**گزارش پیاده‌سازی**

پیاده‌سازی نرم‌افزار شبیه‌ساز معاملات بورس پس از بحث و بررسی با کارشناسان شرکت فن‌آوری بورس تهران و توافق بر سر الگوریتم‌های تولید سفارش‌ها، نهوه‌ی صف بندی آن‌ها و انجام معاملات شروع شد. نحوه‌ی ماژول بندی این نرم‌افزار و وظایف هر کدام از ماژول‌ها در مستند طراحی به تفضیل شرح داده شده است. در این مستند مراحلی که توسط هر کدام از این ماژول‌ها و توابع موجود در آنها برای نیل به هدف شبیه‌سازی صورت می‌گیرد را شرح می‌دهیم.

MainServlet در ماژول web، نقطه‌ی ارتباط با کاربر و Servlet اصلی نرم‌افزار است که با فراخوانی آن به صفحه‌ی اصلی نرم‌افزار هدایت می‌شویم. با ثبت تنظیم‌ها و معرفی فایل‌های داده‌ی مورد نیاز این Servlet تنظیمات مورد نیاز برای کار نرم‌افزار را روی شی‌ای که از کلاس Settings ساخته شده و در spring context موجود است انجام می‌دهد. همینطور این Servlet وظیفه‌ی دریافت اطلاعات مربوط به گزارش‌ها و نموداری که در صفحه‌ی اصلی کشیده می‌شود را از engine بر عهده دارد. Servlet دیگری که در این ماژول موجود است CommandServlet‌ می‌باشد که برای دریافت فرامین و انتقال آنها به Generator از آن بهره می‌گیریم. این Servlet چهار فرمان شروع پروسه (start)، پایان پروسه (stop)، وقفه/ادامه (pause) و شروع مجدد (restart) را به Generator منتقل می‌کند و نتایج این فرمان‌ها را برای نمایش باز می‌گرداند. صفحه‌ی اصلی رابط کاربری نرم‌افزار به نام index.jsp در مسیر main/webapp از زیرشاخه‌ی src پروژه موجود است. به منظور ساده‌سازی و جلوگیری از load شدن‌های چند باره‌ی این صفحه، همه‌ی فعل و انفعالات این صفحه با Servlet ها توسط Javascript و با استفاده از تکنولوژی Ajax صورت می‌پذیرد. بنابراین در حین کار با رابط کاربری این نرم‌افزار ما صرفا با همین صفحه روبرو خواهیم بود. همینطور timer های موجود در این صفحه متناوبا وضعیت engine را چک می‌کنند و در هر لحظه بنا به وضعیت سیستم عملیات مختلف مورد نیاز (کشیدن نمودار، نمایش گزارش و ...) را انجام می‌دهند.

کلاس GeneratorImpl در ماژول Generator که کلاس اصلی این ماژول محسوب می‌شود دارای یک Thread‌ اصلی برای انجام مراحل مختلف پروسه می‌باشد و در هر مرحله بنا به نیاز Thread های دیگری را ایجاد کرده و وظایف مختلف را به آنها محول می‌نماید. نحوه‌ی کار Thread اصلی در این ماژول که با فراخوانی متد startProcess ساخته می‌شود به این شکل است که ابتدا Logger سیستم و سپس engine را از شروع پروسه با خبر می‌سازد تا مراحل اولیه در هر کدام انجام پذیرد. سپس دو Thread ایجاد می‌‌کند که وظیفه‌ی یکی تولید سفارش‌های خرید و دیگری تولید سفارش‌های فروش می‌باشد، و این Thread ها را اجرا می‌کند. پس از آن به مدت تنظیم شده برای پروسه‌ی Pre-opening منتظر می‌ماند. اگر در طول این مدت فرمان پایان از رابط کاربری برسد Thread اصلی interrupt می‌شود. Thread اصلی پس از هر مرحله وضعیت سیستم را که در شی Settings ثبت می‌شود چک می‌کند و بنا به وضعیت ثبت شده عکس‌العمل نشان می‌دهد. پس از انتظار برای پایان یافتن پروسه‌ی Pre-opening ماژول Generator فرمان شروع پروسه‌ی Trading را به engine‌ ارسال می‌کند و خود دوباره دو Thread دیگر برای تولید سفارش ها ایجاد کرده و آنها را اجرا می‌نماید و دوباره تا پایان یافتن مدت زمان Trading منتظر می‌ماند. پس از پایان یافتن مدت زمان trading ماژول Generator فرمان پایان یافتن پروسه را به engine ارسال می‌کند.

کلاس EngineImpl که پیاده‌سازی engine‌ توسط آن صورت گرفته حاوی دو نوع از متد‌هاست. گروهی از این متد‌ها برای گرفتن اطلاعات داخلی engine توسط رابط کاربری به کار می‌روند که نام همه‌ی آنها با get‌ شروع می‌شود. باقی متد‌ها برای ارتباط با Generator به کار می‌روند و فرامین را از او دریافت می‌کنند. Engine داراری دو Map به نام‌های buyQueues و sellQueues می‌باشد که این دو Map برای هر نماد صف‌های خرید و فروش را نگهداری می‌کنند. همینطور map دیگری با نام tradingThreads وجود دارد که به ازای هر نماد یک Thread برای مقایسه‌ی سفارش‌ها و انجام معاملات در آن نگهداری می‌شود. همینطوری engine دارای شمارنده‌‌های مختلفی نیز هست که مقادیر مختلف مورد نیاز برای گزارش‌ها را در آنها نگهداری می‌کند. Engine با استفاده از متد putOrder سفارش‌ها را دریافت می‌کند و آنها را در صف مربوط به خود قرار می‌دهد. این صف‌ها از نوع sorted می‌باشند که هر شی‌ای که در آنها قرار می‌گیرد با توجه به متد compareTo که از کلاس آن override شده به صورت sort شده در صف قرار خواهد گرفت. متد compareTo از کلاس Order که حاوی اطلاعات سفارش است به شکلی پیاده‌سازی شده که صف‌ها با الگوریتم Origin FIFO مرتب شوند. ترتیب sort در دو صف خرید و فروش بر خلاف یکدیگر است. به این معنی که مثلا دو سفارش از مشتریانی با priority یکسان در صف خرید به شکلی sort می‌شوند که سفارش با قیمت بالاتر در ابتدای صف قرار گیرد ولی در صف فروش سفارش با قیمت پاینتر در ابتدای صف قرار خواهد گرفت. پیاده‌سازی Thread های Trading نیز در همین کلاس انجام گرفته. هر Thread در صورتیکه سفارشی در صف‌ها موجود باشد سفارش ابتدایی هر دو صف خرید و فروش را برداشته و باهم مقایسه می‌کند. در صورتیکه این دو سفارش منجر به معامله شدند تعداد آنها را از هم کم می‌کند و اگر از سفارشی تعدادی باقی ماند آنرا به صف باز می‌گرداند. در صورتیکه با سفارش‌های ابتدای هر دو صف امکان انجام معامله وجود نداشته باشد، سفارش‌ها به صف بازگردانده می‌شوند و Thread منتظر می‌ماند تا سفارش جدیدی به صف وارد شود. متد PutOrder وارد شدن سفارش جدید را به اطلاع Thread می‌رساند. با هر سفارش جدیدی که وارد صف شود این پروسه تکرار می‌شود.