به نام خدا

آموزش پنجاه و یکم

اهداف آموزشی این قسمت عبارتند از:

۱. معرفی کلاس Thread ها در زبان برنامه نویسی جاوا
۲. چند مثال از کاربر Thread ها
۳. آشنایی با Interface ها
۴. تفاوت های Interface ها با Class ها

در این آموزش قصد داریم تا یکی از مباحث بسیار مهم در زبان برنامه نویسی جاوا را مورد بررسی قرار دهیم تحت عنوان Thread. به طور کلی می توان گفت که Thread ها دارای کاربردهای فراوانی هستند که از آن جمله می توان به طراحی بازی ها اشاره کرد (لازم به ذکر است که واژه انگلیسی Thread دارای معانی متعددی در زبان فارسی است که از آن جمله می توان به "نخ، رشته، ریسمان" اشاره کرد).

به طور کلی هر زمانیکه بخواهیم در زبان برنامه نویسی جاوا بیش از یک کار را به طور هم زمان پیش ببریم، بایستی از کلاسی تحت عنوان Thread استفاده نماییم. برای درک بهتر مفهوم Thread ها مثالی را مد نظر قرار خواهیم داد. اگر توجه کرده باشیم زمانیکه یک فایل ویدیویی را به صورت آنلاین تماشا می کنیم نیازی نیست که صبر کنیم تا فایل کاملاً بارگذاری شود بلکه به محض آنکه بخش ابتدایی فایل بار گذاری شد، قادر خواهیم بود تا به تماشای ویدیو بپردازیم تا ادامه بارگذاری صورت گیرد. به عبارت دیگر دو کار نمایش ویدیو و بارگذاری آن به صورت هم زمان صورت می گیرند. این دقیقاً کاری است که Thread ها برای انجام آن طراحی شده اند. به عبارت دیگر به جای آنکه صبر کنیم تا یک کار تکمیل شود سپس کار بعدی آغاز گردد (که بالتبع زمان را از دست خواهیم داد)، هر دو کار را به صورت هم زمان پیش خواهیم برد. اهمیت چنین مسئله ای در بازی هایی همچون بازی های اندرویدی بیشتر خود را نشان خواهد داد. فرض کنیم در یک بازی ابرها در آسمان در حال حرکت هستند (کار شماره یک) و در همان حال تعدادی پرنده نیز از دور دست ها وارد صحنه می شوند (کار شماره دو). حال شکارچی می بایست با شلیک به

سمت پرنده ها آنها را شکار کند (کار شماره سه). در واقع این سه کار به طور هم زمان انجام می شوند.

این در حالی است که اگر در طراحی بازی از کلاس Thread استفاده نشود اول بایستی صبر کنیم تا حرکت ابرها به اتمام برسد سپس پرنده ها وارد صحنه شوند و پس از آنکه حرکت پرنده ها تمام شد، شکارچی می تواند بدون آنکه نگران این مسئله باشد که ممکن است تیرهایش به خطا بروند اقدام به شکار پرنده نماید چرا که دیگر پرنده ها نمی توانند حرکت کنند. حال کاملاً مشخص است که بدون استفاده از Thread ها، بازی ها جذابیت خود را از دست خواهند داد.

از سوی دیگر کاربر Thread ها صرفاً در طراحی بازی ها نیست بلکه در اپلیکیشن ها و برنامه های تحت وب و یا دسکتاپ نیز کاربردهای فراوانی دارند. به طور مثال فرض کنیم که یک اپلیکیشن تحت وب با زبان جاوا طراحی کرده ایم که یک وب سرور است. حال این سرور وظیفه دارد تا در آن واحد بیش از یک کاربر را سرویس دهی کند که این کار به سادگی با استفاده از کلاس Thread امکان پذیر خواهد بود.

حال پس از آشنایی با مفهوم Thread در برنامه نویسی لازم است این نکته را نیز متذکر شویم که انجام هم زمان چند کار در زبان برنامه نویسی جاوا اصطلاحاً Concurrency نامیده می شود (این واژه در زبان فارسی به "هم زمانی" و "تقارن" ترجمه می شود).

آشنایی با Interface ها

پیش از ادامه آموزش پیرامون ماهیت Thread ها نیاز به آشنایی با Interface ها در زبان برنامه نویسی جاوا داریم. واژه انگلیسی Interface به معنی "رابط" و "واسطه" می باشد. برای روشن شدن این مطلب مثالی ذکر می کنیم:

فرض کنیم که عضو یک تیم برنامه نویسی هستیم که روی یک پروژه عظیم کار می کنیم. در واقع هر یک از اعضای این تیم مسئول نوشتن بخشی از برنامه است بدون آنکه به کار دیگر اعضاء کاری داشته باشد اما این در حالی است که یک نقشه اصلی ترسیم شده و در آن نقش کلیه اعضاء مشخص شده اند که هر کسی باید بر اساس آن نقشه کار خود را تکمیل کند. در این نقشه آمده است که برنامه نویس الف می بایست کدهای سمت سرور را بنویسد، برنامه نویس ب می بایست کدهای سمت کاربر را بنویسد و برنامه نویس پ بایستی UI نرم افزار را کدنویسی کند و ...

از سوی دیگر این نقشه حاوی اطلاعاتی است مبنی بر اینکه هر برنامه نویس طبق چه استانداردی بایست به کدنویسی بپردازد. به عبارت دیگر برنامه نویسان نمی توانند طبق نظر شخصی به هر سبکی که تمایل دارند کدنویسی کنند. در واقع Interface ها در زبان جاوا همانند آن نقشه می باشند

بطوریکه نه تنها وظایف بخش های مختلف یک برنامه را مشخص می کنند بلکه خصوصیات آن را نیز مشخص می کنند.

شباهت های Interface ها با Class ها را می توان در موارد زیر خلاصه کرد:

۱. یک Interface به هر تعداد که بخواهد می توان متد داشته باشد.
۲. نام فایلی که یک Interface در آن قرار دارد می بایست با نام مد نظر برای Interface یکی باشد و این در حالی است که پسوند فایل همانند کلاس ها java است. مثلا MyInterface.java

در زبان برنامه نویس جاوا Interface ها همانند کلاس ها می باشند اما این در حالی است که در موارد زیر با کلاس ها متفاوت می باشند:

۱. یک کلاس فقط و فقط می تواند از یک کلاس دیگر وارثت داشته باشد اما این در حالی است که یک کلاس می تواند بیش از یک Interface را اجرا کند.
۲. یک کلاس می تواند کلاس دیگر را extends کند اما این در حالی است که یک کلاس می تواند اقدام به implements کردن یک Interface کند (در ادامه آموزش بیشتر با کلید واژه implements آشنا خواهیم شد).
۳. همانطور که در آموزش سی و هفتم توضیح داده شد، یک متد دارای Body و Signature است. کلاس ها می توانند حاوی متدهایی به همراه Body باشند اما این در حالی است که Interface ها فقط می توانند حاوای متدهایی با Signature آنها باشند. به عبارت دیگر متدهای قرار گرفته در Interface ها به صورت Abstract یا "انتزاعی" هستند.
۴. بر خلاف کلاس ها، از روی Interface ها نمی توان Object یا شیئ ساخت.
۵. به طور کلی Interface ها می توانند از روی دیگر Interface ها وارثت داشته باشند.

در آموزش آتی با نحوه ساخت یک Interface بیشتر آشنا خواهیم شد.

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

۱. فایده استفاده از Thread ها در برنامه نویسی چیست؟
۲. منظور از یک Interface در زبان برنامه نویسی جاوا چیست؟

۳. تفاوت های یک Interface با یک Class چیست؟