BUIDLING WITH IDRX STABLECOIN ON LISK BOOTCAMP

Pertemuan ke – 2

SMART CONTRACT DEVELOPMENT ON LISK

1. Development Environment

   IDE adalah singkatan dari **Integrated Development Environment**, atau dalam bahasa Indonesia disebut **Lingkungan Pengembangan Terpadu**. IDE adalah sebuah **aplikasi perangkat lunak** yang digunakan oleh programmer untuk menulis, mengedit, menguji, dan men-debug kode program secara efisien dalam satu tempat.

   Komponen utama IDE biasanya mencakup:

1. **Code Editor**
   Tempat untuk menulis dan mengedit kode program, biasanya dilengkapi dengan fitur seperti syntax highlighting (pewarnaan sintaks), auto-completion, dan indentasi otomatis.
2. **Compiler/Interpreter**
   Alat untuk menerjemahkan kode yang kamu tulis menjadi bahasa mesin agar bisa dijalankan oleh komputer.
3. **Debugger**
   Alat untuk menemukan dan memperbaiki kesalahan dalam program dengan cara menjalankan kode langkah demi langkah.
4. **Build Automation Tools**
   Fitur yang membantu menjalankan tugas-tugas otomatis seperti kompilasi, pengujian, dan deployment.
5. **Terminal/Command Line**
   Beberapa IDE juga menyediakan terminal bawaan untuk menjalankan perintah langsung dari dalam IDE.

   Contoh IDE populer:

- **Visual Studio Code (VS Code)** — ringan, fleksibel, dan mendukung banyak bahasa pemrograman.
- **IntelliJ IDEA** — populer untuk Java dan Kotlin.
- **PyCharm** — khusus untuk Python.

- **Eclipse** — banyak digunakan untuk Java.
- **NetBeans** — juga populer untuk Java, PHP, dan lainnya.

  Kenapa IDE penting?

  Karena IDE menyatukan semua alat yang dibutuhkan untuk pengembangan dalam satu tempat, kamu bisa lebih fokus menulis kode tanpa harus bolak-balik membuka berbagai program.

  Kalau kamu masih belajar, IDE juga sangat membantu karena biasanya punya fitur yang bisa mempermudah pemula, seperti saran otomatis, pengecekan error langsung, dan dokumentasi singkat.

Remix IDE adalah sebuah **IDE berbasis web** (juga tersedia versi desktop) yang **khusus dibuat untuk menulis, menguji, dan men-deploy smart contract yang ditulis dalam bahasa Solidity** untuk platform **Ethereum** dan blockchain yang kompatibel dengan EVM (Ethereum Virtual Machine).

🔧 **Fitur utama Remix IDE:**

1. **Editor Solidity**
   Tempat kamu menulis kode smart contract dalam bahasa Solidity.
2. **Solidity Compiler**
   Meng-compile kode Solidity kamu langsung di browser, menampilkan hasil bytecode dan ABI.
3. **Deploy & Run Transactions**
   Kamu bisa menjalankan dan menguji kontrak kamu langsung dari Remix, baik menggunakan:
   - Virtual Machine bawaan Remix
   - Injected Web3 (seperti MetaMask)
   - Web3 Provider (untuk testnet/mainnet)
4. **Debugger**
   Bisa menelusuri jalannya smart contract per langkah untuk memahami alur dan memperbaiki bug.
5. **Plugin system**
   Remix punya banyak plugin bawaan seperti Solidity Static Analysis, Gas Estimator, Unit Testing, dll.

⚙️ **Kenapa Remix IDE banyak dipakai?**

- **Tidak perlu install apa pun** (cukup buka di browser)
- Cocok banget untuk pemula dan prototyping cepat
- Mendukung banyak fitur lanjutan seperti testing, import library (misalnya dari OpenZeppelin), dan integrasi ke testnet/mainnet
- Mendukung **Solidity versi terbaru**

🌐 **Cara pakai:**

Kamu bisa langsung akses melalui:
👉 https://remix.ethereum.org

Contoh kasus penggunaan:

- Membuat dan menguji kontrak untuk token ERC-20 atau NFT (ERC-721)
- Simulasi crowdfunding berbasis blockchain
- Smart contract untuk voting, escrow, staking, dan lainnya

Visual Studio Code (VS Code) adalah **editor kode sumber (source code editor)** yang **gratis**, **ringan**, dan **open-source** buatan **Microsoft**. VS Code sangat populer di kalangan developer karena mendukung berbagai bahasa pemrograman dan bisa disesuaikan dengan berbagai ekstensi.

🔧 **Fitur Utama VS Code:**

1. **Code Editor**
    - Syntax highlighting (pewarnaan sintaks)
    - Auto-completion dan IntelliSense (saran cerdas)
    - Snippets (potongan kode siap pakai)
2. **Extensibility**
   Bisa ditambah plugin atau ekstensi seperti:
    - ESLint (pengecekan kode)
    - Prettier (penata format kode)
    - Live Server (untuk preview HTML/CSS)

    - Solidity, Python, Java, Node.js, React, dll.

3. **Integrated Terminal**
   Terminal bawaan untuk menjalankan perintah langsung dari dalam editor.
4. **Git Integration**
   Langsung bisa melakukan commit, push, pull, dan merge lewat antarmuka GUI-nya.
5. **Debugger Bawaan**
   Bisa debugging langsung untuk banyak bahasa (JavaScript, TypeScript, Python, dan lainnya).
6. **Multi-platform**
   Bisa dijalankan di **Windows**, **macOS**, dan **Linux**.

⚙️ **Cocok untuk siapa?**

- Pemula maupun profesional
- Pengembang **web**, **backend**, **blockchain**, **mobile**, dan lainnya
- Cocok buat proyek besar maupun kecil

⚙️ **Hardhat**

Hardhat adalah framework JavaScript/TypeScript untuk pengembangan smart contract dengan Solidity.

✅ **Fitur utama Hardhat:**

- Menyediakan **local Ethereum node** untuk testing (Hardhat Network)
- **Plugin system** (misalnya hardhat-waffle, hardhat-ethers, hardhat-deploy)
- **Testing dengan JavaScript atau TypeScript**
- Integrasi mudah dengan **MetaMask**, **Ethers.js**, dan **Web3.js**
- Mendukung **debugging** dan pelacakan gas
- Bisa digunakan bersama frontend React, Next.js, dsb

🔍 **Cocok untuk:**

- Developer yang sudah familiar dengan JavaScript/Node.js
- Proyek yang ingin integrasi backend atau frontend dengan smart contract
- Testing kompleks dan scripting deploy

🚀 **Cara mulai:**

npm install --save-dev hardhat

npx hardhat

⚡ Foundry

Foundry adalah framework pengembangan smart contract yang **cepat dan berbasis Rust**, tapi digunakan untuk menulis dan menguji **kode Solidity langsung** (tanpa JavaScript).

✅ Fitur utama Foundry:

- **forge** – compile, test, deploy smart contract (pakai Solidity)
- **cast** – interaksi langsung dengan blockchain (send tx, call, dsb)
- **Super cepat** karena dibangun dengan bahasa Rust
- Semua testing dilakukan dalam **Solidity**, bukan JavaScript
- Cocok untuk development berbasis CLI (command-line)

🔍 Cocok untuk:

- Developer yang ingin testing sepenuhnya dalam Solidity
- Developer yang suka tooling minimalis dan cepat
- Proyek-proyek DeFi besar (banyak proyek besar migrasi ke Foundry)

🚀 Cara mulai:

curl -L https://foundry.paradigm.xyz | bash

foundryup

🔄 Perbandingan Singkat:

Fitur	Hardhat	Foundry
Bahasa Testing	JavaScript / TypeScript	Solidity
Kecepatan	Cepat, tapi tergantung plugin	Sangat cepat
Plugin support	Banyak	Terbatas, tapi built-in kuat
Learning Curve	Lebih mudah untuk dev JS	Butuh familiaritas dengan CLI
Debugging	Lebih GUI-friendly	CLI-based
Tooling CLI	Sedikit	Sangat powerful (forge, cast)

Kalau kamu:

- Sudah biasa pakai JS/TS → **Hardhat** pas banget.
- Mau lebih performa, efisiensi CLI, dan nulis test di Solidity → **Foundry** cocok banget.

🧪 Apa itu EVM Test Network?

EVM Test Network adalah **jaringan blockchain tiruan (bukan mainnet) yang digunakan untuk menguji smart contract dan aplikasi blockchain (dApps)** sebelum diluncurkan ke jaringan utama.

EVM sendiri berarti **Ethereum Virtual Machine**, yaitu mesin virtual yang mengeksekusi smart contract Ethereum. Jadi testnet EVM bisa digunakan untuk menguji smart contract Ethereum dan blockchain lain yang kompatibel dengan EVM seperti BNB Chain, Polygon, Optimism, Arbitrum, dsb.

💡 Kenapa pakai testnet?

Karena:

- **Gratis** (pakai token palsu dari faucet)
- **Aman** (nggak pakai uang asli)
- Bisa **debug dan testing** dulu sebelum deploy ke jaringan utama (mainnet)

🌐 Contoh EVM Test Network populer:

Nama Testnet	Jaringan Utama	Keterangan Singkat
Sepolia	Ethereum	Testnet resmi Ethereum sejak 2023
Goerli	Ethereum	Testnet lama (mulai ditinggalkan)
Mumbai	Polygon	Untuk uji dApps di Polygon
Chiado	Gnosis Chain	Testnet-nya Gnosis
BNB Testnet	Binance Smart Chain	Uji smart contract di BSC
Arbitrum Sepolia	Arbitrum	Test smart contract layer 2
Lisk Sepolia	Lisk EVM	Testnet Lisk yang kompatibel EVM

🔧 **Cara pakai testnet:**

1. **Pilih testnet** (misalnya Sepolia)
2. **Dapatkan test token** dari faucet
3. Connect wallet (misalnya MetaMask) ke testnet itu
4. Deploy atau testing smart contract dari tools seperti **Remix, Hardhat, Foundry**
5. Lihat hasilnya di block explorer testnet (misalnya Sepolia Explorer)

🔐 **Apakah testnet aman?**

Ya, karena semua transaksi pakai token uji coba (gratis), **tidak berisiko kehilangan uang asli**. Tapi smart contract tetap harus aman karena bug bisa ketahuan sejak dini.

🧪 **Apa itu Lisk Sepolia Testnet?**

Lisk Sepolia adalah **jaringan testnet** untuk **Lisk EVM**, yaitu jaringan blockchain **kompatibel dengan Ethereum (EVM)** yang dikembangkan oleh **Lisk** — sebuah platform Web3 dari Lisk Foundation.

Lisk Sepolia mengikuti nama konvensi Ethereum (seperti “Sepolia”) karena **menggunakan teknologi EVM dan mendukung smart contract Solidity**. Jadi kamu bisa:

- Deploy smart contract Solidity ke Lisk Sepolia
- Tes dApp kamu pakai wallet seperti MetaMask
- Gunakan alat-alat seperti **Remix, Hardhat, atau Foundry**

🧩 **Hubungan Lisk dan EVM**

Lisk dulu terkenal dengan pendekatan SDK sendiri (JavaScript). Tapi sekarang, mereka punya **EVM-based chain**, artinya kamu bisa deploy smart contract seperti di Ethereum, tapi di jaringan Lisk!

🛠️ **Fitur Lisk Sepolia Testnet:**

- **Token uji coba (LSK)** bisa didapat dari faucet (gratis)
- Bisa connect via **MetaMask**
- Bisa dilihat transaksinya di **Lisk block explorer**
- Untuk uji coba aplikasi sebelum deploy ke **Lisk Mainnet**

🌐 **Network Info Lisk Sepolia (buat MetaMask):**

Properti	**Nilai**
Network Name	Lisk Sepolia
RPC URL	https://sepolia-rpc.lisk.com *(cek dokumentasi jika berubah)*
Chain ID	4202 *(cek docs resmi untuk validasi)*
Currency Symbol	LSK
Block Explorer	https://sepolia-blockscout.lisk.com

🚀 **Cara mulai testing di Lisk Sepolia:**

1. **Tambah jaringan Lisk Sepolia ke MetaMask**
2. **Dapatkan LSK uji coba** dari faucet (biasanya via link resmi mereka)
3. **Deploy smart contract** via Hardhat, Remix, atau Foundry
4. **Uji dan pantau transaksi** di explorer mereka

🎯 **Kapan pakai Lisk Sepolia?**

- Saat kamu ingin **testing smart contract Solidity**
- Saat mengembangkan dApp untuk **jaringan Lisk EVM**
- Saat ingin pengalaman mirip Ethereum tapi dengan ekosistem Lisk

#pragma directive (atau sering disebut *pragma directive*) adalah **perintah khusus untuk compiler** dalam bahasa pemrograman seperti C, C++, Solidity, dan lainnya. Kata “pragma” berasal dari “pragmatic” yang artinya praktis—jadi tujuannya adalah memberikan *instruksi tambahan* ke compiler, tapi bukan bagian dari logika program utama.

Contoh dan Penjelasan di Beberapa Bahasa

✅ **1. Di bahasa C/C++**

#pragma once

Artinya: file header ini hanya akan di-*include* satu kali selama proses kompilasi, mencegah duplikasi definisi.

#pragma GCC optimize("O3")

Artinya: minta compiler GCC untuk mengoptimalkan kode dengan level O3 (tingkat optimalisasi tinggi).

✅ 2. Di bahasa Solidity (Ethereum Smart Contract)

pragma solidity ^0.8.0;

Artinya: kontrak ini hanya bisa dikompilasi dengan compiler Solidity versi 0.8.0 atau yang lebih baru (tapi tetap dalam versi mayor 0.8.x). Ini untuk mencegah kontrak dijalankan di compiler versi yang tidak cocok.

Sifat Umum #pragma:

- Bukan bagian dari logika program, tapi bisa memengaruhi **bagaimana compiler memproses kode**.
- Tidak semua compiler mendukung semua #pragma, jadi bisa bersifat **spesifik compiler**.
- Sering dipakai untuk: optimisasi, instruksi keamanan, pengaturan spesifik file header, dll.

✅ Apa Itu Contract Declaration?

Contract declaration adalah **pernyataan atau definisi kontrak** dalam bahasa pemrograman **Solidity** (yang biasa dipakai untuk membuat smart contract di blockchain Ethereum atau EVM-compatible chain seperti Lisk, Polygon, BNB Chain, dll).

📦 Bentuk Dasarnya:

```
contract NamaKontrak {
    // isi kontrak di sini
}
```

🧠 Artinya:

Kamu sedang **mendeklarasikan sebuah smart contract**, mirip seperti kamu membuat class di OOP (Object-Oriented Programming). Di dalamnya, kamu bisa menambahkan:

- variabel (state)
- fungsi
- event
- constructor
- modifier
- dll

📜 **Contoh:**

```
// Versi compiler Solidity
pragma solidity ^0.8.0;

// Deklarasi kontrak
contract VotingContract {
    string public candidate;

    constructor(string memory _candidate) {
        candidate = _candidate;
    }

    function vote() public {
        // logika voting nanti di sini
    }
}
```

📝 Dalam contoh di atas:

- contract VotingContract adalah **contract declaration**.
- Di dalamnya ada variabel candidate, constructor, dan fungsi vote().

📌 **Analoginya:**

Kalau kamu pernah belajar OOP di JavaScript atau TypeScript:

```
class VotingContract {
  candidate: string;

  constructor(candidate: string) {
    this.candidate = candidate;
  }

  vote() {
    // voting logic
  }
}
```

✅ Sama kayak itu, cuma dalam versi smart contract yang akan hidup di blockchain!

🔒 **Apa itu Modifier?**

modifier dipakai untuk **mengontrol perilaku fungsi**, misalnya:

- siapa saja yang boleh akses (akses kontrol)
- cek kondisi tertentu sebelum/atau sesudah fungsi dijalankan

---

✨ **Contoh Modifier:**

```
modifier onlyOwner() {
    require(msg.sender == owner, "Kamu bukan pemilik kontrak");
    _; // ini tempat fungsi utama akan dijalankan
}
```

Lalu bisa dipakai di fungsi:

```
function withdraw() public onlyOwner {
    // hanya pemilik yang bisa tarik dana
```

}

🧱 **Kesimpulan:**

Iya, **modifier adalah bagian dari isi contract declaration**, dan fungsinya untuk **mengontrol akses dan perilaku fungsi di dalam smart contract.** 🔐

✅ **Apa Itu State Variable di Solidity?**

State variable adalah **variabel yang disimpan secara permanen di blockchain**. Artinya, nilainya **tersimpan di storage Ethereum (atau EVM chain lain)**, bukan hanya di memori sementara.

📍 Letaknya **di luar fungsi**, langsung di dalam contract { ... }.

🔐 **Ciri-Ciri State Variable:**

- Disimpan di blockchain → ada **biaya gas** untuk menyimpan/mengubahnya.
- Bisa bersifat public, private, atau internal.
- Digunakan untuk menyimpan data penting seperti:
    - Alamat pemilik
    - Saldo
    - Jumlah vote
    - Nama program
    - dsb

📜 **Contoh:**

```
// Versi compiler
pragma solidity ^0.8.0;

// Deklarasi kontrak
contract SimpleStorage {
  // ⬅️ Ini contoh state variable
```

```
    uint256 public storedNumber;

    function setNumber(uint256 _num) public {
        storedNumber = _num;
    }

    function getNumber() public view returns (uint256) {
        return storedNumber;
    }
}
```

🧠 Di atas:

- storedNumber adalah **state variable**
- Ia tersimpan di blockchain
- Bisa diakses publik melalui getNumber()

🎯 **Perbandingan:**

Tipe	Lokasi	Tersimpan permanen?	Biaya gas?
state variable	di luar fungsi	✅ ya	✅ ya
local variable	di dalam fungsi	❌ tidak	❌ tidak

🧠 **Singkatnya:**

State variable = data utama kontrak yang disimpan di blockchain secara permanen.

Kalau kamu lagi bikin smart contract dan butuh menyimpan data seperti nama kandidat, total vote, pemilik kontrak, atau dana terkumpul — itu semua disimpan pakai **state variable**.