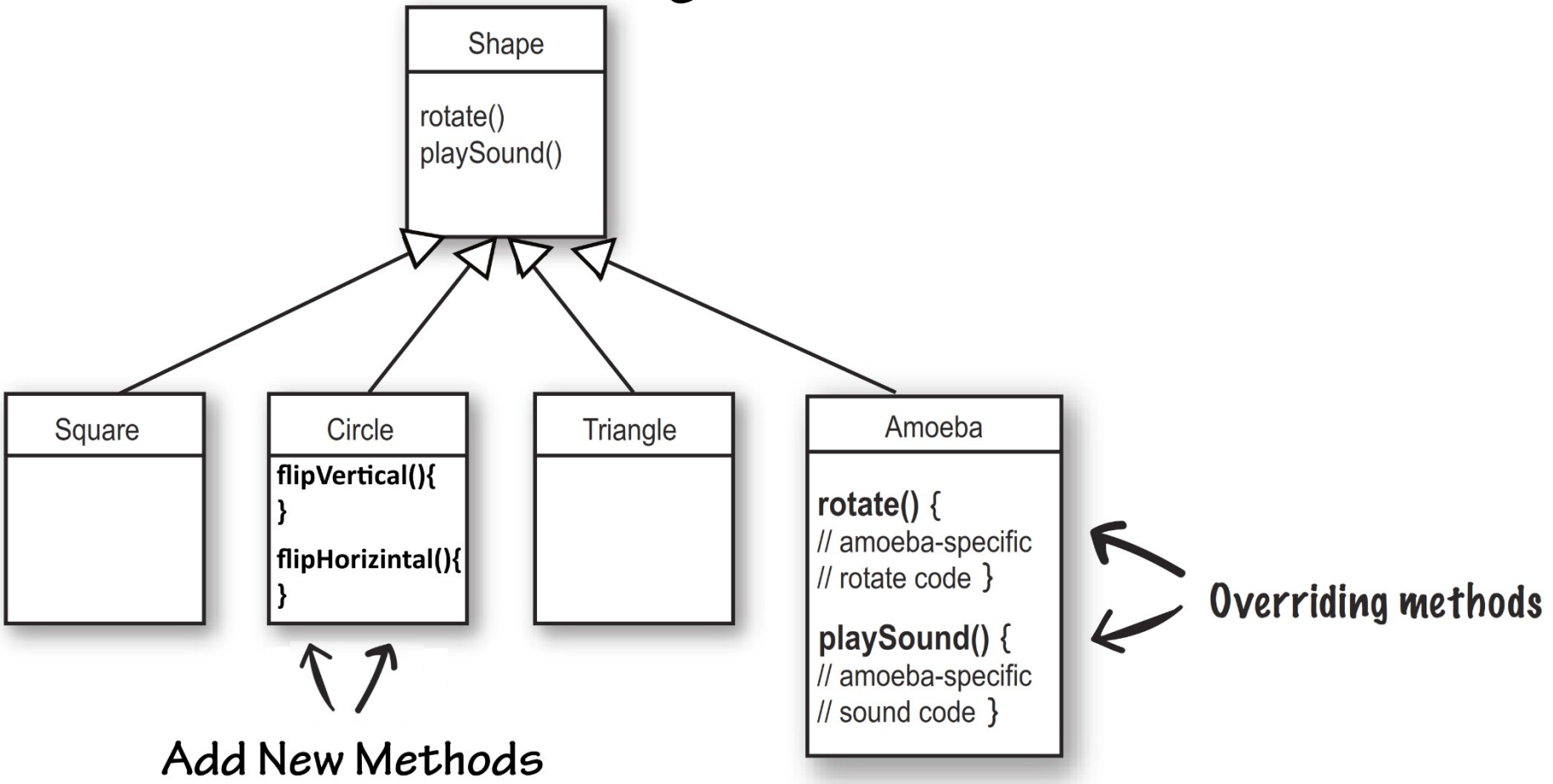
ارث بری

- در وراثت ، کلاس فرزند ، از طریق دو راه میتواند تفاوت ایجاد کند :

1) اضافه کردن متود

2) تغییر دادن پیاده سازی یک متود از کلاس پدر overriding

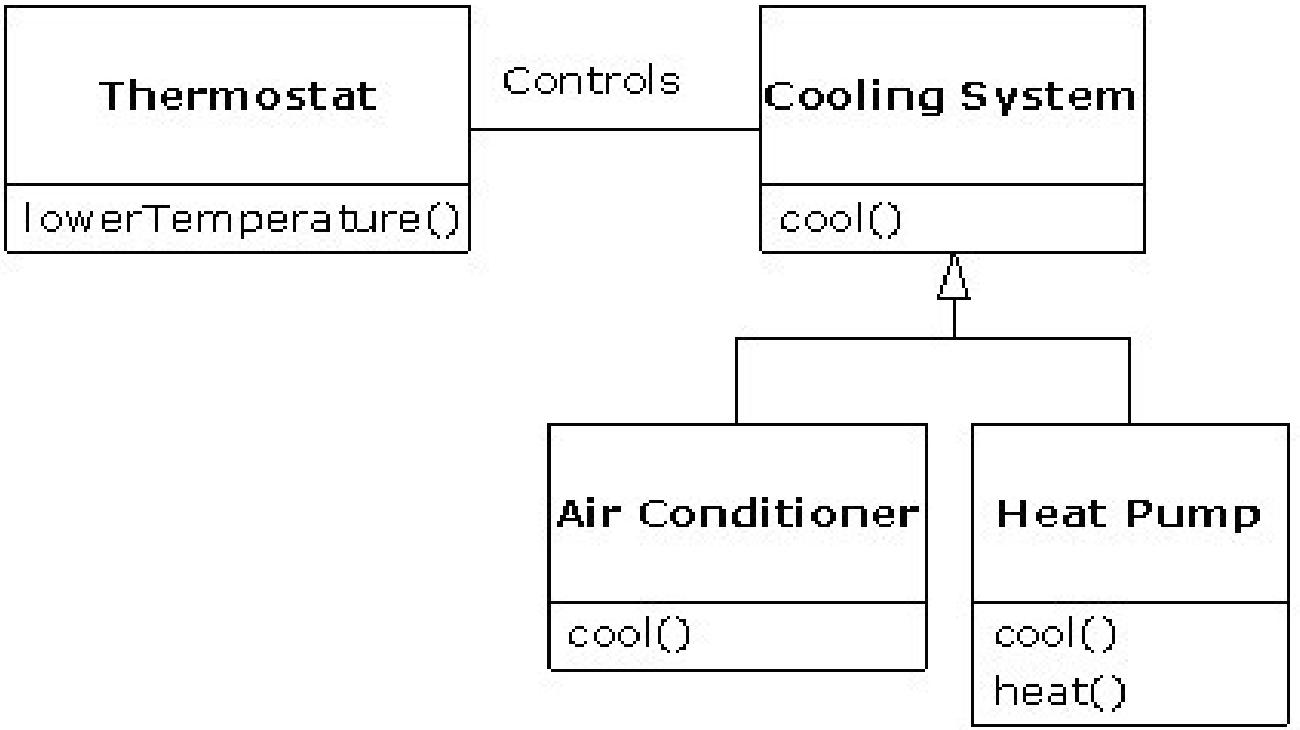
ارث بری


Shape
rotate()
playSound()
Square
Circle
flipVertical(){
}
flipHorizintal(){
}
Triangle
Amoeba
rotate() {
// amoeba-specific
// rotate code }
playSound() {
// amoeba-specific
// sound code }
Overriding methods
Add New Methods

رابطه ی is-a در مقابل is-like-a

- وقتی کلاس فرزند دقیقاً مثل کلاس پدر باشد به آن جانشینی مطلق یا pure subtitution میگویند . و به رابطه ی بین آن دو is-a اطلاق میشود . (A Triangle is a shape)
- گاهی اوقات کلاس فرزند تغییرات جدیدی به کلاس پدر اضافه میکند. با اینکه رابطه کماکان پدر-فرزندی باقی می ماند اما این نوع جانشینی ۱۰۰٪ نیست . به این نوع رابطه is like a میگویند. (A circle is like a shape)

رابطه ی is-a در مقابل is-like-a



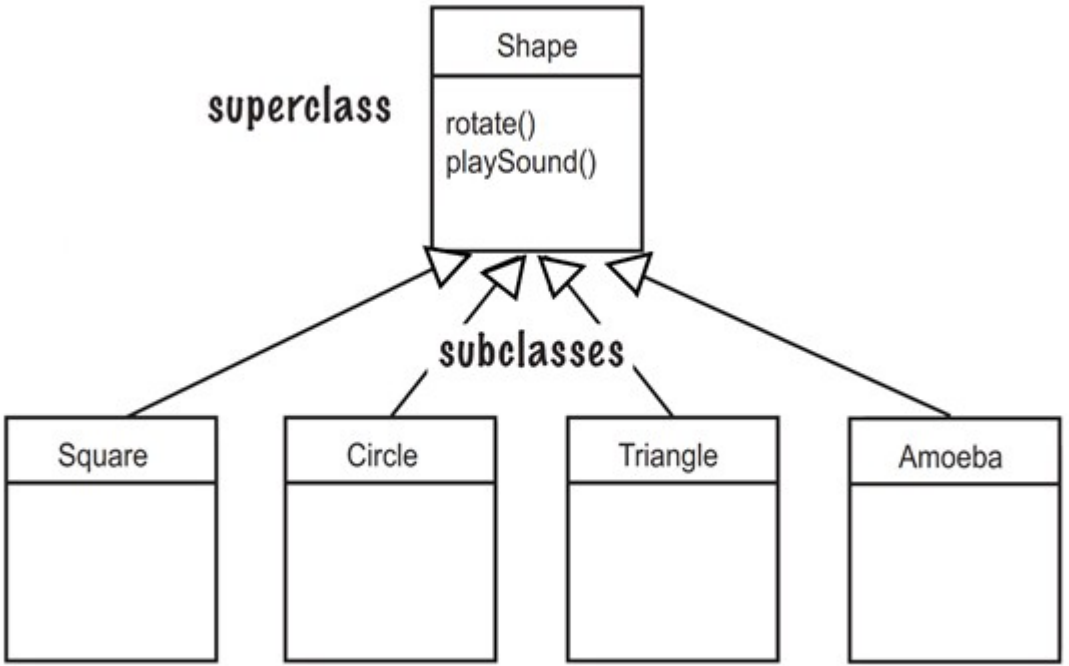
چند ریختگی Polymorphism

- در ادامه ی مبحث ارث بری ، گهگاه نیاز داریم با یک کلاس به شکلی که وجود دارد برخورد نکنیم و به جای آن به شکل کلاس پدر برخورد صورت گیرد. این ویژگی اجازه میدهد کدی بنویسید که به یک نوع خاص بستگی ندارد.
- این نوع کد با اضافه شدن کلاس های جدید متأثر نمیشود.
- اما یک مشکل وجود دارد ، اگر با یک کلاس به شکل کلاس پدرش برخورد شود ،اگر قرار باشد یک متود صدا زده شود کامپایلر از کجا باید تشخیص دهد چه قطعه کدی صدا زده شود؟

Polymorphism چند ریختگی

```
void doSomething(Shape shape) {
    shape.rotate();
  // ...
    shape.playSound();
}

Circle circle = new Circle();
Triangle triangle = new Triangle();
Line line= new Line();
doSomething(circle);
doSomething(triangle);
doSomething(line);
```

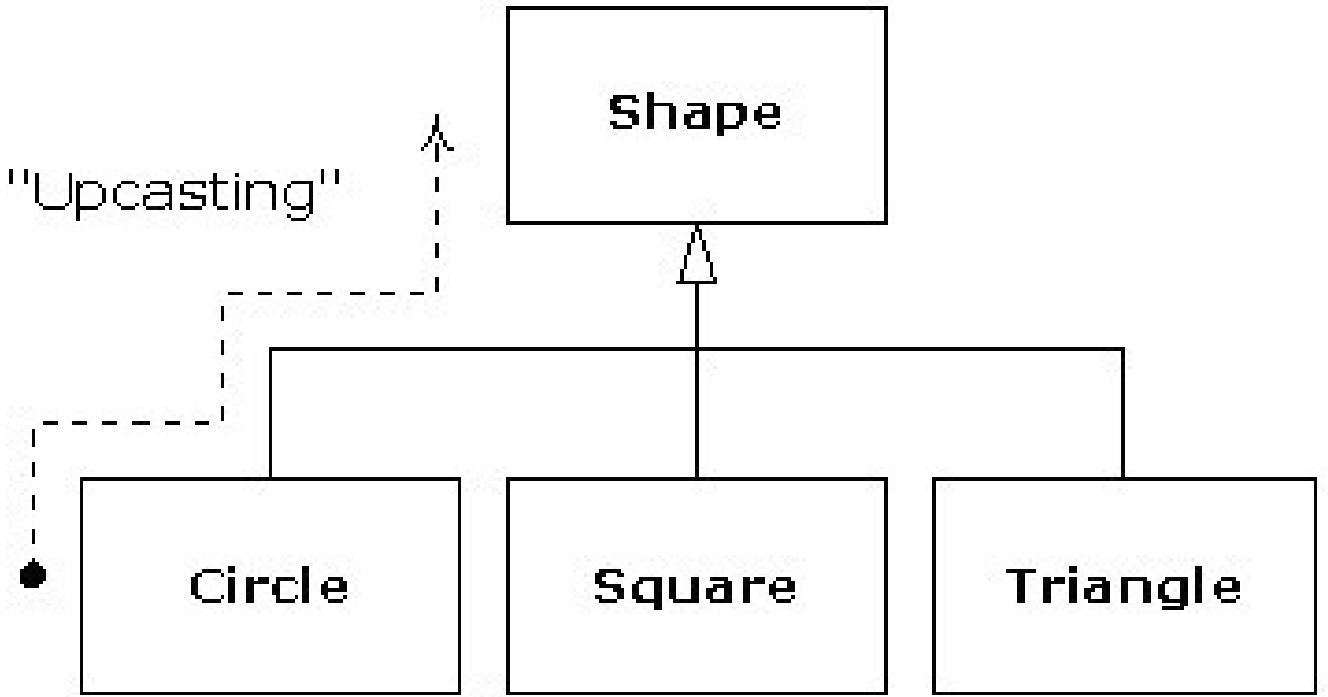


چند ریختگی Polymorphism

- پاسخ سؤال قبل این است که در زبانهای غیر شئ گرا کامپایلر از مفهوم اتصال بلادرنگ (early binding) استفاده میکند و به همین خاطر در آنها چند ریختگی معنا ندارد.
- در زبانهای شئ گرا ، زبان ها از مفهومی به نام اتصال تأخیری (late binding) استفاده میکنیم . در این روش نحوه ی چگونگی ارسال پیغام بین اشیاء ، زمان اجرا تحلیل و تشخیص داده میشود

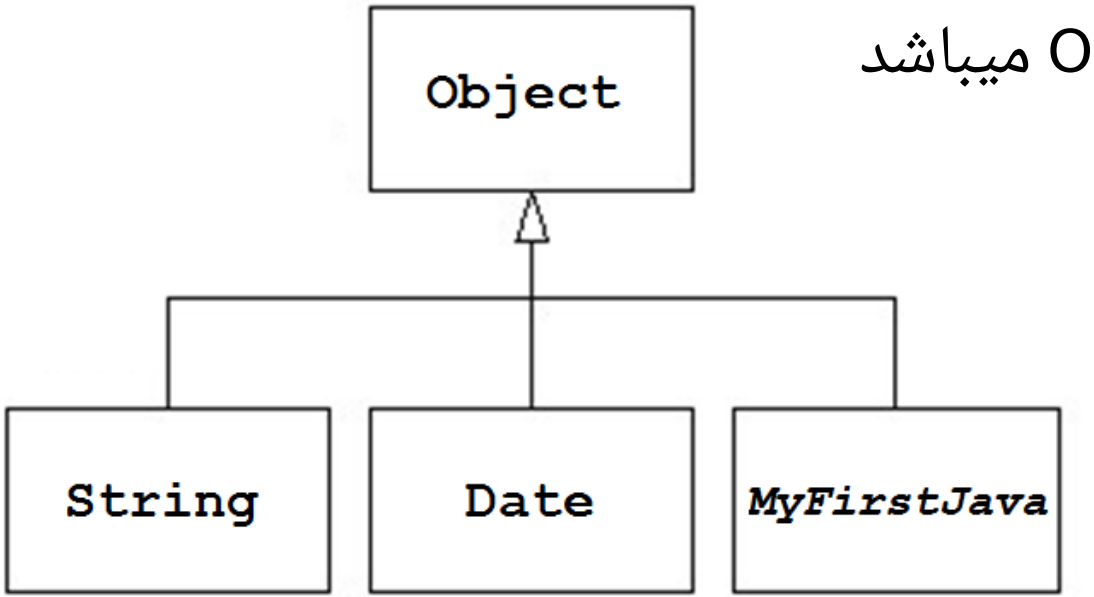
چند ریختگی Polymorphism

- به روندی با کلاس های فرزند به شکل کلاس پدر رفتار میشود upcasting میگوند



ساختار تک ریشه ای

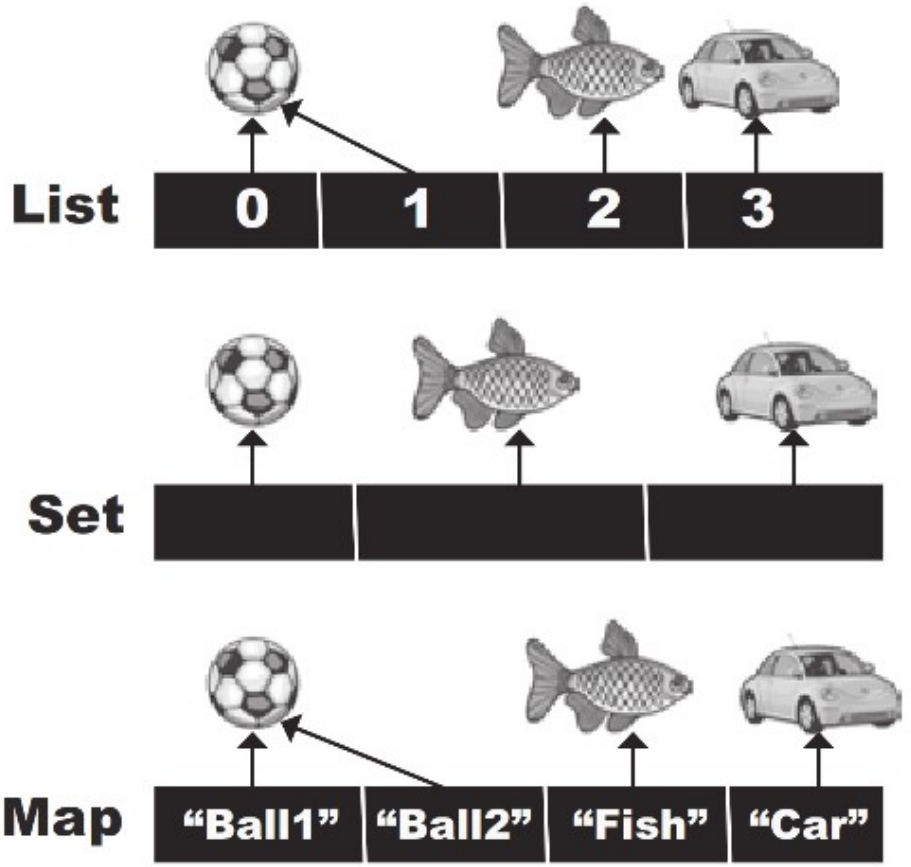
- در ساختار تک ریشه ای ، همه ی اشیاء همواره از کلاس واحد ارث بری کرده اند ، با این حساب همگی آنها رفتاری مشابه خواهند داشت.
- نام این کلاس واحد Object میباشد



نگه دارنده ها (Containers)

- عموماً ، ممکن است شما از اینکه چه تعداد شئ نیاز دارید ،اطلاع نداشته باشید.
- شما میتوانید این مشکل را توسط آرایه ها (array) حل کنید که در اکثر زبانها موجود است . اما آرایه ها ثابت هستند به همین خاطر به سراغ Collection ها میرویم.

نگه دارنده ها (Containers)



Generics – Parameterized types

- جهت استفاده از یک container به راحتی در آن اشیا را اضافه میکنیم . اما از آنجایی که container ها فقط مقادیر Object ها را میشناسد ، تمامی اشیاء به Object، Upcast خواهند شد و ماهیت اصلی شان را از دست میدهند.
- در اینجا برای بازگرداندن یک شیء به ماهیت اصلی اش باید از downcast استفاده کنیم.

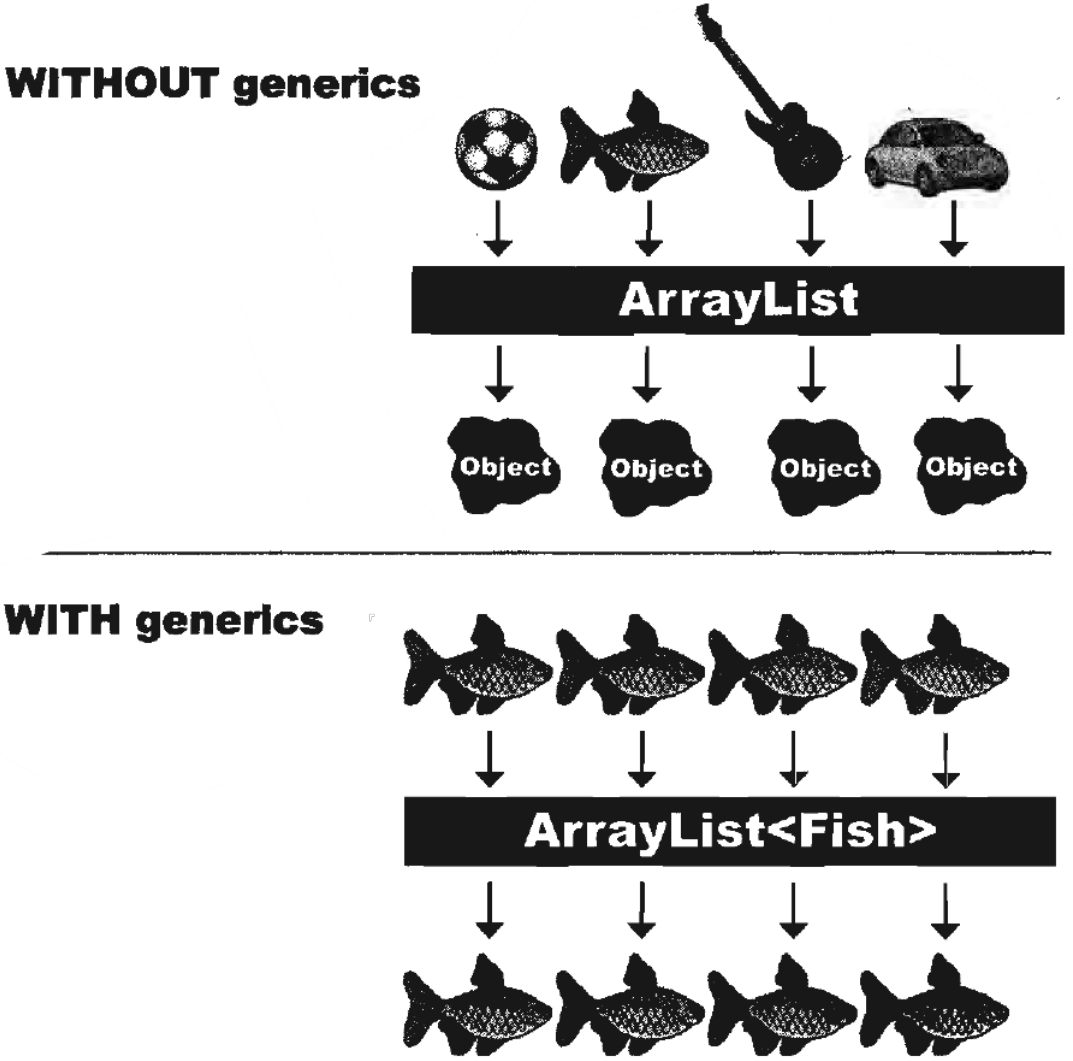
Generics – Parameterized types

- این عملیات هزینه بر است ، لذا برای رفع این مشکل از parameterized type ها استفاده میکنیم.
- Parameterized type کلاسی است که کامپایلر میتواند خودکار برای استفاده در یک نوع خاص بکار گیرد .

```
ArrayList<Shape> shapes = new ArrayList<Shape>();
```

Generics –

Parameterized types



چرخه ی حیات اشیا

- زمانی که کارمان با یک شیء تمام میشود نیاز داریم که آن را از حافظه پاک کنیم
- در یک برنامه نویسی ساده ما یک شئ ایجاد میکنیم ، از آن استفاده میکنیم و در نهایت آن را پاک میکنیم. سؤال مهم اینجاست که چه زمانی برای حذف یک شیء مناسب است ؟
- راهکار ارائه شده این است که همه ی اشیاء را در محفظه ای به نام heap بسازیم . در این روش تا زمان اجرا نیازی نداریم که از تعداد اشیا اطلاع داشته باشیم

چرخه ی حیات اشیا

- همیشه برای ایجاد یک شئ از کلید واژه ی new استفاده میکنیم
- Java خصوصیتی دارد که به آن Garbage Collector میگویند
- Garbage Collectorبرنامه ای است که خودکار اشیائی که دیگر به آنها احتیاجی نیست ، از حافظهheap حذف میکند

مدیریت استثنائات (Exception Handling)

- استثنائات به خطا هایی اطلاق میشود که از سمت زبان برنامه نویسی مستقیماً گزارش میشود. در واقع یک استثناء خود یک شی است که از طرف بخشی که خطا رخ داده به سمتی که خطاها را مدیریت(caught) می کند ، پرتاب (thrown) میشود.
- اگر کدی را مینویسید که از متودی استفاده کند که از قبل احتمال پرتاب استثنا را گزارش داده است ، باید آن کدتان را برای مدیریت استثنا مجهز کنید وگرنه در زمان کامپایل با خطا مواجه میشوید.

مدیریت استثنائات

(Exception Handling)

برنامه نویسی همزمانی

- مفهوم اساسی برنامه نویسی کامپیوتر این ایده بوده است که بتوان چندین عملیات را در در یک واحد زمانی انجام داد. خیلی از برنامه های طراحی شده احتیاج دارند بتوانند در یک زمان خاص عملیاتی را رها کرده و به عملیات جدیدی بپردازند و مجدداً به عملیات اصلی بازگردند یا هر دو را همزمان اجرا کنند.
- در این روش کلاس های جدیدی در قالب وظایف جدیدی ساخته میشوند که در واقع مسائل مختلف را پارتیشن کنند.
- در این برنامه ها به بخش هایی که جداگانه در حال اجرا هستند Thread و به مفهوم کلی آن همزمانی یا concurrency میگویند