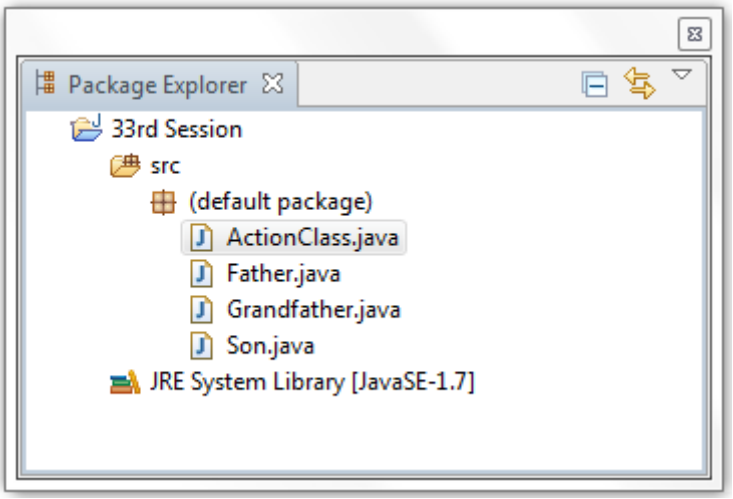
به نام خدا

آموزش سی و سوم

اهداف آموزشی این قسمت عبارتند از:

۱. آشنایی با مبحث وراثت در جاوا
۲. آشنایی با مفهوم Override کردن
۳. آشنایی با نحوه ساخت یک شیئ از روی یک کلاس

پس از آشنایی با مفهوم وراثت در زبان برنامه نویسی جاوا در قسمت سی و دوم در قالب مثال پدر بزرگ، پدر و پسر، اکنون قصد داریم تا با نوشتن پروژه ای مرتبط، این ویژگی منحصر به فرد زبان برنامه نویسی جاوا را مورد بررسی قرار دهیم. برای این منظور پروژه ای تحت عنوان 33rd Session به معنی **"جلسه سی و سوم"** ایجاد کرده و سه کلاس مجزا تحت عناوین Grandfather و Father و Son به معانی به ترتیب **"پدر بزرگ، پدر** و **پسر"** ایجاد می کنیم. برای ساخت این سه کلاس نیازی نیست تا گزینه public static void main را تیک بزنیم چرا که این کلاس ها به منزله نقطه شروع برنامه ما نخواهند بود. سپس کلاسی تحت عنوان ActionClass به معنی **"کلاس اجرایی"** ایجاد کرده و از آنجا که می خواهیم این کلاس به منزله کلاسی باشد که برنامه ما از طریق آن آغاز می شود، پس گزینه public static void main را برای آن در حین ساخت تیک دار می کنیم. اکنون پروژه ما می بایست به شکل زیر تکمیل شده باشد:



برای شروع برنامه نویسی این پروژه، کار خود را از کلاس پدر بزرگ شروع می کنیم. پس از باز کردن این کلاس، کدی مشابه کد زیر خواهیم داشت:

```
public class Grandfather {

}
```

اکنون نیاز داریم تا متدهایی به منظور ذخیره سازی خصوصیات پدر بزرگ در این کلاس ایجاد کنیم. برای همین منظور، کد فوق را به صورت زیر تکمیل می کنیم:

```
public class Grandfather {

	public void showGrandfatherHeight() {
		String height = "Short";
		System.out.println(height);
	}
	public void showGrandfatherSkinColor() {
		String skinColor = "Bright";
		System.out.println(skinColor);
	}
	public void showGrandfatherBoldness() {
		String boldness = "Bold";
		System.out.println(boldness);
	}
	public void showGrandfatherBehavior() {
		String behavior = "Angry";
		System.out.println(behavior);
	}
	public void showGrandfatherCreativity() {
		String creativity = "Very Creative";
		System.out.println(creativity);
	}
	public void showGrandfatherActivity() {
		String activity = "Very Active";
		System.out.println(activity);
	}
	public void showGrandfatherNationality() {
		String nationality = "Iranian";
		System.out.println(nationality);
	}

}
```

همانطور که مشاهده می شود Modifier متدهای خود را public قرار داده ایم چرا که می خواهیم در مابقی کلاس ها هم در دسترس باشند(در آموزش سی و ششم با انواع Modifier ها آشنا خواهیم شد). نام انتخابی برای اولین متد showGrandfatherHeight به معنای **"قد پدر بزرگ را نشان بده"** می باشد. دستوری که برای این متد در نظر گرفته شده اند، به این صورت است که می بایست مقدار شیئ ایجاد شده از روی کلاس String تحت عنوان height به معنی **"قد"** که دارای مقدار اولیه Short به معنی **"کوتاه"** می باشد را نمایش دهد.

متد دوم showGrandfatherSkinColor به معنای **"رنگ پوست پدر بزرگ را نشان بده"** می باشد. دستوری که برای این متد در نظر گرفته شده اند، به این صورت است که می بایست مقدار شیئ ایجاد شده از روی کلاس String تحت عنوان skinColor به معنی **"رنگ پوست"** دارای مقدار اولیه Bright به معنی **"روشن"** می باشد را نمایش دهد.

متد سوم showGrandfatherBoldness به معنای **"میزان طاسی پدر بزرگ را نشان بده"** می باشد. دستوری که برای این متد در نظر گرفته شده اند، به این صورت است که می بایست مقدار شیئ ایجاد شده از روی کلاس String تحت عنوان boldness به معنی **"میزان طاسی"** دارای مقدار اولیه Bold به معنی **"طاس"** می باشد را نمایش دهد.

متد چهارم showGrandfatherBehavior به معنای **"خلق و خوی پدر بزرگ را نشان بده"** می باشد. دستوری که برای این متد در نظر گرفته شده اند، به این صورت است که می بایست مقدار شیئ ایجاد شده از روی کلاس String تحت عنوان behavior به معنی **"خلق و خود"** دارای مقدار اولیه Angry به معنی **"عصبانی"** می باشد را نمایش دهد.

متد پنجم showGrandfatherCreativity به معنای **"خلاقیت پدر بزرگ را نشان بده"** می باشد. دستوری که برای این متد در نظر گرفته شده اند، به این صورت است که می بایست مقدار شیئ ایجاد شده از روی کلاس String تحت عنوان creativity به معنی **"خلاقیت"** دارای مقدار اولیه Very Creative به معنی **"خیلی خلاق"** می باشد را نمایش دهد.

متد ششم showGrandfatherActivity به معنای **"فعالیت پدر بزرگ را نشان بده"** می باشد. دستوری که برای این متد در نظر گرفته شده اند، به این صورت است که می بایست مقدار شیئ ایجاد شده از روی کلاس String تحت عنوان Activity به معنی **"فعالیت"** دارای مقدار اولیه Active به معنی **"فعال"** می باشد را نمایش دهد.

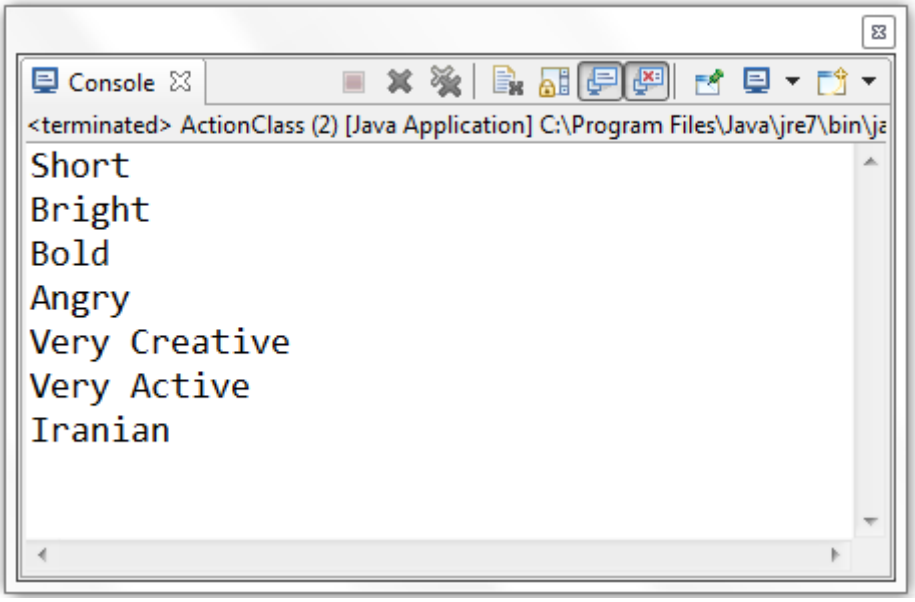
در نهایت متد هفتم showGrandfatherNationality به معنای **"ملیت پدر بزرگ را نشان بده"** می باشد. دستوری که برای این متد در نظر گرفته شده اند، به این صورت است که

می بایست مقدار شیئ ایجاد شده از روی کلاس String تحت عنوان nationality به معنی "**ملیت**" دارای مقدار اولیه Iranian به معنی "**ایرانی**" می باشد را نمایش دهد.

اکنون به منظور تست کردن برنامه خود، یک شیئ از روی کلاس Grandfather در کلاس ActionClass ایجاد کرده سپس متدهای موجود در کلاس پدر بزرگ را فرا می خوانیم. برای این منظور، کلاس ActionClass را باز کرده و آن را به شکل زیر تکمیل می کنیم:

```
public class ActionClass {
	public static void main(String[] args) {
		Grandfather grandfatherObject = new Grandfather();
		grandfatherObject.showGrandfatherHeight();
		grandfatherObject.showGrandfatherSkinColor();
		grandfatherObject.showGrandfatherBoldness();
		grandfatherObject.showGrandfatherBehavior();
		grandfatherObject.showGrandfatherCreativity();
		grandfatherObject.showGrandfatherActivity();
		grandfatherObject.showGrandfatherNationality();

	}

}
```

همانطور که در کد فوق ملاحظه می شود، پس از ساخت یک شیئ از روی کلاس Grandfather تحت عنوان grandfatherObject به معنی "**شیئ پدر بزرگ**"، از این شیئ استفاده کرده و متدهای ایجاد شده در کلاس "**پدر بزرگ**" را فرا می خوانیم. حال پس از اجرای برنامه خروجی مشاهده خواهد شد:



می بینیم که کلیه Attribute ها و Behavior هایی که برای کلاس پدر بزرگ تعریف شده بودند نمایش داده می شوند. اکنون نوبت به تکمیل کلاس پدر می رسد. برای این منظور کلاس Father را باز کرده و آن را به شکل زیر تکمیل می کنیم:

```
public class Father extends Grandfather {

}
```

در کد فوق کلید واژه extends به این معنا است که کلاس Father خصوصیات خود را از کلاس Grandfather به ارث خواهد برد. به عبارت دیگر از این پس هر ویژگی که کلاس پدر بزرگ داشته باشد، کلاس پدر دقیقاً همان ویژگی ها را به ارث خواهد برد. همانطور که در قسمت سی و دوم گفته شد، پدر برخی خصوصیاتش با خصوصیات پدر بزرگ متفاوت است. به عبارت دیگر پدر بزرگ دارای خلق و خوی عصبانی هستند در حالیکه پدر خوش رفتار می باشند، پدر بزرگ خیلی خلاق هستند اما پدر خلاقیتش به اندازه پدر بزرگ نیست. از سوی دیگر پدر دارای دو ویژگی دیگر است که پدر بزرگ فاقد آن ها بود که عبارتند از تحصیلات لیسانس و همچنین میزان مطالعه زیاد.

در زبان برنامه نویسی جاوا به منظور حل مشکلاتی از این دست از دستور Override استفاده می شود. به عبارت دیگر زمانیکه بخواهیم که یک Subclass برخی از ویژگی های Superclass را به ارث نبرد، می بایست آن خصوصیات را Override یا **"رونویسی"** کرد. برای این منظور کلاس پدر را به شکل زیر Override می کنیم:

```
public class Father extends Grandfather {
	@Override
	public void showGrandfatherBehavior() {
		String behavior = "Well=behaved";
		System.out.println(behavior);
	}
	@Override
	public void showGrandfatherCreativity() {
		String creativity = "Creative";
		System.out.println(creativity);
	}
}
```

در حقیقت Override کردن به این شکل صورت می گیرد که ما می بایست پیش از نوشتن نام متد از دستور @Override استفاده کرده سپس آن متد را مجدداً نوشته وصرفاً تغییری در Attribute های آن متد بوجود می آوریم. همانطور که در کد فوق مشاهده می شود، از کل متدهای موجود در کلاس پدر بزرگ فقط دو نمونه که نیاز به Override شدن داشتند را مجدداً نوشته و مقدار آن ها را تغییر داده ایم. در واقع مقدار نمونه behavior را معادل با Well-behaved به معنی **"خوش رفتار"** در نظر گرفته ایم و مقدار نمونه creativity را معادل با Creative به معنی **"خلاق"** در نظر گرفته ایم(اگر خاطرمان باشد مقدار این نمونه برای کلاس پدر بزرگ معادل با Very Creative به معنی **"بسیار خلاق"** بود).

اکنون که موارد مد نظر را Override کردیم، نوبت به اضافه کردن خصوصیاتی می رسد که فقط در پدر وجود دارند که عبارتند از مدرک لیسانس و مطالعه زیاد پدر. از این رو کلاس خو را به شکل زیر تکمیل می کنیم:

```
public class Father extends Grandfather {
	@Override
	public void showGrandfatherBehavior() {
		String behavior = "Well-behaved";
		System.out.println(behavior);
	}
	@Override
	public void showGrandfatherCreativity() {
		String creativity = "Creative";
		System.out.println(creativity);
	}
	public void showFatherEducation() {
		String education = "BA";
		System.out.println(education);
	}
	public void showFatherStudyTime() {
		String studyTime = "Much";
		System.out.println(studyTime);
	}

}
```

همانطور که ملاحظه می شود متدی جدید تحت عنوان showFatherEducation به معنی **"تحصیلات پدر را نشان بده"** با یک شیئ جدید از کلاس String تحت عنوان

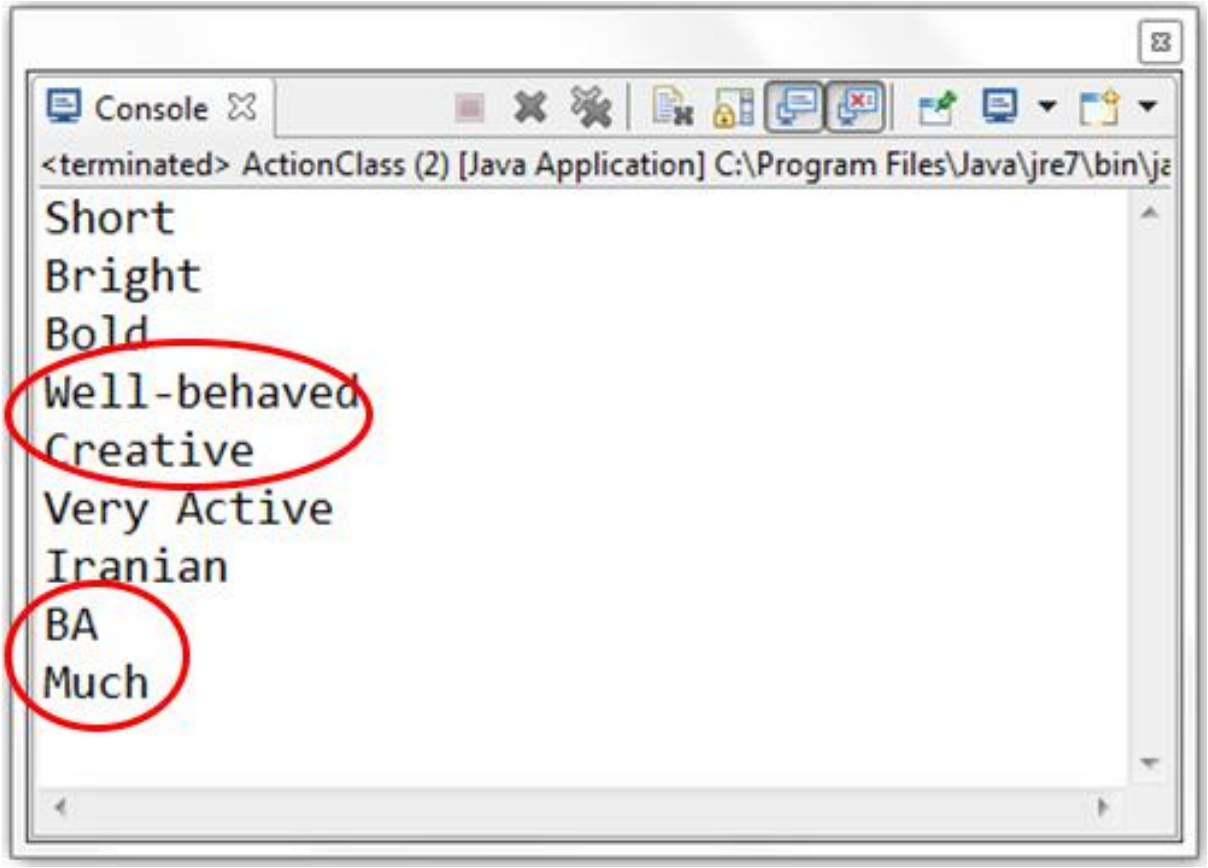
education به معنی **"تحصیلات"** اضافه شده است. مقدار اولیه در نظر گرفته شده برای نمونه education معادل با BA به معنی **"لیسانس"** است(عبارت BA مخفف واژگان Bachelor of Arts می باشد). سپس متدی دیگری تحت عنوان showFatherStudyTime به معنی **"میزان مطالعه پدر را نشان بده"** ایجاد کرده و شیئی از کلاس String تحت عنوان studyTime به معنی **"زمان مطالعه"** در آن ایجاد کرده و مقدار اولیه وارد شده برای نمونه studyTime را معادل با Much به معنی **"زیاد"** قرار داده ایم.

به طور خلاصه این دو متد جدیدی که ایجاد کردیم فقط مخصوص کلاس پدر و کلاس های دیگری است که از این کلاس ارث بری می کنند می باشد(منظور کلاس پسر است).

برای مشاهده کردن عملکرد کلاس پدر، در کلاس ActionClass یک شیئ از روی کلاس پدر ساخته و متدهای آن را به صورت زیر فرا می خوانیم:

```
public class ActionClass {
	public static void main(String[] args) {
		// Grandfather grandfatherObject = new Grandfather();
		// grandfatherObject.showGrandfatherHeight();
		// grandfatherObject.showGrandfatherSkinColor();
		// grandfatherObject.showGrandfatherBoldness();
		// grandfatherObject.showGrandfatherBehavior();
		// grandfatherObject.showGrandfatherCreativity();
		// grandfatherObject.showGrandfatherActivity();
		// grandfatherObject.showGrandfatherNationality();

		Father fatherObject = new Father();
		fatherObject.showGrandfatherHeight();
		fatherObject.showGrandfatherSkinColor();
		fatherObject.showGrandfatherBoldness();
		fatherObject.showGrandfatherBehavior();
		fatherObject.showGrandfatherCreativity();
		fatherObject.showGrandfatherActivity();
		fatherObject.showGrandfatherNationality();
		fatherObject.showFatherEducation();
		fatherObject.showFatherStudyTime();

	}

}
```

همانطور که در کد فوق مشاهده می شود کدهای مربوط به کلاس پدر بزرگ را به منظور جلوگیری از شلوغ شدن خروجی برنامه Comment کرده ایم. حال یک شیئ از روی کلاس پدر به اسم fatherObject به معنی **"شیئ پدر"** ساخته و کلیه متدهای مربوط به پدر بزرگ به علاوه دو متد جدید مربوط به پدر را در آن فرا می خوانیم. اکنون می توانیم برنامه را اجرا کنیم:



همانطور که در خروجی برنامه مشاهده می شود، کلاس پدر کلیه خصوصیات کلاس پدر بزرگ را به ارث برده است با این تفاوت که مقدار نمونه های behavior و creativity به ترتیب به مقادیر Well-behaved و Creative تغییر پیدا کرده اند یا به عبارتی بهتر بگوییم Override شده اند. علاوه بر این پدر دارای دو Attribute دیگر بود که با دایره قرمز رنگ دوم در تصویر فوق مشخص شده اند(علاوه بر این اگر به کد خود به دقت بیشتری نگاه کنیم خواهیم دید که متدهایی که کلاس پدر از کلاس پدر بزرگ به ارث برده است مرتبط با پدر بزرگ هستند مثلا متد showGrandfatherHeight اما متدهای جدید مرتبط به خود کلاس پدر هستند).

اکنون زمان آن فرا رسیده است که به تکمیل کلاس پسر که آخرین کلاس است بپردازیم. برای این منظور کلاس پسر را باز نموده و آن را به صورت زیر تکمیل می کنیم:

```
public class Son extends Father{

}
```

در حقیقت از آنجا که می خواهیم کلاس پسر کلیه خصوصیات خود را از کلاس پدر به ارث ببرد، پس کلید واژه extends را نوشته و سپس نام کلاس Father را می نویسیم. به طور خلاصه بایستی گفت که از آنجا که کلاس پدر برخی خصوصیات خود را از کلاس پدر بزرگ به ارث می برد و کلاس پسر هم برخی خصوصیات خود را قرار است از کلاس پدر به ارث ببرد، این نتیجه را می توان گرفت که کلاس پسر آن دسته از خصوصیاتی را که کلاس پدر از کلاس پدر بزرگ به ارث برده بود را نیز به ارث خواهد برد.

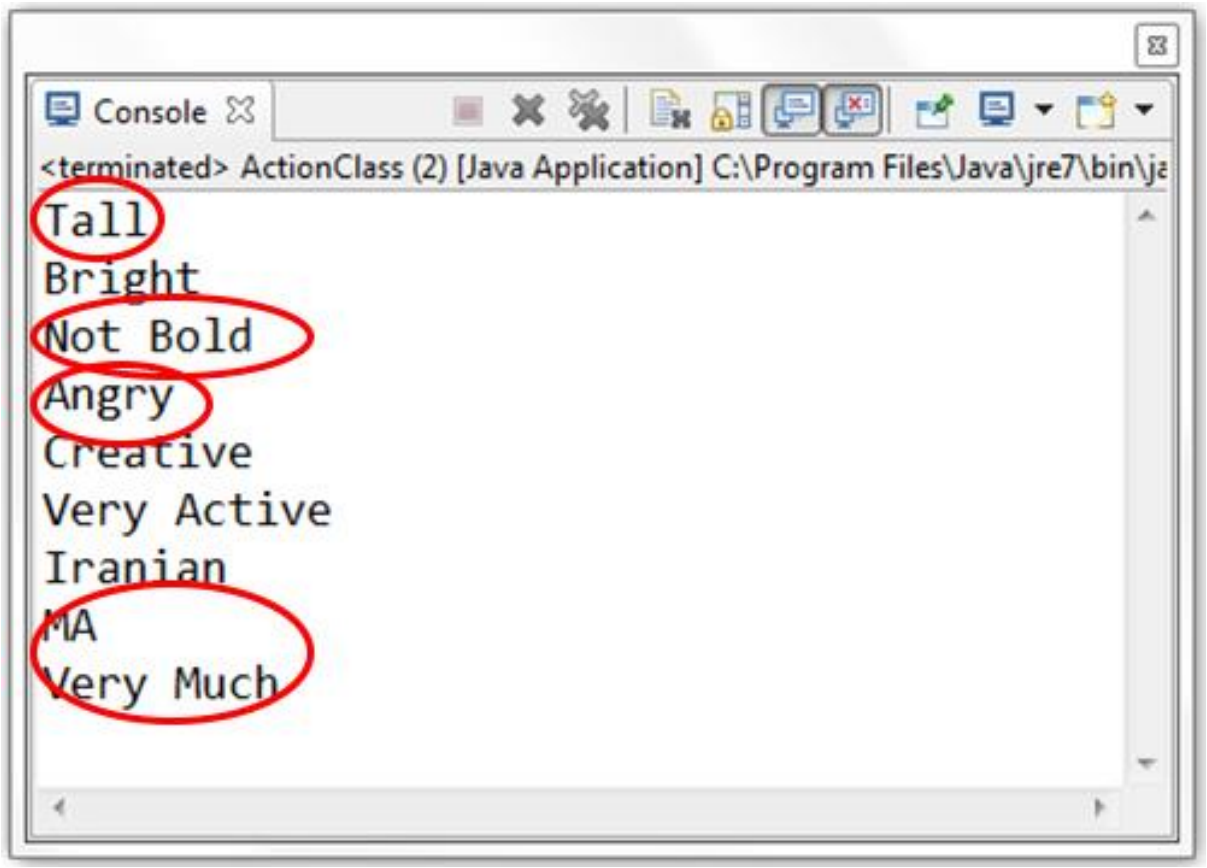
اگر به قسمت آموزشی سی و دوم رجوع کنیم، خواهیم دید که پسر بر خلاف پدر و پدر بزرگ خود قد بلند است، اصلاً طاس نیست، همانند پدر بزرگ خود عصبانی است، بر خلاف پدرش که لیسانس دارد، پسر فوق لیسانس دارد و در نهایت خیلی اهل مطالعه است. حال با آگاهی از این موارد خواهیم توانست کد مربوط به کلاس پسر را ویرایش نماییم:

```
public class Son extends Father {
	@Override
	public void showGrandfatherHeight() {
		String height = "Tall";
		System.out.println(height);
	}
	@Override
	public void showGrandfatherBoldness() {
		String boldness = "Not Bold";
		System.out.println(boldness);
	}
	@Override
	public void showFatherEducation() {
		String education = "MA";
		System.out.println(education);
	}
	@Override
	public void showFatherStudyTime() {
		String studyTime = "Very Much";
		System.out.println(studyTime);
	}

}
```

همانطور که در کد فوق مشاهده می شود، متد اول مربوط به Override قد است که مقدار آن Tall به معنی "**بلند**" قرار داده شده است، Override دوم مربوط به طاسی است که مقدار آن Not Bold به معنی "**طاس نیست**" قرار داده شده، Override سوم مربوط به خلق و خو است که معادل با Angry به معنی "**عصبانی**" قرار داده شده، Override چهارم مربوط به سطح تحصیلات است که معادل با MA به معنی "**فوق لیسانس**" قرار داده شده و در نهایت Override پنجم مربوط به میزان مطالعه است که مقدار آن Very Much به معنی "**خیلی زیاد**" قرار داده شده است(عبارت MA مخفف واژگان Master of Arts است). اکنون می توانیم به کلاس ActionClass رفته و آن را به شکل زیر بازنویسی کنیم:

```
public class ActionClass {
	public static void main(String[] args) {
		// Grandfather grandfatherObject = new Grandfather();
		// grandfatherObject.showGrandfatherHeight();
		// grandfatherObject.showGrandfatherSkinColor();
		// grandfatherObject.showGrandfatherBoldness();
		// grandfatherObject.showGrandfatherBehavior();
		// grandfatherObject.showGrandfatherCreativity();
		// grandfatherObject.showGrandfatherActivity();
		// grandfatherObject.showGrandfatherNationality();

		// Father fatherObject = new Father();
		// fatherObject.showGrandfatherHeight();
		// fatherObject.showGrandfatherSkinColor();
		// fatherObject.showGrandfatherBoldness();
		// fatherObject.showGrandfatherBehavior();
		// fatherObject.showGrandfatherCreativity();
		// fatherObject.showGrandfatherActivity();
		// fatherObject.showGrandfatherNationality();
		// fatherObject.showFatherEducation();
		// fatherObject.showFatherStudyTime();

		Son sonObject = new Son();
		sonObject.showGrandfatherHeight();
		sonObject.showGrandfatherSkinColor();
		sonObject.showGrandfatherBoldness();
		sonObject.showGrandfatherBehavior();
		sonObject.showGrandfatherCreativity();
		sonObject.showGrandfatherActivity();
		sonObject.showGrandfatherNationality();
		sonObject.showFatherEducation();
		sonObject.showFatherStudyTime();

	}

}
```

همانطور که در کد فوق مشاهده می شود کلیه ی کد های پیشین را Comment کرده ایم و سپس یک شیئ از روی کلاس Son به اسم sonObject به معنی **"شیئ پسر"** ساخته ایم و پس از آن کلیه ی متدها را فرا خوانده ایم. پس از اجرای برنامه تصویر زیر مشاهده خواهد شد:



همانطور که در تصویر فوق مشاهده می شود، بیضی قرمز رنگ اول نشانگر اولین Override در کلاس Son است که متد مرتبط با قد را بازنوسی کرده است. بیضی قرمز رنگ دوم نشانگر دومین Override است که متد مرتبط با طاسی را بازنویسی کرده است. بیضی قرمز رنگ سوم نشانگر سومین Override است که متد مرتبط با خلق و خو را بازنوسی کرده است و در نهایت بیضی قرمز رنگ چهارم نشانگر چهارمین و پنجمین Override است که متدهای مرتبط با سطح تحصیلات و میزان مطالعه را بازنوسی کرده اند.

شاید با مثالی که در قسمت پیشین زده شد و پروژه ای که در این قسمت برنامه نوسی شد، بهتر به این مسئله پی ببریم که طراحان زبان برنامه نوسی جاوا دیدشان به شیئ گرایی در این زبان از دید ایشان به اشیاء در دنیای واقعی نشات گرفته است که همین وجه تشابه مابین شیئ گرایی در زبان برنامه نوسی جاوا و روابط اشیاء در دنیای واقعی می تواند ما را در درک بهتر OOP یاری رساند.

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

۱. چرا بایستی از دستور Override برای برخی متدها استفاده کرد؟
۲. چرا کلاس پسر برخی ویژگی های کلاس پدر بزرگ را به ارث برده است؟