بسمه تعالی

# گزارشکار تکلیف اول برنامه نویسی
# پیاده سازی رمز ارز

استاد مربوطه : دکتر جهانشاهی

تدریسیار مربوطه: مهندس کیان بهزاد

نویسنده : محمدمهدی شریفیان(9823053)

# مقدمه

در این پروژه دو کلاس Server و Client را پیاده  سازی کردیم که در مجموع نمایش مختصری از نحوه کار رمز ارز ها به صورت  Centralized ارائه شد که چالش هایی مانند پیاده سازی دو کلاس وابسته بدون استفاده از nested class، دسترسی به متغییر های private از خارج از کلاس  ، کار با پوینتر را دارا  بود.

# کلاس ها

همانطور که گفته شد در این پروژه پیاده سازی دو کلاس Client و Server مد نظر بود که که هر کدام از آنها ویژگی های خاص و  رفتار های خاصی را داشتند که در ادامه به بررسی آنها میپردازیم.

## class Client

هر instance از این کلاس دارای متغییر ها و توابعی است که به در ادامه به توضیح آنها میپردازیم. به طور کلی این کلاس رفتار یک کاربر را شبیه سازی میکند که میتواند کار هایی نظیر انتقال رمز ارز و یا تولید nonce برای استخراج رمز ارز را انجام دهد. و ویژگی هایی مانند id, public key, private key را دارد. در ادامه به بررسی توابع و متغیر های این کلاس میپردازیم.

```
private :
   Server const* const server;
   const std::string id;
   std::string public_key;
   std::string private_key;
```

همانطور که دیده میشود تمام متغییر های این کلاس از نوع private هستند و خارج از کلاس به آنها دسترسی نداشتیم. متغیر اول از نوع Server است که این متغیر دارای آدرس ثابت است و خود این متغیر هم از ثابت است و نمیتوان اشاره گر به این متغیر، محتویات و خود این این متغیر را برای هرInstance از این کلاس تغییر داد.متغیر بعدی id هرClient میباشد که Client های دیگر میتوانند با استفاده از آن انتقال رمز ارز هارا انجام دهند و برای هر Client همواره ثابت است و قابلیت تغییر ندارد. دو متغیر بعدی برای احراز هویت هر Client بکار میروند که در ابتدای بوجود آمدن هر Client مقدار دهی میشوند. و در فرایند انتفال رمز ارز کاربرد دارند.

```
class Server; // depentent to Client(Class)

class Client
{
public  :
   Client(std::string _id, const Server& _server);
   std::string get_id();
   std::string get_publickey() const;
   double get_wallet();
   std::string sign(std::string txt) const;
   bool transfer_money(std::string receiver, double value);
   size_t generate_nonce();
```

همانطور که در ابتدای کد بالا دیده میشود قبل از تعریف کلاس Client کلاسServer اعلان شده است این عمل در تعریف کلاس Server هم برای کلاس Client انجام شده است همچنین در فایل های cpp. برای هر کلاس فایل h. کلاس دیگر نیز include شده است. این کار به این دلیل است که دو کلاس به هم وابسته هستند و در تعریف هر کدام از دیگری استفاده شده است.

```
Client(std::string _id, const Server& _server);
```

اولین تعریف مربوط به Constructor این کلاس میباشد. در این قسمت متغییر های const یعنی id وserver مقدار دهی میشوند سپس public_key و private_key مقدار دهی میشوند چونکه id و server متغیر های const هستند به صورت delegate مقدار دهی میشوند.

```
std::string get_id();
```

این تابع برای دسترسی به متغییر id از خارج از کلاس تعبیه شده است و Client id را به صورت string برمیگرداند.

```
std::string get_publickey() const;
```

عملکرد این تابع مشابه تابع get_id میباشد و کار دسترسی به متغیر private را برای متغیر public_key انجام میدهد.

```
double get_wallet();
```

این تابع  برای دسترسی به wallet از طریق server برای هر Client میباشد تابع get_wallet در کلاسServer  نیز تعبیه شده است و این تابع از طریق تابع مشابه در کلاس Server به wallet دسترسی پیدا میکند.

```
std::string sign(std::string txt) const;
```

این تابع عمل sign کردن درخواست های Client به Server  را انجام میدهد. این عمل به این دلیل انجام میشود تا Server بتواند از طریق  public_key مطمئن شود درخواست از طریق Client صاحب wallet انجام شده است و نه توسط شخص دیگر. عمل sign کردن با استفاده از private_key و از طریق توابع crypto.h انجام میگیرد.

```
bool Client::transfer_money(std::string receiver, double value)
```

این تابع برای انتقال رمز ارز مورد استفاده قرار میگیرد و هر Client میتواند با استفاده از این تابع به گیرنده مورد نظر رمز ارز انتقال دهد. در این تابع در ابتدا چک میشود که آیا گیرنده جزو Client های موجود در  Server فرستنده میباشد یا خیر در صورت وجود گیرنده درخواست  به فرمت استاندارد نوشته شده و توسط تابع sign که قبلا توضیح داده شد با استفاده از public_key امضا میشود و سپس با استفاده از تابع add_pending_trx که در کلاس Server تعبیه شده است  به متغیرpending_trx  اضافه میشود. در تابع add_pending_trx داشتن وجه کافی در wallet فرستنده و اینکه درخواست توسط خود فرستنده به server فرستاده شده باشد بررسی میشود. اگر درخواست به صورت موفقیت آمیز به pending_trx اضافه شد خروجی تابع true خواهد بود.

## size_t Client::generate_nonce()

این تابع به Client این امکان را میدهد که یک nonce برای استخراج رمز ارز تولید کند و از این طریق رمز ارز استخراج کند. از این تابع در تابع mine که در کلاس Server تعبیه شده است استفاده میشود. هر nonce یک عدد رندوم از نوع unsigned long int میباشد.

## class Server

در پیاده سازی ما سرور نقش های مختلفی را دارد مانند نظارت بر Client ها و wallet های آنها، انجام تراکنش ها ، استخراج رمز ارز (mining) و... . در ادامه به بررسی متغیر ها و توابع پیاده شده در این کلاس میپردازیم.

```
private :
  std::map<std::shared_ptr<Client>, double> clients;
```

این متغیر یک map میان pointer به هر Client و مقدار wallet هر Client میباشد که دسترسی به آن فقط از داخل Server امکان پذیر میباشد.

```
class Client; // dependent to Server(Class)

extern std::vector<std::string> pending_trxs; // prevent from redefine

class Server
{
public :

    Server();
    std::shared_ptr<Client> add_client(std::string id);
    std::shared_ptr<Client> get_client(std::string id) const;
    double get_wallet(std::string id) const;
    static bool parse_trx(std::string trx, std::string &sender, std::string
&receiver, double &value) ;
    bool add_pending_trx(std::string trx, std::string signature) const;
    size_t mine();
    friend void  show_wallets(const  Server& server);
```

همانطور که دیده میشود در بالای تعریف این کلاس نیز کلاس دیگر تعریف شده است نکته دیگر واژه extern در پشت تعریف pending_trx میباشد این واژه برای جلوگیری از دوباره تعریف شدن این متغیر در اینجا قرار داده شده است به این صورت کامپایلر میفهمد که این متغیر در جای دیگر تعریف میشود و هنگامی که در لینک کردن فایل های h. به این تعریف میرسد خطای redefinition را برنمیگرداند.

```
Server();
```

Constructor این کلاس تعریف خاصی ندارد و آرگومانی را به عنوان ورودی نمیگیرد و متغیر خاصی را هم مقدار دهی نمیکند. زیرا تنها متغیر کلاس یعنی clients متغیری const نمیباشد و میتواند در جاهای دیگر نیز مقدار دهی شود.

```
  std::shared_ptr<Client> add_client(std::string id);
```

این تابع برای اضافه کردن یکClient به Server میباشد. با استفاده از تابع get_client که در ادامه توضیح داده میشود .چک میکند که Client با id ورودی وجود نداشته باشد در صورت وجود یک عدد رندوم 4 رقمی را به انتهای id وارد شده اضافه میکند. پس از این مرحله یک Client با id یونیک که Client درست میکند. سپس مپ پوینتر به این Client وwallet ش را به متغیر clients میافزاید. هر Client در ابتدا 5 رمز ارز در wallet خود دارد. پس از انجام این مراحل اشاره گر به این Client را به عنوان خروجی برمیگرداند.

```
std::shared_ptr<Client> get_client(std::string id) const;
```

این تابع به سرور اجازه میدهد که با داشتن Client id هایی که در این سرور هستند به اشاره گر به آنها دسترسی داشته باشد این عمل به این گونه انجام میشود که در متغیر clients جست و جو میشود تا Client با id خواسته شده پیدا شود و سپس اشاره گر به آن به عنوان خروجی برگردانده میشود. در پیاده سازی این تابع از تابع get_id پیاده سازی شده در

کلاس Client استفاده شده است زیرا دسترسی به id هر Client که یک متغیر private است از بیرون کلاس ممکن نیست. اگر Client با id خواسته شده در متغیر clients پیدا نشود nullptr به عنوان خروجی برگردانده میشود.

```
double Server :: get_wallet(std::string id) const
```

این تابع برای گرفتن مقدار wallet هر Client مورد استفاده قرار میگیرد. در درون کلاس Server دسترسی به wallet هر Client مقدور میباشد اما پیاده سازی این تابع به تکرار نشدن کد کمک میکند. همچنین این تابع در get_wallet تعبیه شده در کلاس Client نیز استفاده شده است و این امکان را میدهد که هر Client با فراخوانی get_wallet کلاس خود و ارتباط با Server خود به مقدار wallet خودش دسترسی پیدا کند.پیاده سازی آن نیز به این صورت است که در متغیر Client clients با id ورودی پیدا میشود و مقدار wallet آن به صورت خروجی برگردانده میشود.

```
static bool parse_trx(std::string trx, std::string &sender, std::string
&receiver, double &value) ;
```

این تابع تراکنش به فرمت استاندارد گفته شده را گرفته و چک میکند که به فرم استاندارد باشد و سپس فرستنده گیرنده و مبلغ تراکنش را به عنوان خروجی برمیگرداند خروجی های این تابع به این شکل داده میشوند که متغیر ها به صورت رفرنس به تابع داده شده و تابع آنها را با مقادیر مورد نظر مقدار دهی میکند. اگر تراکنش به فرمت درست و معتبر باشد مقدار true و در غیر این صورت مقدار false برگردانده میشود. نکته قابل توجه در این تابع static بودن آن است یعنی تابع در خارج از کلاس هم قابلیت فراخوانی دارد و scope تعریف کلاس تنها مربوط به داخل کلاس نمیباشد. و این امکان را دارد که از خارج کلاس نیز فراخوانی شود. پیاده سازی تابع نیز به این شکل میباشد که ابتدا sender, receiver و value استخراج شده و سپس درستی فرمت تراکنش چک میشود پس از آن درست بودن فرمت value چک میشود و در صورت درست بودن آن مقدار درست برگردانده میشود.

در استخراج موارد گفته شده از کتابخانه <sstream > استفاده شده است.

```
bool Server::add_pending_trx(std::string trx, std::string signature) const
```

این تابع برای افزودن یک تراکنش به متغیر pending_trx استفاده میشود. و به نوعی وظیفه آن مطمئن شدن از درست بودن تراکنش است. این تابع در ابتدا با استفاده از تابع parse_trx فرستنده گیرنده و مبلغ تراکنش را از تراکنش استخراج میکند. سپس public_key فرستنده با استفاده از id استخراج شده و معتبر بودن تراکنش با استفاده از آن انجام میشود اگر تراکنش تایید شود که از سوی فرستنده بوده و همچنین فرستنده حداقل مبلغی برابر با مبلغ تراکنش را در wallet خود داشته باشد تراکنش به متغیر pending_trx اضافه میشود و مقدار true برگردانده میشود در غیر اینصورت مقدار false برگردانده میشود.

```
size_t Server::mine()
```

این تابع عمل استخراج رمز ارز را انجام میدهد.در این تابع تراکنش های موجود در pending_trx اثر داده شده و همچنین از pending_trx پاک میشوند. همچنین به Clientای که تراکنش هارا تایید کند مقدار 6.25 رمز ارز جایزه داده میشود. پیاده سازی این تابع به این صورت است که ابتدا mempool که حاصل چسباندن هر کدام از تراکنش های موجود در pending_trx میباشد درست میشود. سپس از هر Client موجود در Server یک nonce که یک عدد رندوم از نوع unsigned long int میباشد گرفته شده و به انتهای mempool اضافه میشود. سپس string حاصل با hash Sha256 میشود و اگر hash حاصل دارای سه صفر متوالی در 10 کاراکتر اول خود باشد mineing موفقیت آمیز بوده و به miner 6.25 از رمز ارز داده میشود. پس از ان تمام تراکنش ها اثر داده شده و سپس از pending_trx پاک میشوند.

# نکات مهم

در سراسر تعریف توابع در دو کلاس کلمه const در جلوی اعلان توابع به چشم میخورد این عمل به کامپایلر میفهماند که این توابع میتوانند بر روی Instance هایی از کلاس که const هستند فراخوانی شوند و همچنین توابعی که درون این توابع فراخوانی میشوند نیز چنین قابلیتی را دارند در پیاده سازی این پروژه هنگام فراخوانی توابع Server از درون یک Client چونکه متغیر server از نوع const بود باید این نکته رعایت میشد تا بتوان توابع پیاده سازی شده را از طریق server فراخوانی کرد.

تابع show_wallet که وظیفه نمایش هر Client به همراه مقدار walletش را دارد در خارج از کلاس Server تعریف میشود اما در تعریف آن از متغیرclients که یک متغیر private در کلاس Server میباشد استفاده شده است. برای اینکه این تابع به این متغیر دسترسی داشته باشد باید این تابع با کلاس friend Server شود در این صورت این تابع به متغیر های private کلاس Server دسترسی خواهد داشت.

[لینک گیت هاب]