

BITCOIN

SOFTWARE ARCHITECTURE & DESIGN

GROUP-3





TOPICS

What will we present today.

01

BITCOIN Architecture

02

BITCOIN Design Pattern

BITCOIN คืออะไร?

เป็นสกุลเงินดิจิทัล ประเภท Cryptocurrency
ที่ใช้เทคโนโลยี Blockchain มาใช้ในการกำกับธรรม
การเงิน โดยเป็นครั้งแรก จากทั่วทุกมุมโลก

โดย Bitcoin core คือชื่อ Open source
software ที่ทำให้ Bitcoin สามารถใช้งานได้





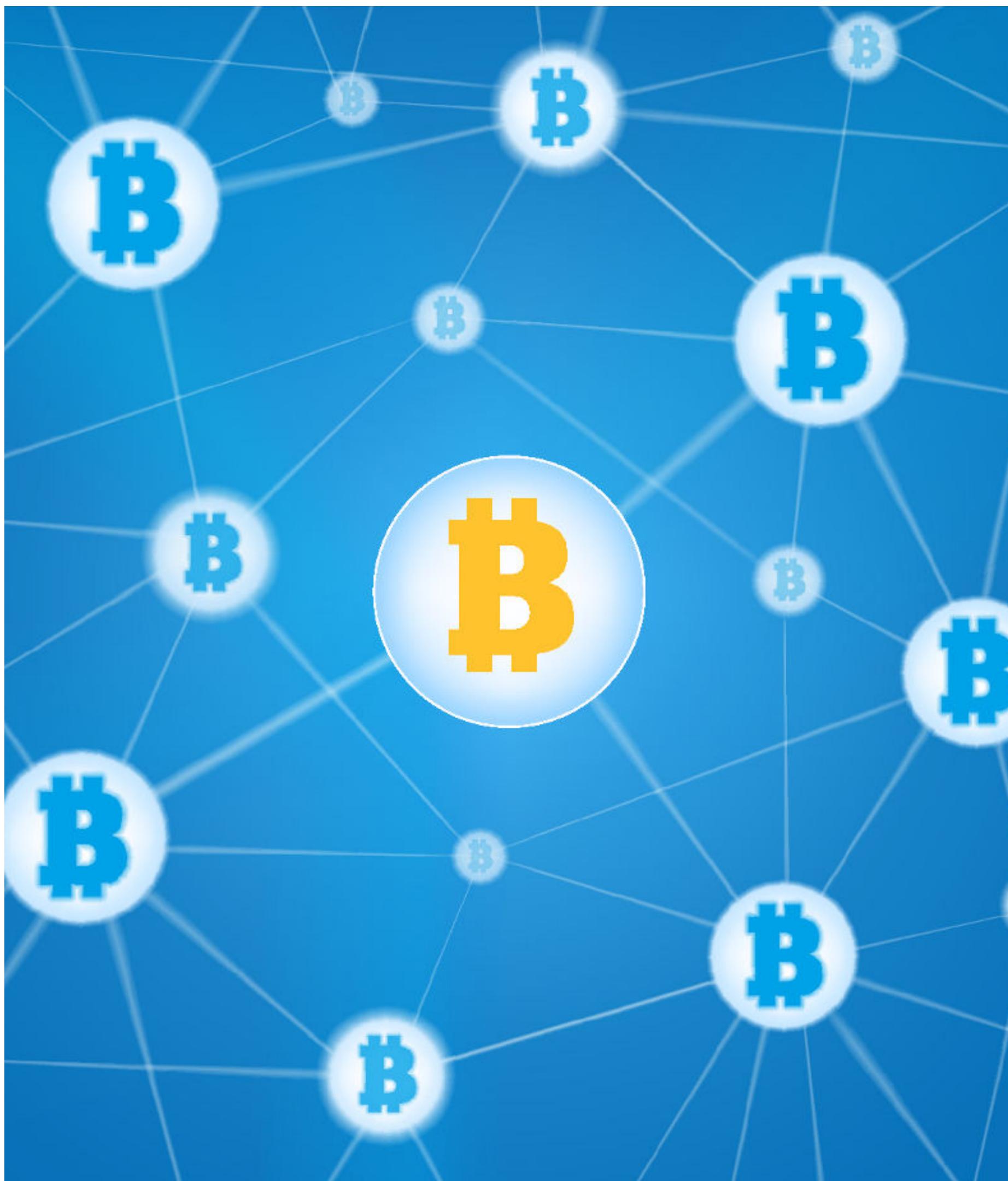
ACT I

BITCOIN ARCHITECTURE

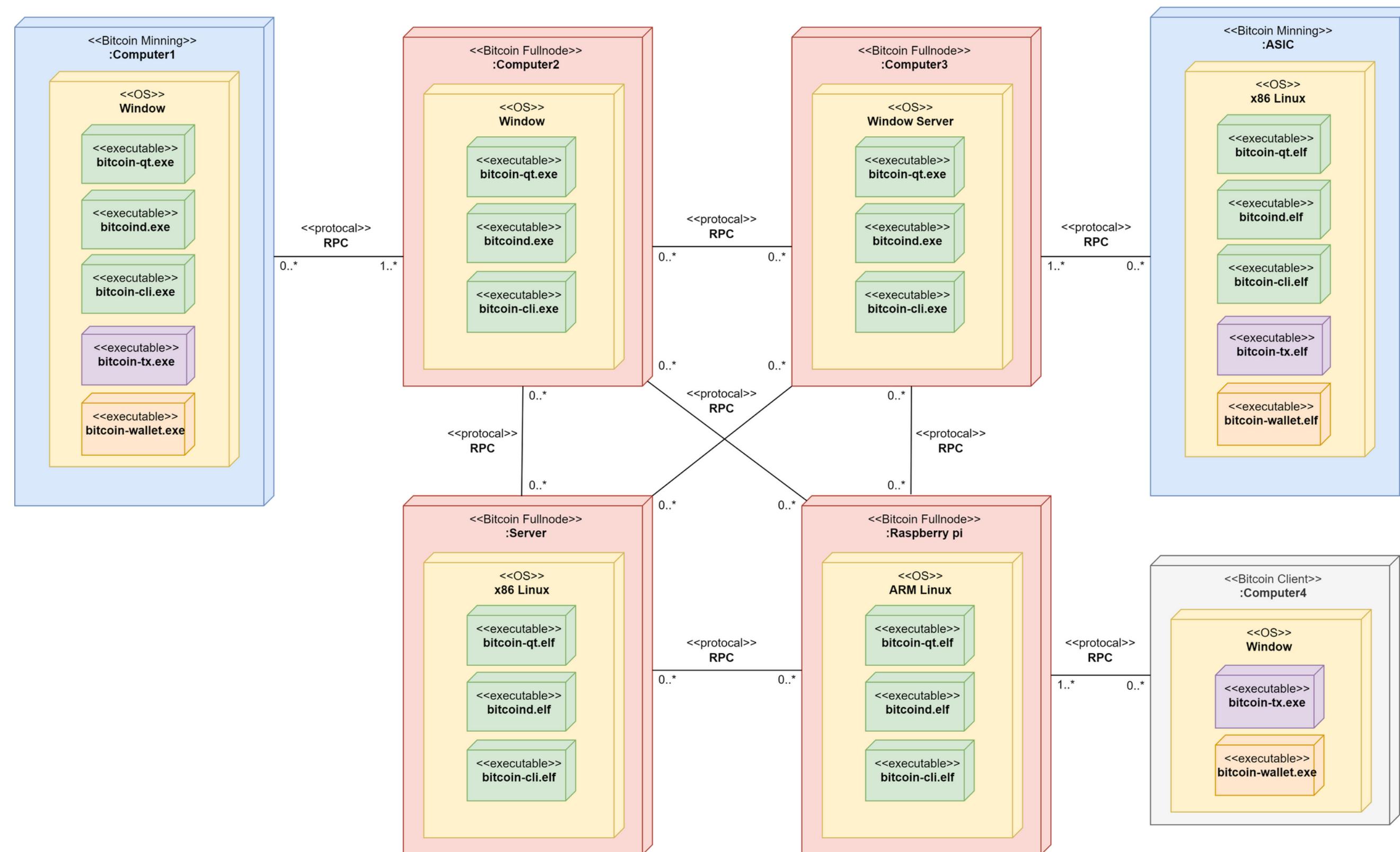
Architecture

Peer-to-Peer

Bitcoin ใช้เทคโนโลยี Blockchain
ในการกำกับธรรม ซึ่งทุกๆ ส่วนจะต้องเชื่อม
ต่อกันและกันในแต่ละ Node โดยสามารถ
ส่งข้อมูลหากันและกันได้



Bitcoin Peer-to-Peer Deployment Diagram





Quality Attributes

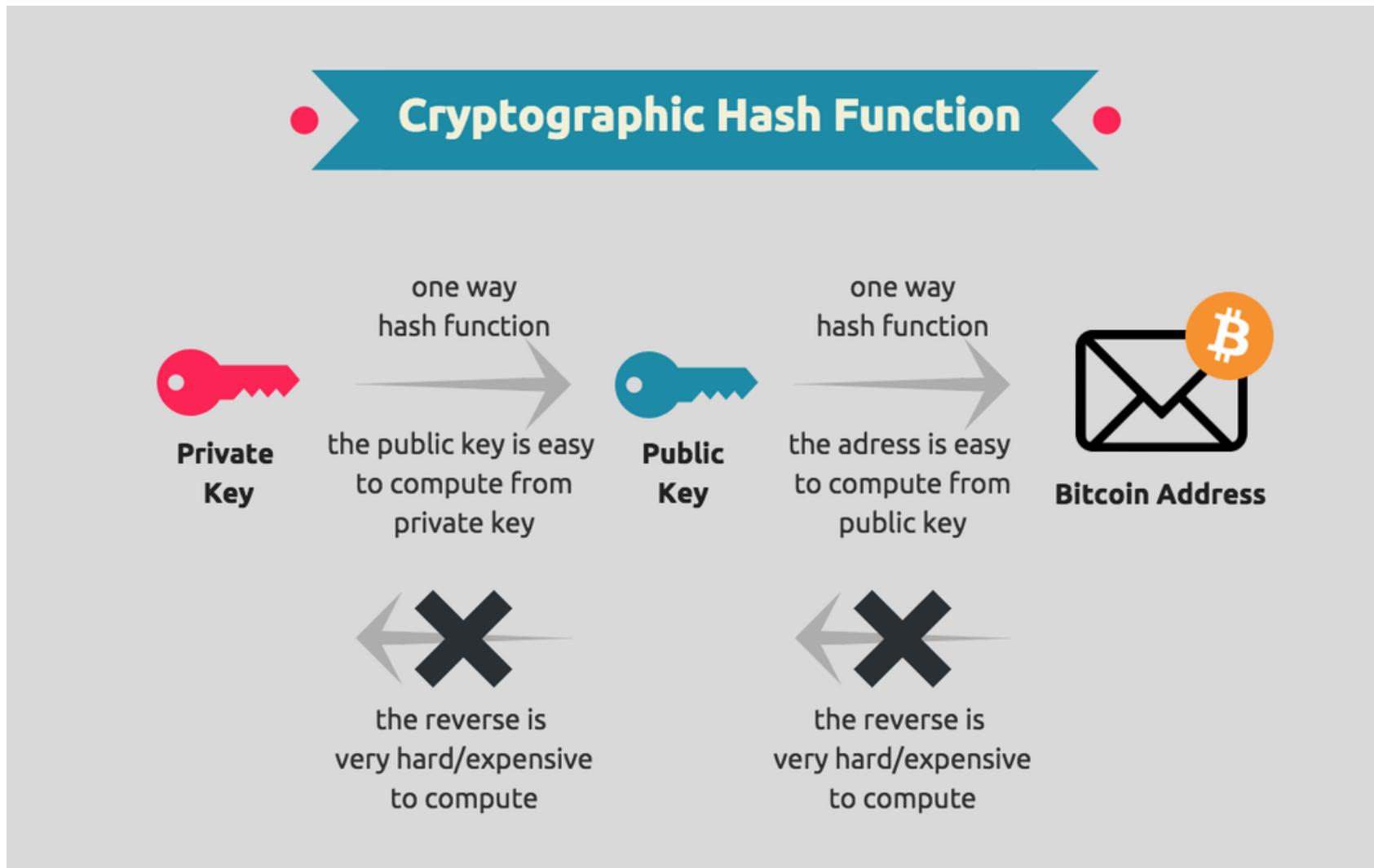
Bitcoin project

1. Security



เนื้องจากข้อมูลบน Bitcoin network ไม่สามารถที่จะแก้ไขได้ หรือถ้าจะแก้ไขก็จำเป็นต้องใช้กรัพยากรที่สูงมาก จนอาจจะไม่คุ้มค่าที่จะแก้ไข Bitcoin มี Network ที่ใหญ่มากๆ ใช้งานอยู่ทั่วโลก ทำให้ยากต่อการที่จะมีกลุ่มคนใดกลุ่มคนนึง ช่วยกันแก้ไขข้อมูลบน Bitcoin network ได้ จึงทำให้ Bitcoin มีความปลอดภัยที่สูงมาก

การเพิ่ม Security ของ Bitcoin จะทำได้โดยการมอบ Permission ที่ Node สามารถทำได้ ทำให้ Node มีข้อจำกัดในการเข้าถึงข้อมูลและไม่สามารถเข้าถึงข้อมูลอย่างอิสระได้



1. Security

" How is Bitcoin secure? Bitcoin is backed by a special system called the blockchain. Compared to other financial solutions, the blockchain is an improved technology that relies on secure core concepts and cryptography.

Blockchain uses volunteers – lots of them – to sign hashes that validate transactions on the Bitcoin network using cryptography. This system makes it so transactions are generally irreversible, and the data security of Bitcoin is strong. "

<https://blockchainhub.net/blog/blog/cryptography-blockchain-bitcoin/>

<https://www.avg.com/en/signal/is-bitcoin-safe>

<https://bitcoin.org/en/faq#security>

2. Availability



Bitcoin ทำงานอยู่บนระบบ decentralized blockchain ซึ่งเป็นการกระจายการทำงานให้ผู้ใช้งาน Bitcoin ต่างจากระบบธนาคารในปัจจุบันที่ธนาคารเป็นผู้ดูแลทุกอย่าง ซึ่งเป็นแนวคิดแบบ centralized โดยข้อเสียของแนวคิดนี้คือ ถ้าวันหนึ่งระบบมีปัญหา ทุกคนก็จะไม่สามารถใช้งานอะไรได้เลย ต่างจาก Bitcoin ที่มี Availability ที่สูงมาก

เนื่องจากมี node ของ Bitcoin ทำงานอยู่ทั่วโลก ซึ่งแต่ละ Node ติดต่อกันด้วยเครือข่ายแบบ Peer to Peer ทำให้ข้อมูลบน network สามารถเข้าถึงได้ตลอดเวลาจากทั่วทุกที่ โดยแค่ต้องติดต่อเพียง node ใด node หนึ่งของ Bitcoin เท่านั้น ก็สามารถเชื่อมต่อกันได้ อีกทั้งยังมีความปลอดภัยของข้อมูลในการเข้ารหัส ทำให้ bitcoin มีความโปร่งใสและน่าเชื่อถือสูงมาก



Bitcoin.org

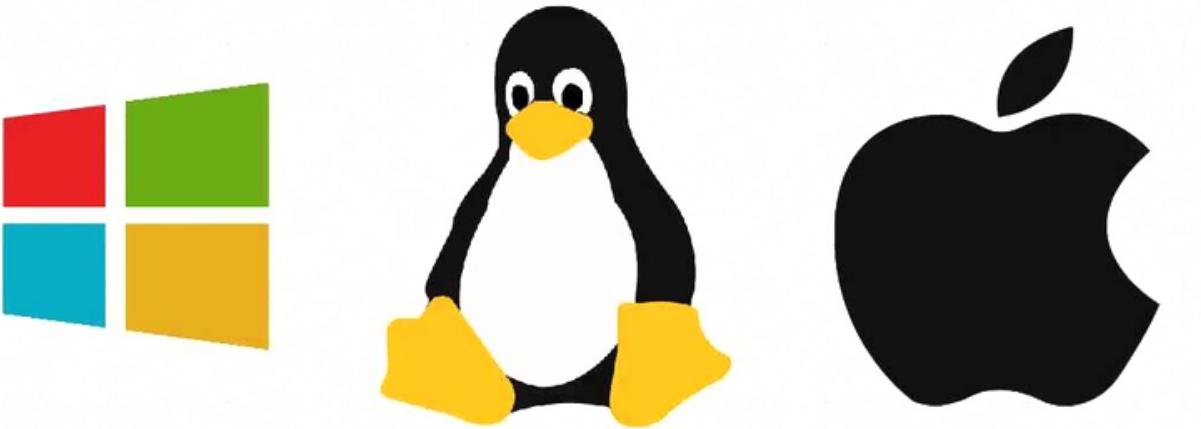
Official Bitcoin website

<https://bitcoin.org/en/faq#what-are-the-advantages-of-bitcoin>

2. Availability

" Transparent and neutral - All information concerning the Bitcoin money supply itself is readily available on the block chain for anybody to verify and use in real-time. No individual or organization can control or manipulate the Bitcoin protocol because it is cryptographically secure. This allows the core of Bitcoin to be trusted for being completely neutral, transparent and predictable."

3. Portability



สำหรับตัว Bitcoin Core ได้มีการทำการ Compile ตัวโปรเจค Bitcoin ไว้รองรับทุกสถาปัตยกรรมของ Cpu ทั้ง Arm และ x86 อีกทั้งยังรองรับการทำงานแบบ 32-bit และ 64-bit

ดังนั้น Bitcoin Core จึงสามารถนำไปทำงานทั้ง ระบบ Windows, OS X และ ระบบปฏิบัติการ Linux หลายๆตัวที่ได้รับความนิยมได้ นอกจากนี้ผู้ใช้ยังสามารถนำ Bitcoin Core ไป compile สำหรับ ระบบปฏิบัติการ Linux ที่ตัวเองต้องการจะติดตั้งก็ได้เช่นกัน

ในโปรเจค Bitcoin นี้ ได้มีการพัฒนาโดยใช้ภาษา C/C++ ซึ่งเป็น ภาษาที่รองรับสำหรับ Compilers ที่สามารถรันได้บน Platforms ต่างๆมากมายที่มีการรองรับตัวภาษานี้อยู่ โดยใช้ standard library C/C++ เพื่อที่จะทำให้สามารถรันบน Platforms ได้ก็ได้ โดยไม่ต้อง แก้ไขโค้ดเลยหรือมีการแก้ไขเพียงเล็กน้อยเท่านั้น

Latest version: 22.0 



Download Bitcoin Core

Or choose your operating system



Windows
exe - zip



ARM Linux
64 bit - 32 bit



Mac OS X
dmg - tar.gz



RISC-V Linux
64 bit



Linux (tgz)



PPC64 Linux
64 bit - 64 bit LE



Snap Store Linux

[SHA256 binary hashes](#)

[SHA256 hash signatures](#)

[Download torrent](#) 

[Source code](#)

[Show version history](#)

Refresh expired keys using:

```
gpg --keyserver https://keys.openpgp.org --refresh-keys
```

<https://bitcoincore.org/en/download/>

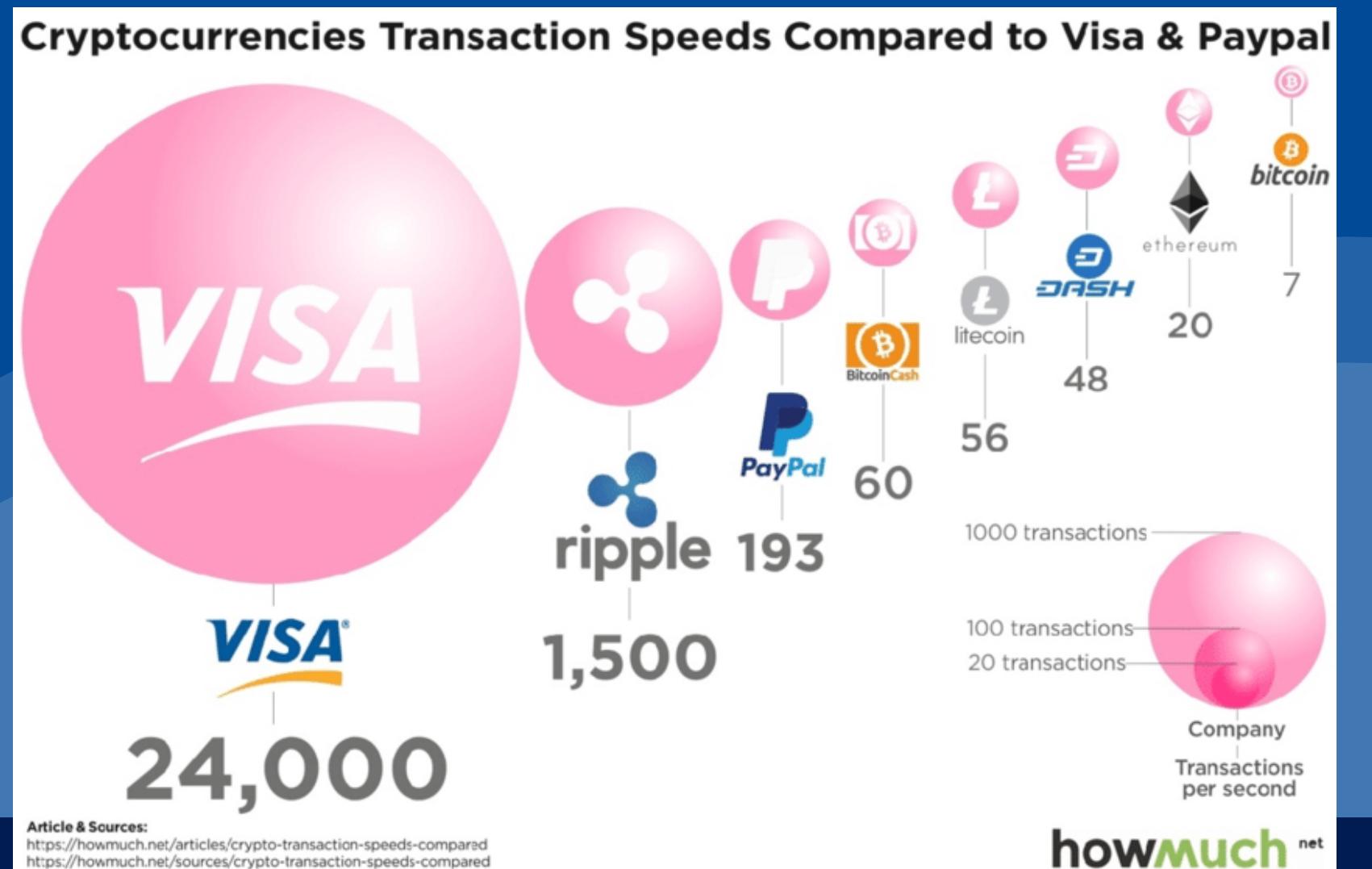
3. Portability

" Desktop or laptop hardware running recent versions of Windows, Mac OS X, or Linux.

" C++ is one of the most frequently used languages in the world and as an open language, C++ has a wide range of compilers that run on many different platforms that support it.

Code that exclusively uses C++'s standard library will run on many platforms with few to no changes."

จุดอ่อน



ปริมาณของผู้ใช้งาน

จุดอ่อนหลักคือปริมาณผู้ใช้งานที่เข้าถึงระบบ ON-CHAIN ได้อย่างจำกัด ทำให้การกำกับดูแลเกิดขึ้นได้ช้า

วิธีแก้ไข

ได้มีการสร้าง OFF-CHAIN SOLUTION ขึ้นมา และยังมี
การทำ Private Blockchain Framework ซึ่งสามารถช่วยลด
Traffic ใน Network ได้



BITCOIN

Design Pattern



Design Patterns

IN BITCOIN

01

BUILDER

02

SINGLETON

03

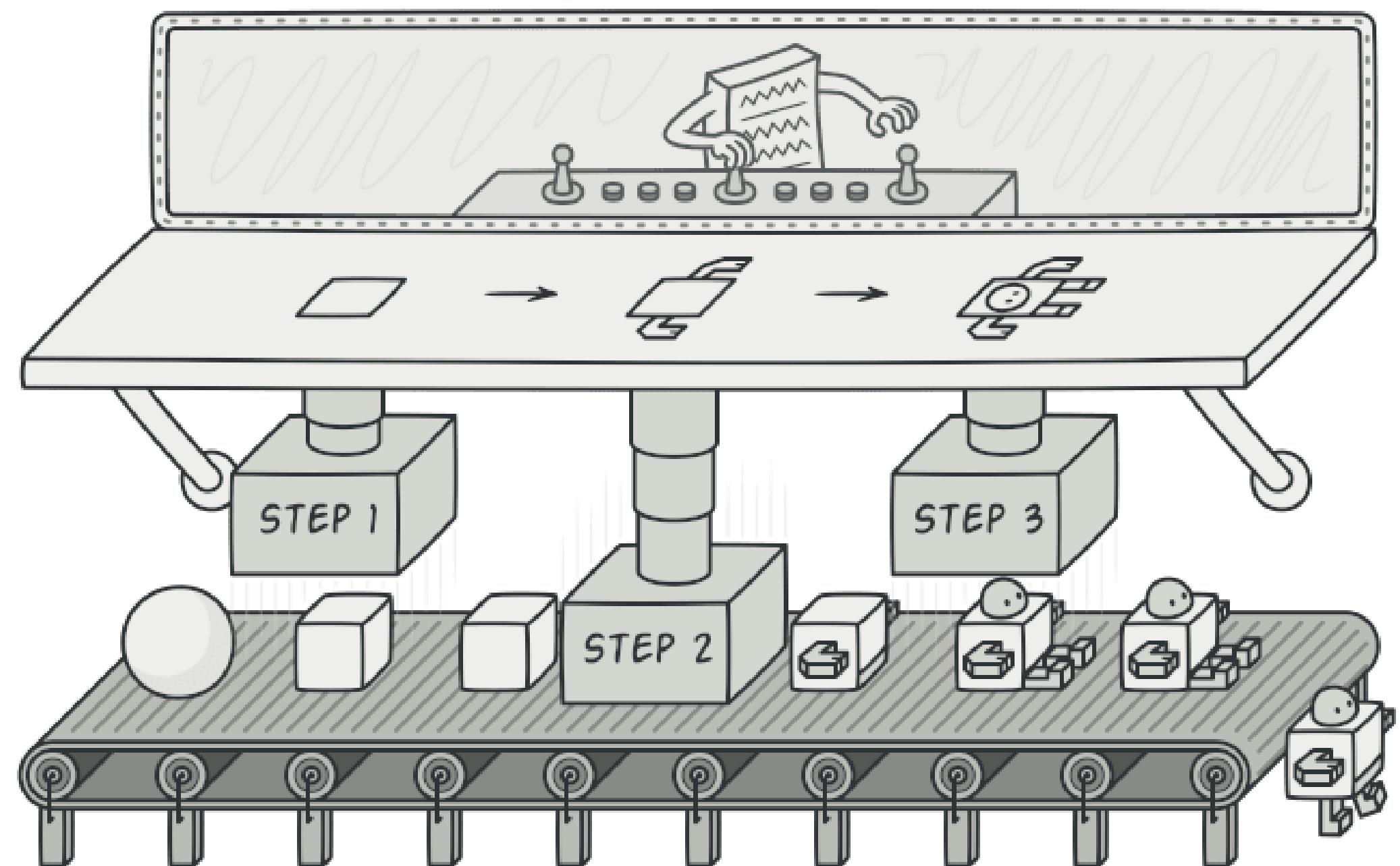
ITERATOR

04

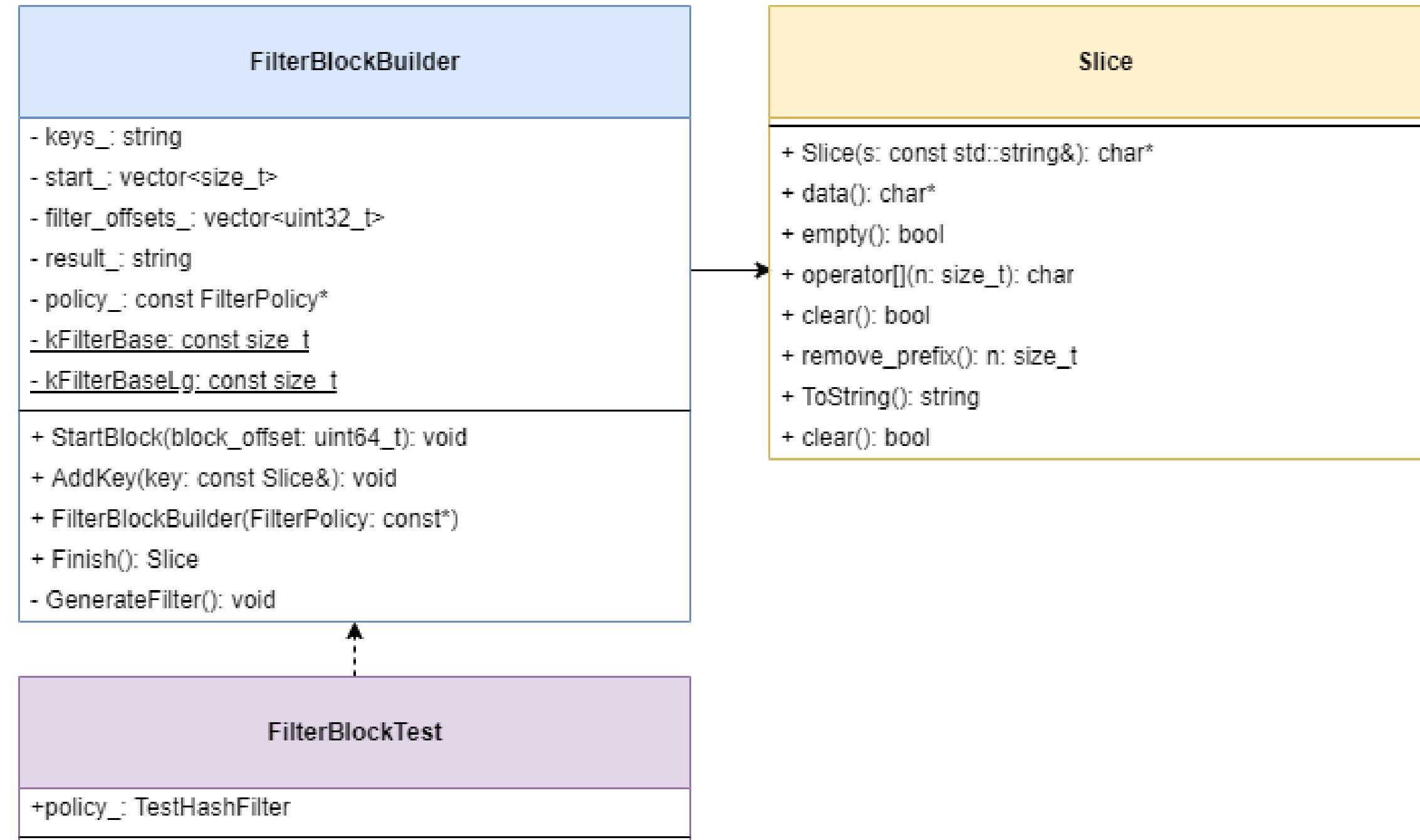
NULL OBJECT



Builder



Builder



Builder

```
● ● ●

1 class FilterBlockBuilder {
2 public:
3     explicit FilterBlockBuilder(const FilterPolicy* );
4
5     FilterBlockBuilder(const FilterBlockBuilder&) = delete;
6     FilterBlockBuilder& operator=(const FilterBlockBuilder&) = delete;
7
8     void StartBlock(uint64_t block_offset);
9     void AddKey(const Slice& key);
10    Slice Finish();
11
12 private:
13     void GenerateFilter();
14
15     const FilterPolicy* policy_;
16     std::string keys_;           // Flattened key contents
17     std::vector<size_t> start_; // Starting index in keys_ of each key
18     std::string result_;        // Filter data computed so far
19     std::vector<Slice> tmp_keys_; // policy_->CreateFilter() argument
20     std::vector<uint32_t> filter_offsets_;
21 }
```

```
● ● ●

1 TEST(FilterBlockTest, SingleChunk) {
2     FilterBlockBuilder builder(&policy_);
3     builder.StartBlock(100);
4     builder.AddKey("foo");
5     builder.AddKey("bar");
6     builder.AddKey("box");
7     builder.StartBlock(200);
8     builder.AddKey("box");
9     builder.StartBlock(300);
10    builder.AddKey("hello");
11    Slice block = builder.Finish();
12    FilterBlockReader reader(&policy_, block);
13    ASSERT_TRUE(reader.KeyMayMatch(100, "foo"));
14    ASSERT_TRUE(reader.KeyMayMatch(100, "bar"));
15    ASSERT_TRUE(reader.KeyMayMatch(100, "box"));
16    ASSERT_TRUE(reader.KeyMayMatch(100, "hello"));
17    ASSERT_TRUE(reader.KeyMayMatch(100, "foo"));
18    ASSERT_TRUE(!reader.KeyMayMatch(100, "missing"));
19    ASSERT_TRUE(!reader.KeyMayMatch(100, "other"));
20 }
```

filter_block.h (24-102)

https://github.com/bitcoin/bitcoin/blob/7fcf53f7b4524572d1d0c9a5fdc388e87eb02416/src/leveldb/table/filter_block.h

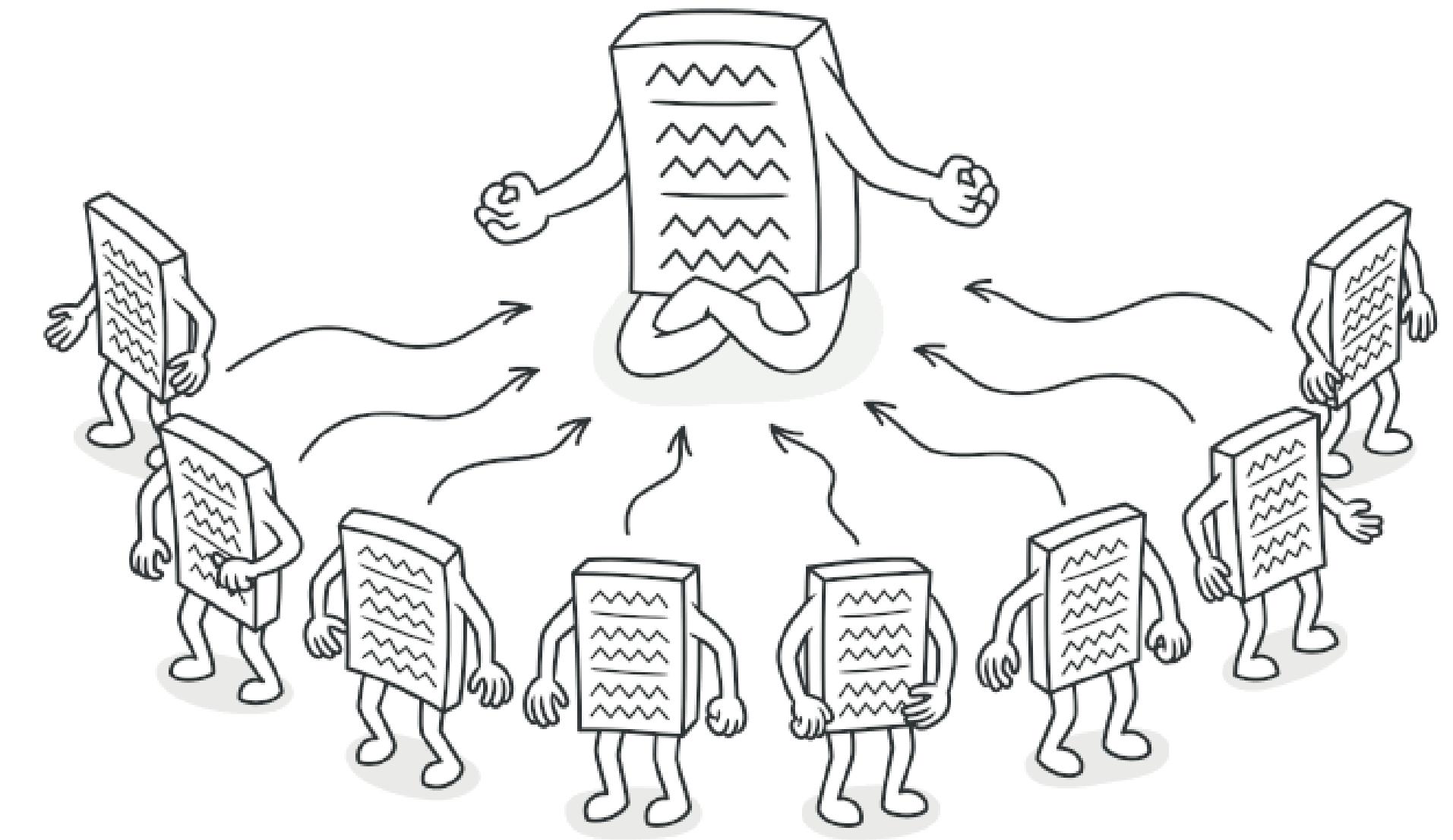
filter_block_test.cc (53-72)

https://github.com/bitcoin/bitcoin/blob/7fcf53f7b4524572d1d0c9a5fdc388e87eb02416/src/leveldb/table/filter_block_test.cc

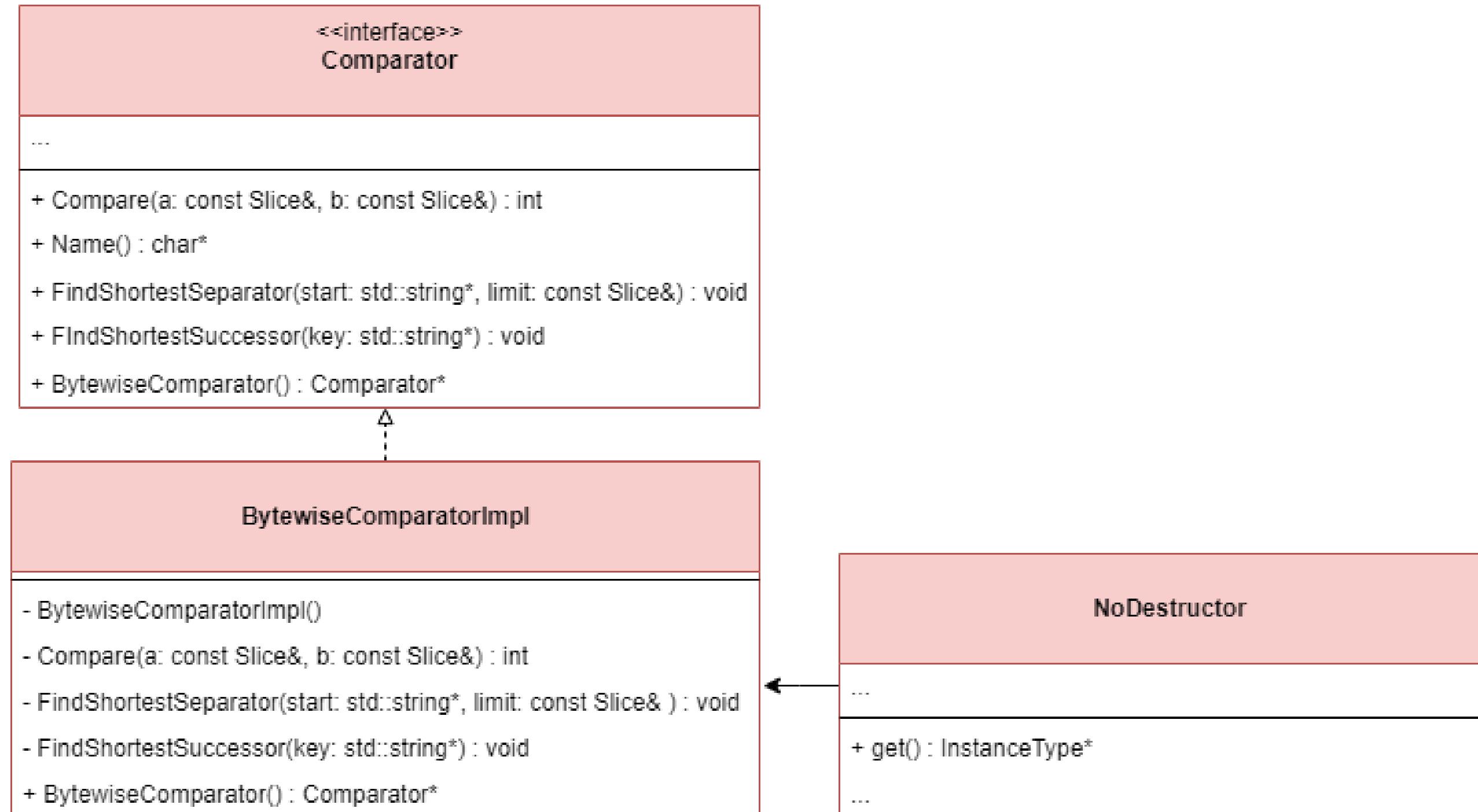
slice.h (28-92)

<https://github.com/bitcoin/bitcoin/blob/7fcf53f7b4524572d1d0c9a5fdc388e87eb02416/src/leveldb/include/leveldb/slice.h>

Singleton



Singleton



Singleton

```
1  namespace leveldb {
2
3      Comparator::~Comparator() = default;
4
5      namespace {
6          class BytewiseComparatorImpl : public Comparator {
7              public:
8                  BytewiseComparatorImpl() = default;
9
10             const char* Name() const override { return "leveldb.BytewiseComparator"; }
11
12             int Compare(const Slice& a, const Slice& b) const override {
13                 return a.compare(b);
14             }
15
16             void FindShortestSeparator(std::string* start,
17                                         const Slice& limit) const override {
18                 // Find length of common prefix
19                 size_t min_length = std::min(start->size(), limit.size());
20                 size_t diff_index = 0;
21                 while ((diff_index < min_length) &&
22                         ((*start)[diff_index] == limit[diff_index])) {
23                     diff_index++;
24                 }
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54             const Comparator* BytewiseComparator() {
55                 static NoDestructor<BytewiseComparatorImpl> singleton;
56                 return singleton.get();
57             }
58         }
59     } // namespace Leveldb
60 } // namespace Leveldb
```

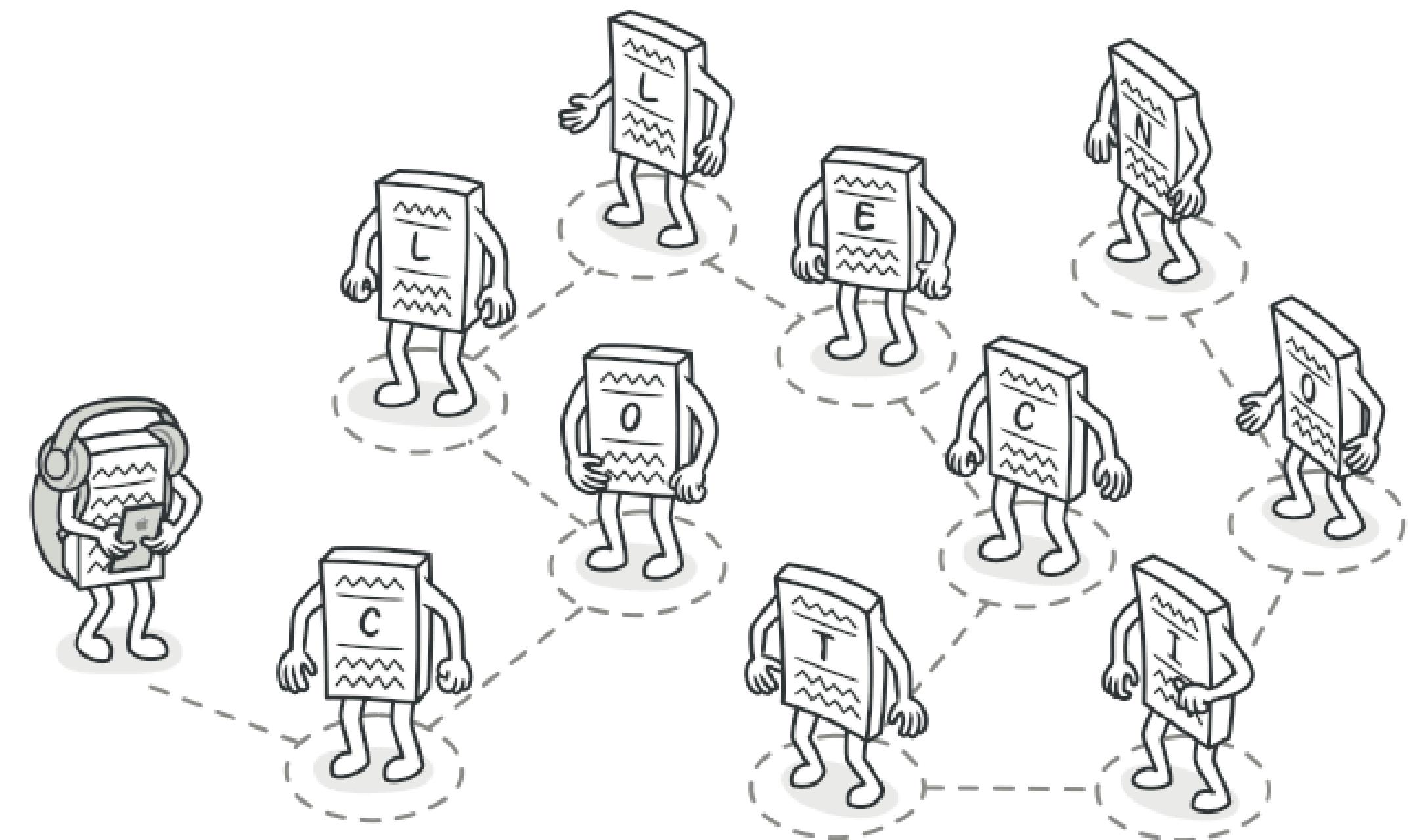
```
1  namespace leveldb {
2
3      // Wraps an instance whose destructor is never called.
4      //
5      // This is intended for use with function-level static variables.
6      template <typename InstanceType>
7      class NoDestructor {
8          public:
9              template <typename... ConstructorArgTypes>
10                 explicit NoDestructor(ConstructorArgTypes&&... constructor_args) {
11                     static_assert(sizeof(instance_storage_) >= sizeof(InstanceType),
12                             "instance_storage_ is not large enough to hold the instance");
13                     static_assert(
14                         alignof(decltype(instance_storage_)) >= alignof(InstanceType),
15                         "instance_storage_ does not meet the instance's alignment requirement");
16                     new (&instance_storage_)
17                         InstanceType(std::forward<ConstructorArgTypes>(constructor_args)...);
18                 }
19
20                 ~NoDestructor() = default;
21
22                 NoDestructor(const NoDestructor&) = delete;
23                 NoDestructor& operator=(const NoDestructor&) = delete;
24
25                 InstanceType* get() {
26                     return reinterpret_cast<InstanceType*>(&instance_storage_);
27                 }
28
29             private:
30                 typename std::aligned_storage<sizeof(InstanceType),
31                                         alignof(InstanceType)>::type instance_storage_;
32             };
33
34 } // namespace Leveldb
```

comparator.cc (70-73) <https://github.com/bitcoin/bitcoin/blob/7fcf53f7b4524572d1d0c9a5fdc388e87eb02416/src/leveldb/util/comparator.cc>

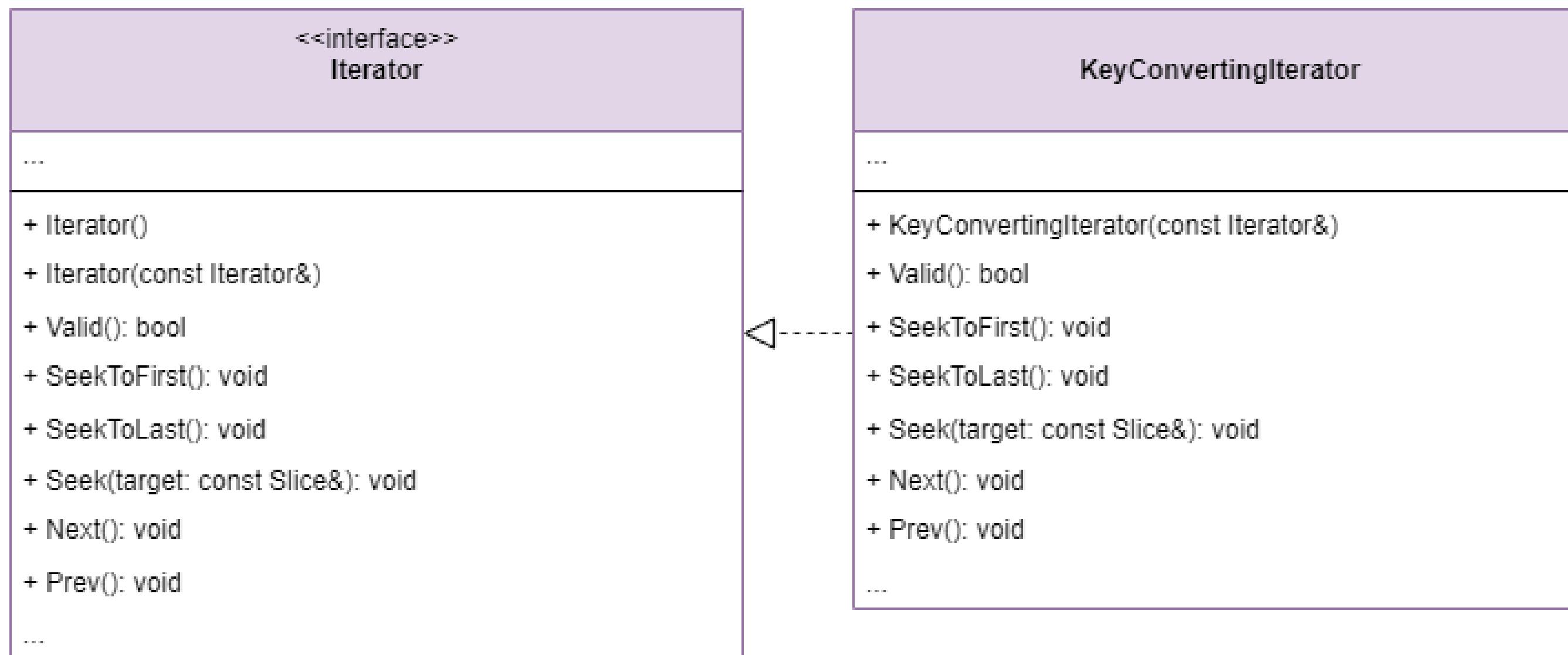
no_destructor.h (17-42) https://github.com/bitcoin/bitcoin/blob/7fcf53f7b4524572d1d0c9a5fdc388e87eb02416/src/leveldb/util/no_destructor.h

comparator.h (20-55) <https://github.com/bitcoin/bitcoin/blob/7fcf53f7b4524572d1d0c9a5fdc388e87eb02416/src/leveldb/include/leveldb/comparator.h>

Iterator



Iterator



Iterator

```
● ○ ●
1 class LEVELDB_EXPORT Iterator {
2 public:
3     Iterator();
4
5     Iterator(const Iterator&) = delete;
6     Iterator& operator=(const Iterator&) = delete;
7
8     virtual ~Iterator();
9     virtual bool Valid() const = 0;
10    virtual void SeekToFirst() = 0;
11    virtual void SeekToLast() = 0;
12    virtual void Seek(const Slice& target) = 0;
13    virtual void Next() = 0;
14    virtual void Prev() = 0;
15    virtual Slice key() const = 0;
16    virtual Slice value() const = 0;
17    virtual Status status() const = 0;
18    using CleanupFunction = void (*)(void* arg1, void* arg2);
19    void RegisterCleanup(CleanupFunction function, void* arg1, void* arg2);
```

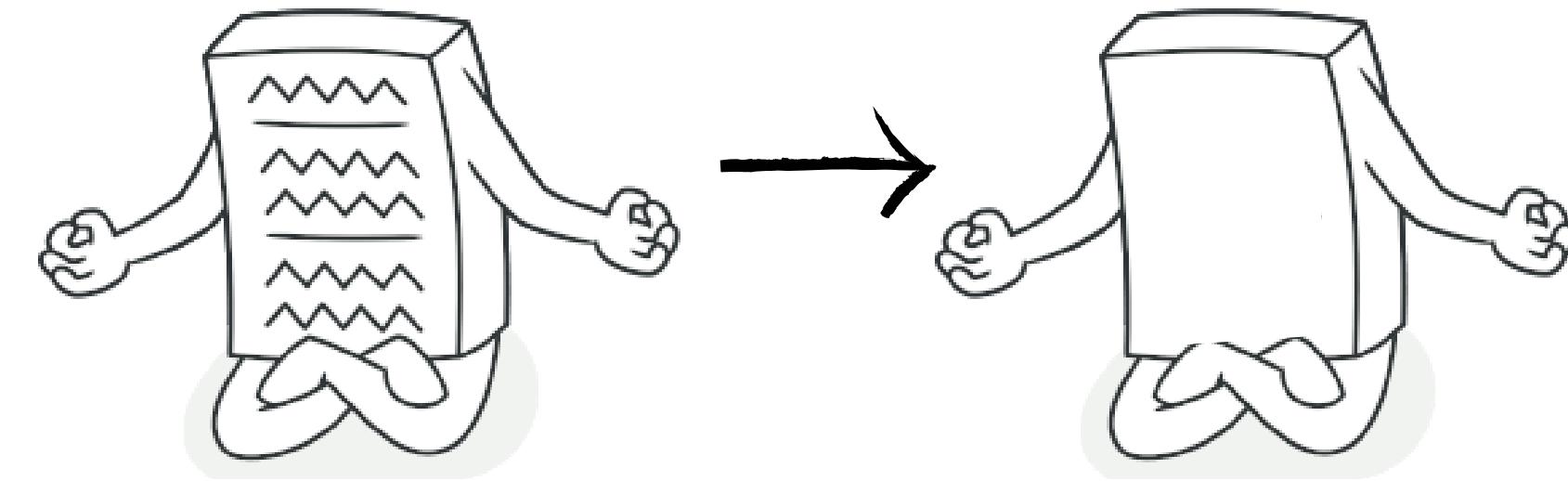
```
● ○ ●
1 class KeyConvertingIterator : public Iterator {
2 public:
3     explicit KeyConvertingIterator(Iterator* iter) : iter_(iter) {}
4
5     KeyConvertingIterator(const KeyConvertingIterator&) = delete;
6     KeyConvertingIterator& operator=(const KeyConvertingIterator&) = delete;
7
8     ~KeyConvertingIterator() override { delete iter_; }
9
10    bool Valid() const override { return iter_->Valid(); }
11    void Seek(const Slice& target) override {
12        ParsedInternalKey ikey(target, kMaxSequenceNumber, kTypeValue);
13        std::string encoded;
14        AppendInternalKey(&encoded, ikey);
15        iter_->Seek(encoded);
16    }
17    void SeekToFirst() override { iter_->SeekToFirst(); }
18    void SeekToLast() override { iter_->SeekToLast(); }
19    void Next() override { iter_->Next(); }
20    void Prev() override { iter_->Prev(); }
21
22    Slice key() const override {
23        assert(Valid());
24        ParsedInternalKey key;
25        if (!ParseInternalKey(iter_->key(), &key)) {
26            status_ = Status::Corruption("malformed internal key");
27            return Slice("corrupted key");
28        }
29        return key.user_key;
30    }
31
32    Slice value() const override { return iter_->value(); }
33    Status status() const override {
34        return status_.ok() ? iter_->status() : status_;
35    }
36
37 private:
38     mutable Status status_;
39     Iterator* iter_;
40};
```

iterator.h (24-81)

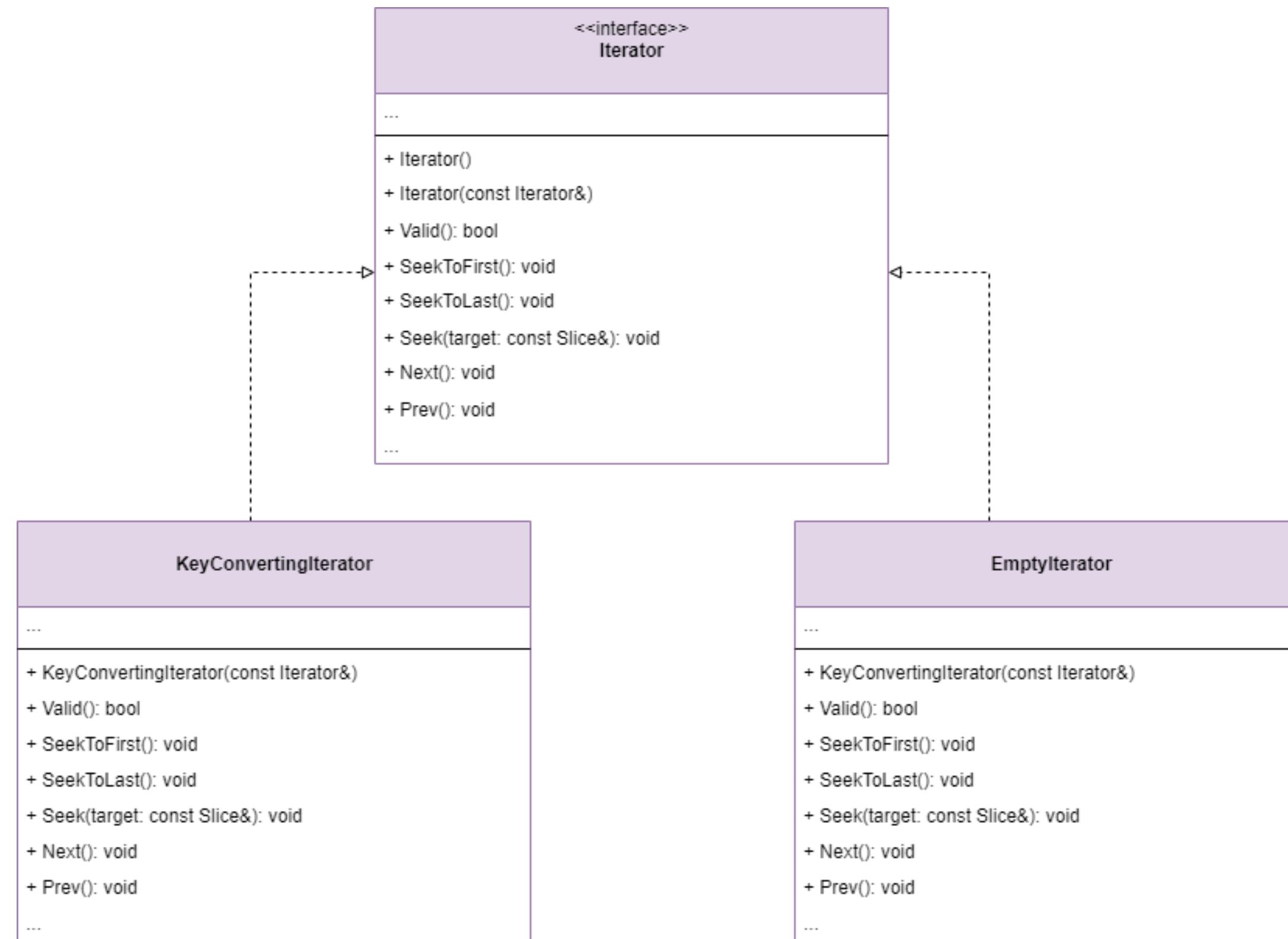
<https://github.com/bitcoin/bitcoin/blob/7fcf53f7b4524572d1d0c9a5fdc388e87eb02416/src/leveldb/include/leveldb/iterator.h>

table_test.cc (261 - 300) https://github.com/bitcoin/bitcoin/blob/7fcf53f7b4524572d1d0c9a5fdc388e87eb02416/src/leveldb/table/table_test.cc

Null Object



Null Object



Null Object



```
1 class LEVELDB_EXPORT Iterator {
2 public:
3     Iterator();
4
5     Iterator(const Iterator&) = delete;
6     Iterator& operator=(const Iterator&) = delete;
7
8     virtual ~Iterator();
9     virtual bool Valid() const = 0;
10    virtual void SeekToFirst() = 0;
11    virtual void SeekToLast() = 0;
12    virtual void Seek(const Slice& target) = 0;
13    virtual void Next() = 0;
14    virtual void Prev() = 0;
15    virtual Slice key() const = 0;
16    virtual Slice value() const = 0;
17    virtual Status status() const = 0;
18    using CleanupFunction = void (*)(void* arg1, void* arg2);
19    void RegisterCleanup(CleanupFunction function, void* arg1, void* arg2);
```



```
1 class EmptyIterator : public Iterator {
2 public:
3     EmptyIterator(const Status& s) : status_(s) {}
4     ~EmptyIterator() override = default;
5
6     bool Valid() const override { return false; }
7     void Seek(const Slice& target) override {}
8     void SeekToFirst() override {}
9     void SeekToLast() override {}
10    void Next() override { assert(false); }
11    void Prev() override { assert(false); }
12    Slice key() const override {
13        assert(false);
14        return Slice();
15    }
16    Slice value() const override {
17        assert(false);
18        return Slice();
19    }
20    Status status() const override { return status_; }
21
22 private:
23     Status status_;
24 };
```



```
1 class KeyConvertingIterator : public Iterator {
2 public:
3     explicit KeyConvertingIterator(Iterator* iter) : iter_(iter) {}
4     KeyConvertingIterator(const KeyConvertingIterator&) = delete;
5     KeyConvertingIterator& operator=(const KeyConvertingIterator&) = delete;
6
7     ~KeyConvertingIterator() override { delete iter_; }
8
9     bool Valid() const override { return iter_->Valid(); }
10    void Seek(const Slice& target) override {
11        ParsedInternalKey ikey(target, kMaxSequenceNumber, kTypeValue);
12        std::string encoded;
13        AppendInternalKey(&encoded, ikey);
14        iter_->Seek(encoded);
15    }
16    void SeekToFirst() override { iter_->SeekToFirst(); }
17    void SeekToLast() override { iter_->SeekToLast(); }
18    void Next() override { iter_->Next(); }
19    void Prev() override { iter_->Prev(); }
20
21    Slice key() const override {
22        assert(Valid());
23        ParsedInternalKey key;
24        if (!ParseInternalKey(iter_->key(), &key)) {
25            status_ = Status::Corruption("malformed internal key");
26            return Slice("corrupted key");
27        }
28        return key.user_key;
29    }
30
31    Slice value() const override { return iter_->value(); }
32    Status status() const override {
33        return status_.ok() ? iter_->status() : status_;
34    }
35
36    private:
37        mutable Status status_;
38        Iterator* iter_;
39    };
40};
```

iterator.h (24-81)

<https://github.com/bitcoin/bitcoin/blob/7fcf53f7b4524572d1d0c9a5fdc388e87eb02416/src/leveldb/include/leveldb/iterator.h>

iterator.cc (43-66)

<https://github.com/bitcoin/bitcoin/blob/7fcf53f7b4524572d1d0c9a5fdc388e87eb02416/src/leveldb/table/iterator.cc>

table_test.cc (261 - 300)

https://github.com/bitcoin/bitcoin/blob/7fcf53f7b4524572d1d0c9a5fdc388e87eb02416/src/leveldb/table/table_test.cc

Our Bitcoin Documentation



<https://docs.google.com/document/d/1VQUSAFWKW2doh8wIQZe4aP394fesWaxxP3LEclal8OM/edit?usp=sharing>

62010966
สุกธิราช ภูโภ



CRYPTO AVENGERS

เรียนวิชานี้มัน SAD ดีวะ

62010718
ภูมิพงศ์ บุญดี

62010986
สุรวิช ยอดแสง

62010948
สิริวัชญ์ สุขวัฒนาวิทย์

62010336
รุนกร ชาญเชิงพาณิช

62010345
รุ่งดา สินอนันต์วันิช