باسمه تعالی

سیستم‌های عامل پیشرفته (۴۰-۵۳۳)

Advanced Operating Systems

دانشکده‌ی مهندسی کامپیوتر

علیرضا آقامحمدی

پروژهٔ اول: لحظه‌نگاری سراسری **موعد ارسال: ۱۳۹۶/۰۱/۱۷**

مقدمه

سامانهٔ لحظه‌نگاری سراسری، یک سامانهٔ متن‌باز با انعطاف‌پذیری بالا مبتنی بر زبان جاوا است. هستهٔ این نرم‌افزار یک سامانهٔ بانک‌داری توزیع شده با قابلیت انتقال پول از یک شعبه به شعب دیگر است. برای حفظ صحت، برنامه از ثبت حالت سراسری سامانه پشتیبانی می‌کند. از جمله ویژگی‌های فنی این محصول می‌توان به موارد ذکر شده در جدول ۱ اشاره کرد.

جدول ۱: ویژگی‌های فنی سامانهٔ لحظه‌نگاری

ویژگی	مقدار
زبان	جاوا نسخهٔ ۸
سیستم‌عامل	مستقل از سیستم‌عامل
پروانه	MIT

معماری

توسعه و بهبود سامانهٔ لحظه‌نگاری سراسری نیازمند درک مناسبی از معماری آن است.

کلاس‌ها و توابع مهم

در این بخش به شرح و معرفی کلاس‌ها و توابع مهم می‌پردازیم.

۱.۰ Snapshot

از کلاس مذکور برای مدیریت رویدادهای لحظه‌نگاری استفاده می‌شود. به عنوان نمونه شعبهٔ بانک در یک لحظهٔ زمانی خاص عملیات لحظه‌نگاری را آغاز می‌کند.

۱.۱.۰ startSnapshot

با استفاده از این تابع عملیات لحظه‌نگاری آغاز می‌شود. شعبهٔ بانک، شناسه‌های دیگر شعب را ذخیره می‌کند.

```
public void startSnapshot(Bank bank, int balance, List<Bank> banks){
  ...
```

```
  incomingChannels
    .addAll(banks
    .parallelStream()
    .filter(b->b.getId()!=bank.getId())
    .map(Bank::getId)
    .collect(Collectors.toSet()));
}
```

incrementMoneyInTransit ۲.۱.۰

با استفاده از این تابع مقدار اعتبار در حال انتقال به بانک، ثبت می‌شود.

```
public void incrementMoneyInTransit(int recipientBankId, Bank bank){
...
if(incomingChannels.contains(recipientBankId))
  this.moneyInTransit += bank.getBalance();
}
```

stopRecording ۳.۱.۰

با استفاده از این تابع عملیات لحظه‌نگاری از بانک مقصد را متوقف می‌شود.

```
public void stopRecording(Bank bank){
  incomingChannels.remove(bank.getId());
}
```

BankServerRemote ۲.۰

از کلاس مذکور برای انجام عملیات انتقال، دریافت و لحظه‌نگاری سراسری به شکل از راه دور استفاده می‌شود.

sendMoney ۱.۲.۰

با استفاده از این تابع عملیات انتقال اعتبار از بانک مبدأ به مقصد انجام می‌شود.

```
public void sendMoney(...) throws RemoteException {
  withdrawLock.lock();
  try {
    boolean isWithdraw = bankDao.withdraw(bank);
```

```
    if(isWithdraw){
      ...
      boolean isTransferred = bankDao.getRemoteBank(...).recieveMoney(...);
      ...
    }
  } finally {
    withdrawLock.unlock();
  }
}
```

recieveMoney ۲.۲.۰

با استفاده از این تابع عملیات دریافت اعتبار توسط بانک مقصد انجام می‌شود.

```
public boolean recieveMoney(...) throws RemoteExceeption {
  depositLock.lock();
  try {
    bankDao.getBank(...).getSnapshot().incrementMoneyInTransit(...);
    bankDao.deposit(bank);
    return true;
  } finally {
    depositLock.unlock();
  }
}
```

recieveToken ۳.۲.۰

با استفاده از این تابع عملیات لحظه‌نگاری سراسری توسط بانک مبدأ انجام می‌شود.

```
public void receiveToken(int receiverBankId, int senderBankId) throws RemoteException {
  tokenLock.lock();
  depositLock.lock();
  withdrawLock.lock();
  try {
I   ...
    Snapshot snapshot = bankDao.getBank(senderBankId).getSnapshot();
    if (!snapshot.isRecording()) {
      snapshot.startSnapshot(...);
      ExecutorService executorService = Executors.newFixedThreadPool(...);
      bankDao()
```

```
        .allBanks()
        .parallelStream()
        .filter(b -> b.getId() != senderBankId)
        .forEach(bank -> executorService.execute(() -> {
          ...
          bankDao.getRemoteBank(bank).recieveToken(...);
          ...
        }));
    }
    snapshot.stopRecording(bankDao.getBank(...));
    if(!snapshot.isRecording()){
      ...
      bankDao.getBank(...).getSnapshot().stopSnapshot();
    }
  } finally {
    tokenLock.unlock();
    depositLock.unlock();
    withdrawLock.unlock();
  }
}
```