دانشگاه صنعتی شریف
دانشکده‌ی مهندسی کامپیوتر

سامانه‌های بی‌درنگ

عنوان:

گزارش پروژه پایانی

اعضای گروه :

امیرمهدی نامجو، پرهام صارمی، صبا هاشمی

استاد :

دکتر انصاری

تابستان ۱۴۰۱

فهرست مطالب

مقدمه ۳
ساختار پروژه ۳
توضیحات کلاس‌ها ۳
Main ۳
Setting ۳
Clock ۳
ClockThread ۳
TimeStepThread ۳
ThreadBlock ۴
CpuUtilityThread ۴
IClock ۴
ClockData ۴
DummyClock ۴
UserInterface ۴
UIThread ۴
TerminalUI ۴
GUI ۴
BaseFrame ۴
فرایند مدیریت تردها ۴
اجرای برنامه ۵

مقدمه

در این پروژه با استفاده از کتابخانه‌ی RealTime جاوا به پیاده‌سازی سیستمی شامل ۴ عدد ساعت که هر کدام تایم مخصوص به خود را نشان می‌دهند پرداخته‌ایم. هر ساعت یک اولویت دارد که با توجه به لود و کارکرد CPU عملکرد بعضی از ساعت‌ها متوقف می‌شود. این سیستم با استفاده از قابلیت‌های کتابخانه‌ی RealTime برای مولتی‌تردینگ و همچنین استفاده از Swing برای UI پیاده‌سازی شده است. در ادامه به صورت مفصل به تشریح اجزای پروژه می‌پردازیم.

ساختار پروژه

```
clock
├── Main.java
├── Setting.java
├── ClockThread.java
│   ├── Clock
│   └── ClockThread
├── TimeStepThread.java
├── ThreadBlock.java
├── CpuUtilityThread.java
└── UserInterface.java
    ├── IClock
    ├── ClockData
    ├── DummyClock
    ├── UserInterface
    ├── UIThread
    ├── TerminalUI
    ├── GUI
    └── BaseFrame
```

توضیحات کلاس‌ها

Main

این کلاس مدیریت ساخت کلاس‌های دیگر و پیکربندی کلی کلاس‌ها در کنار هم را بر عهده دارد و با اجرای تابع main این کلاس سیستم شروع به کار می‌کند.

Setting

این کلاس برای تنظیمات اولیه‌ی پروژه مورد استفاده قرار می‌گیرد. مواردی مانند تعداد ساعت‌ها، نوع رابط کاربری، اولویت ساعت‌ها و مقادیر اولیه‌ی آن‌ها به کمک این کلاس تنظیم می‌شود.

Clock

این کلاس نگه‌دارنده‌ی مقدار اصلی time است و فقط دو متد getTime و increase را ارائه می‌دهد.

ClockThread

این کلاس که از RealtimeThread ارث‌بری می‌کند وظیفه‌ی بروزرسانی هر کدام از ساعت‌ها را از روی زمانی که در کلاس Clock قرار دارد، بر عهده دارد.

TimeStepThread

این کلاس که از RealtimeThread ارث‌بری می‌کند وظیفه‌ی بروزرسانی زمان ذخیره شده در کلاس Clock را برعهده دارد.

ThreadBlock

وظیفه‌ی این کلاس نگه‌داری از کلاس ClockThread و هم چنین کلاس مربوط به رابط کاربری هر ساعت در یک مکان است. هم چنین در این کلاس یک قفل قرار دارد که برای جلوگیری از ادامه کار یک ساعت در صورت بالا رفتن لود CPU مورد استفاده قرار می‌گیرد.

CpuUtilityThread

این کلاس که از RealtimeThread ارث‌بری می‌کند وظیفه‌ی مدیریت تردها را با توجه به لود سیستم بر عهده دارد. در این کلاس به تمامی ThreadBlock ها دسترسی دارد و در صورت زیاد شدن لود CPU به کمک قفل موجود در ساختار ThreadBlock از ادامه‌ی اجرای تردهای با اولویت کم‌تر جلوگیری می‌کند.

IClock

IClock یک اینترفیس است که دو متد getID و getTime را ارائه می‌دهد. برای این که بتوانیم رابط کاربری را مستقل از ساختار کلاس‌های اصلی ساعت‌ها بسازیم و آن را تست کنیم، از این اینترفیس استفاده می‌کنیم. کلاس ClockThread و DummyClock این اینترفیس را محقق می‌کنند.

ClockData

این کلاس با گرفتن زمان بر حسب ثانیه، زمان را بر حسب ساعت، دقیقه و ثانیه ارائه می‌دهد.

DummyClock

این کلاس اینترفیس IClock را محقق می‌کند و برای تست رابط کاربری استفاده می‌شود.

UserInterface

این کلاس برای تست رابط کاربری بدون نیاز به کلاس‌های اصلی ساعت‌ها و با استفاده از DummyClock مورد استفاده قرار می‌گیرد.

UIThread

این کلاس که از RealtimeThread ارث‌بری می‌کند فقط به عنوان پدر مشترک دو نوع کلاس رابط کاربری در کلاس‌های اصلی ساعت‌ها مورد استفاده قرار می‌گیرد.

TerminalUI

این کلاس که از UIThread ارث‌بری می‌کند برای نمایش ساعت‌ها تحت ترمینال به کار می‌رود.

GUI

این کلاس که از UIThread ارث‌بری می‌کند برای نمایش ساعت‌ها به صورت گرافیکی به کار می‌رود.

BaseFrame

این کلاس صفحه‌ی اصلی نمایش به صورت گرافیکی را آماده می‌کند، که در صورت استفاده از رابط کاربری گرافیکی مورد استفاده قرار می‌گیرد.

فرایند مدیریت تردها

تردهای ساعت و هم‌چنین رابط کاربری در کلاس Main ایجاد شده و شروع به اجرا می‌کنند.

ترد مربوط به کلاس CpuUtilityThread که وظیفه‌ی کنترل لود CPU را بر عهده دارد دارای اولویت بیشینه و تردهای مربوط به رابط کاربری کم‌ترین اولویت را دارند.

هم چنین ترد مربوط به کلاس TimeStepThread که وظیفه‌ی بروزرسانی ساعت اصلی را دارد بیشترین اولویت را دارا است.

اولویت اولیه‌ی خود ساعت‌ها از طریق آرگومان‌های اجرای برنامه قابل تنظیم است. در حین اجرای برنامه نیز تغییر اولویت ساعت‌ها از طریق رابط کاربری امکان‌پذیر می‌باشد.

ترد کلاس CpuUtilityThread هر ۱۰۰ میلی‌ثانیه میزان لود CPU را مانیتور می‌کند. در صورتی که لود از مقدار ۷۰ درصد بالاتر رفته باشد، به ماکسیمم دو ترد ساعت اجازه‌ی کار می‌دهد و قفل مربوط به ساعت‌های دیگر را می‌گیرد تا آن‌ها منتظر قفل بخوابند و نتوانند آپدیت شوند. هم چنین اگر لود از مقدار ۹۰ درصد بالاتر برود، فقط یک ترد ساعت اجازه‌ی کار خواهد داشت.

به همین صورت در صورتی که لود از مقدار ۹۰ درصد پایین بیاید امکان اجازه کار به ماکسیمم دو ترد و اگر از ۷۰ درصد پایین‌تر بیاید اجازه‌ی کار به همه‌ی تردها داده می‌شود.

اجرای برنامه

هنگام اجرای برنامه می‌توان نوع رابط کاربری، تعداد ساعت‌ها، مقادیر اولیه‌ی اولویت‌ها و هم چنین مقادیر اولیه‌ی ساعت‌ها را مشخص کرد. فرمت ورودی به صورت زیر است:

Main [GuiType] [NumberOfClocks] [InitialPriorityArray] [InitialTimeInSeconds]

تایپ رابط کاربری می‌تواند صفر یا یک باشد. صفر به معنای رابط گرافیکی و یک به معنای رابط ترمینال است. به طور مثال فرض کنید تنظیمات زیر مد نظر باشد:

- GUI TYPE = 0
- Number of Clocks = 4
- Initial Priority Array = [3 1 1 2]
- Initial time in seconds = [10 100 500 1000]

برای این تنظیمات باید از کامند زیر استفاده کنیم:

```
Main 0 4 3 1 1 2 10 100 500 1000
```