به نام خدا

آموزش سی و دوم

اهداف آموزشی این قسمت عبارتند از:

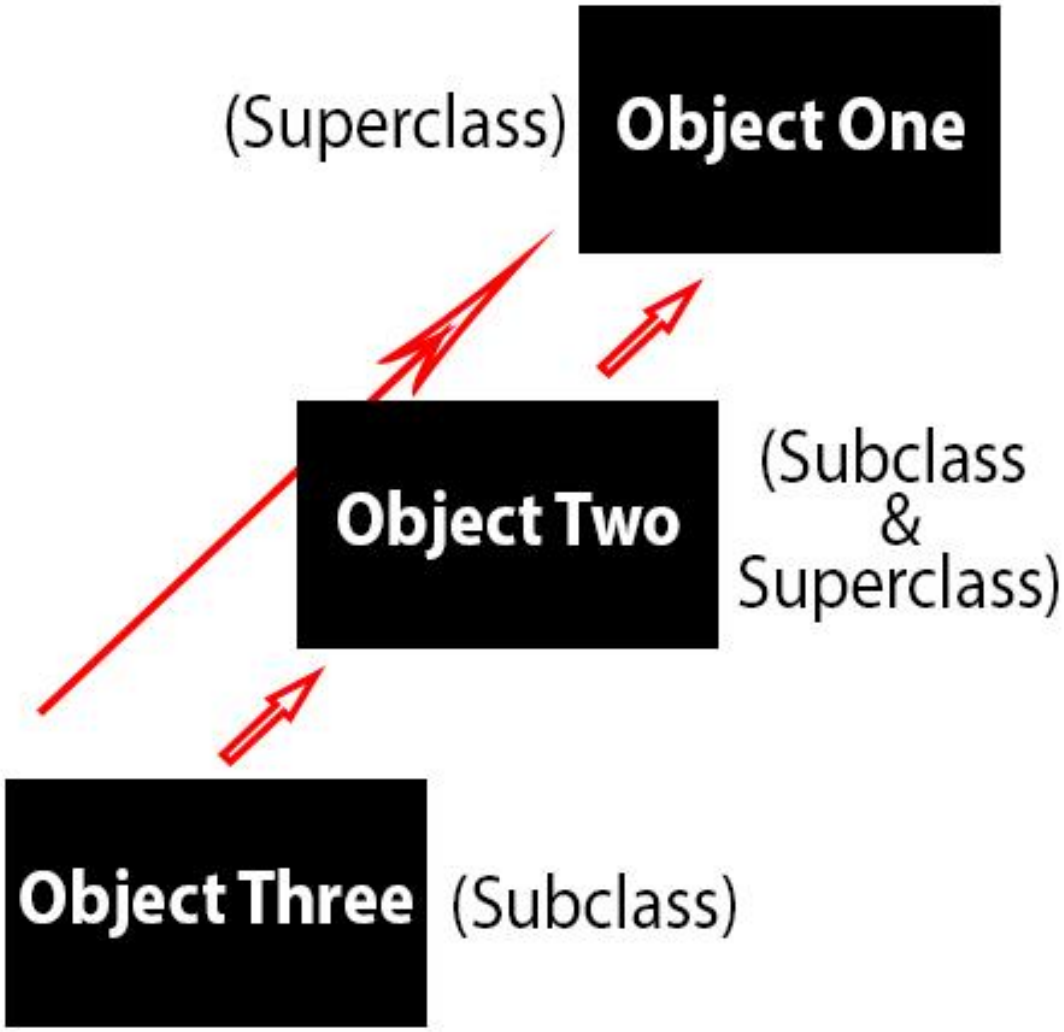
۱. معرفی مفهوم وراثت در زبان جاوا
۲. آشنایی با وراثت سلسله مراتبی
۳. آشنایی با Superclass و Subclass
۴. آشنایی با مفهوم Override در زبان جاوا

در آموزش قسمت سی و یکم توانستیم با موفقیت اولین شیئ خود را در زبان برنامه نویسی شیئ گرای جاوا ایجاد کنیم. حال در این قسمت قصد داریم تا Inheritance یا **به ارث بردن** در زبان برنامه نویسی جاوا را مورد بررسی قرار دهیم.

در حقیقت در برنامه نویسی گاهی برنامه نویسان مجبور هستند Object های متفاوتی ایجاد کنند. در همین حین ممکن است برخی از Object ها خصوصیاتشان مشابه برخی دیگر Object ها باشد. حال اگر برنامه نویس بخواهد تک تک این Object ها را ایجاد کند و خصوصیات یکسانشان را نیز برای هر کدام تعریف کند، اگر در حین نوشتن پروژه بخواهد تا تغییری در برخی خصوصیات Object ها اعمال کند، مجبور است تا کلیه Object ها را بازنویسی کند و در همین جا است که یکی از بهترین ویژگی های زبان برنامه نویسی جاوا که یک زبان برنامه نویسی شئ گرا است را می تواند مورد استفاده قرار داده تا دیگر مجبور نباشد در صورت بروز یک تغییر، کل خصوصیات و یا رفتارهای Object های ایجاد شده را بازنویسی کند. در حقیقت این ویژگی Inheritance یا **"به ارث بردن"** نام دارد.

به طور خلاصه اگر برنامه نویس بخواهد یک Object ایجاد کند که تمامی و یا برخی از ویژگی های یک Object دیگر را داشته باشد می تواند از قابلیت ارث بری در زبان برنامه نویسی جاوا استفاده کند. در ضمن اگر برنامه نویس بخواهد تا یک Object کلیه ی ویژگی های Object دیگری را به ارث برد به جز یک ویژگی که می بایست تغییر کند، این کار از طریق وراثت باز هم

امکان پذیر است. از سوی دیگر وراثت سلسله مراتبی نیز امکان پذیر است. فرض کنیم که ما سه Object داریم به طوریکه Object شماره دو از Object شماره یک ارث بری می کند و Object شماره سه از Object شماره دو. در واقع Object شماره سه نه تنها از ویژگی های Object شماره دو برخوردار خواهد بود، بلکه از آنجا که Object شماره دو از Object شماره یک ارث بری می کند، Object شماره سه ویژگی های Object شماره یک را نیز در بر خواهد داشت. در این مثال Object شماره یک به عنوان Superclass یا **"کلاس اصلی"** تلقی می شود و Object های شماره دو و سه به عنوان Subclass یا **"کلاس زیرشاخه"** تلقی خواهند شد. جالب است بدانیم که در مثال فوق اگرچه که Object شماره دو برای Object شماره یک به عنوان یک Subclass تلقی می شود اما این در حالی است که برای Object شماره سه به عنوان یک Superclass مد نظر قرار داده می شود. این رابطه به خوبی در تصویر زیر قابل مشاهده است:



حال برای آنکه درک بهتری از وارثت در زبان برنامه نویسی جاوا داشته باشیم مثالی از دنیای واقعی زده و سپس آن را به یک پروژه شیئ گرایی تبدیل می کنیم. پدر بزرگ خود را در نظر بگیریم که مثلاً دارای یکسری خصوصیات از این قرار است:

پدر بزرگ		
ردیف	عنوان	مقدار
۱	قد	کوتاه
۲	رنگ پوست	روشن
۳	مقدار طاسی	طاس
٤	خلق و خو	عصبانی
٥	خلاقیت	بسیار خلاق
٦	فعالیت	فعال
۷	ملیت	ایرانی

در واقع برخی از این خصوصیات در ژن ایشان غالب بوده و به فرزندشان که پدرمان باشد انتقال پیدا کرده است. در حقیقت از موارد فوق، خصوصیاتی که پدرمان از پدر بزرگمان به ارث برده عبارتند از:

پدر		
ردیف	عنوان	مقدار
۱	قد	کوتاه
۲	رنگ پوست	روشن
۳	مقدار طاسی	طاس
٤	خلق و خو	عصبانی خوش اخلاق
٥	خلاقیت	بسیار خلاق
٦	فعالیت	فعال
۷	ملیت	ایرانی
۸	سواد	لیسانس
۹	سطح مطالعه	زیاد

همانطور که در جدول فوق مشاهده می شود پدرمان مورد ۴ را از پدر بزرگ به ارث نبرده است. به عبارت دیگر ایشان به جای اینکه عصبانیت را از پدر خود به ارث برده باشد، در عوض فرد خوش اخلاقی است. از سوی دیگر در مورد ۵ هم پدر خلاقیتشان به اندازه پدر بزرگ نیست. به عبارت دیگر ایشان بسیار خلاق نیستند بلکه خلاق هستند. به این اتفاق در زبان برنامه نویسی جاوا Override یا "**رونویسی**" می گویند. به عبارت ساده تر، زمانیکه ما بخواهیم که یک کلاس Subclass تعدادی از ویژگی های کلاس Superclass را به ارث نبرد، می بایست آن ویژگی مد نظر را Override کرد. علاوه بر این به خاطر داشته باشیم که هر موقع یک ویژگی هم در کلاس Superclass موجود باشد و هم در کلاس Subclass ، در این مواقع Compiler ویژگی موجود در Subclass را مد نظر قرار خواهد داد. در نهایت به خاطر داشته باشیم که برای Override کردن اجازه نداریم تا Modifier متد و یا تعداد و یا نوع پارامترهای آن را تغییر دهیم(البته می توانیم نام پارامترهای آن را تغییر دهیم).

علاوه بر مواردی که پدر از پدر خویش به ارث برده، ایشان دارای دو خصیصه دیگر نیز می باشند که مخصوص به خودشان است که این دو خصیصه در موارد ۸ و ۹ گنجانده شده اند. در واقع

میزان سواد ایشان در سطح لیسانس بوده و سطح مطالعه ایشان زیاد می باشد. اکنون ببینیم که پسر چه ویژگی هایی را از پدر به ارث برده است:

پسر		
ردیف	عنوان	مقدار
۱	قد	~~کوتاه~~ بلند
۲	رنگ پوست	روشن
۳	مقدار طاسی	~~طاس~~
٤	خلق و خو	عصبانی ~~خوش اخلاق~~
٥	خلاقیت	~~بسیار~~ خلاق
٦	فعالیت	فعال
۷	ملیت	ایرانی
۸	سواد	~~لیسانس~~ فوق لیسانس
۹	سطح مطالعه	~~زیاد~~ خیلی زیاد

همانطور که در جدول فوق مشاهده می شود، پسر در مورد ۱ برخلاف پدرش و پدر بزرگش قد بلند است، پسر در مورد ۳ برخلاف پدرش و پدر بزرگش طاس نیست، پسر در مورد ۴ برخلاف پدرش که فرد خوش اخلاقی بود، عصبانیت را مجددا از پدر بزرگ خود به ارث برده است. در مورد ۸ پسر بر خلاف پدرش تحصیلاتش را ادامه داده و فوق لیسانس دارد. در مورد ۹ برخلاف پدر که زیاد مطالعه می کند، پسر خیلی زیاد مطالعه می کند. به منظور درک بهتر این روابط از نمودار زیر استفاده می کنیم:

همانطور که در تصویر فوق نشان داده شده است، پدر بزرگ با رنگ قرمز، پدر با رنگ سبز و پسر با رنگ آبی مشخص شده اند. در مستطیل مشکی رنگ خصوصیاتی نشان داده شده است که بین هر سه نسل مشترک می باشند. سپس خصوصیاتی که خاص هر فرد است در بالای سرش مشخص شده است.

در قسمت آموزشی سی و سوم، از خصوصیات مابین پدر بزرگ، پدر و پسر استفاده کرده و پروژه ای جدید می نویسیم که در آن کلیه وراثت ها مورد استفاده قرار گرفته باشند.

پس از مطالعه این آموزش انتظار می رود بتوانیم به سؤالات زیر پاسخ بدهیم:

۱. وارثت در زبان جاوا به به چه صورت به وراثت در دنیای واقعی تشبیه کرد؟
۲. چه زمانیکه در برنامه نویسی بایستی اقدام به رونویسی کنیم؟
۳. تفاوت کلاس اصلی با کلاس زیرشاخته چیست؟