**Challenge Chapter 2**

* **Study Kasus:**

Dalam studi kasus ini, kita akan membuat sebuah Basic Banking System sederhana menggunakan konsep Pemrograman Berorientasi Objek (OOP) dan mensimulasikan operasi transaksi yang asynchronous.

* **Ketentuan:**

1. Membuat repositori baru di GitHub dengan nama **"Basic-Banking-System"**
2. Salin file **bank\_account.js** Challenge 1 ke direktori proyek di repositori baru
3. Membuat file JavaScript baru dengan nama **banking\_system.js**
4. Implementasikan kelas BankAccount dengan metode **deposit()** dan **withdraw()**
5. Gunakan **setTimeout()** untuk mensimulasikan operasi transaksi yang asynchronous
6. Commit dan push perubahan ke repositori GitHub

* **Langkah-Langkah:**

1. Buatlah repositori baru di GitHub dengan nama **Basic-Banking-System**.
2. Salin file **bank\_account.js** dari Challenge 1 ke direktori proyek di repositori baru tersebut. File ini akan berisi definisi kelas BankAccount yang akan digunakan dalam implementasi sistem perbankan.
3. Buatlah file JavaScript baru dengan nama **banking\_system.js**. File ini akan menjadi pusat implementasi sistem perbankan menggunakan konsep OOP.
4. Implementasikan kelas BankAccount dengan minimal dua metode:

* **deposit(amount)**: Metode ini akan menerima jumlah uang yang akan disimpan ke dalam akun.
* **withdraw(amount)**: Metode ini akan menerima jumlah uang yang akan ditarik dari akun, asalkan saldo mencukupi.

1. Gunakan fungsi **setTimeout()** untuk mensimulasikan operasi transaksi yang asynchronous. Misalnya, setelah melakukan deposit atau withdrawal, Anda dapat menunda eksekusi selama beberapa detik sebelum mengembalikan hasil operasi.
2. Lakukan commit dan push perubahan yang telah Anda buat ke repositori GitHub.

* **Code banking\_system.js:**

// Membuat kelas RekeningBank

class BankAccount {

  constructor(*nomorRekening*, *saldo*) {

*this*.nomorRekening = nomorRekening;

*this*.saldo = saldo;

  }

  // Metode setor() untuk menambahkan dana ke rekening

  deposit(*amount*) {

    console.log(

      `Memulai setor sebesar ${amount} ke rekening ${*this*.nomorRekening}`

    );

    setTimeout(() => {

*this*.saldo += amount;

      console.log(

        `Setor sebesar ${amount} berhasil. Saldo saat ini: ${*this*.saldo}`

      );

    }, 2000); // Menggunakan setTimeout() untuk mensimulasikan operasi asynchronous

  }

  // Metode tarik() untuk menarik dana dari rekening

  withdraw(*amount*) {

    console.log(

      `Memulai penarikan sebesar ${amount} dari rekening ${*this*.nomorRekening}`

    );

    setTimeout(() => {

      if (amount <= *this*.saldo) {

*this*.saldo -= amount;

        console.log(

          `Penarikan sebesar ${amount} berhasil. Saldo saat ini: ${*this*.saldo}`

        );

      } else {

        console.log(

          `Penarikan sebesar ${amount} gagal. Saldo tidak mencukupi.`

        );

      }

    }, 2000); // Menggunakan setTimeout() untuk mensimulasikan operasi asynchronous

  }

}

// Membuat objek RekeningBank

const rekening1 = **new** BankAccount(12345, 1000);

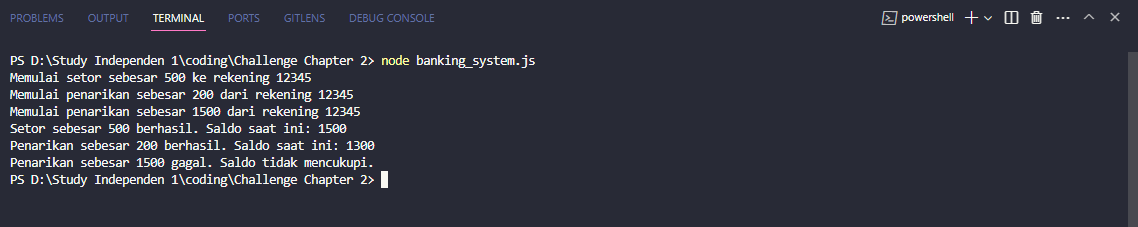
// Melakukan setor dan tarik dengan operasi asynchronous

rekening1.deposit(500);

rekening1.withdraw(200);

rekening1.withdraw(1500);

* **Output:**

****

* **Penjelasan Code:**

class BankAccount {

  constructor(*nomorRekening*, *saldo*) {

*this*.nomorRekening = *nomorRekening*;

*this*.saldo = *saldo*;

  }

1. **class BankAccount {…}**

Ini adalah deklarasi awal dari kelas BankAccount. Kelas ini akan memiliki dua properti dan beberapa metode untuk berinteraksi dengan objek-objek yang dibuat berdasarkan kelas ini.

1. **constructor(nomorRekening, saldo) {…}**

Ini adalah konstruktor kelas BankAccount. Konstruktor adalah metode khusus yang dipanggil saat objek baru dibuat berdasarkan kelas ini. Dalam hal ini, konstruktor menerima dua parameter: nomorRekening dan saldo.

1. **this.nomorRekening = nomorRekening;**

Ini adalah pernyataan yang menginisialisasi properti nomorRekening pada objek yang dibuat berdasarkan kelas ini dengan nilai yang diberikan saat objek dibuat.

1. **this.saldo = saldo;**

Ini adalah pernyataan yang menginisialisasi properti saldo pada objek yang dibuat berdasarkan kelas ini dengan nilai yang diberikan saat objek dibuat.

deposit(*amount*) {

    console.log(

      `Memulai setor sebesar ${*amount*} ke rekening ${*this*.nomorRekening}`

    );

    setTimeout(() => {

*this*.saldo += *amount*;

      console.log(

        `Setor sebesar ${*amount*} berhasil. Saldo saat ini: ${*this*.saldo}`

      );

    }, 2000); // Menggunakan setTimeout() untuk mensimulasikan operasi asynchronous

  }

Kode di atas adalah sebuah metode (fungsi) yang disebut deposit(amount) yang ditambahkan ke dalam kelas BankAccount. Metode ini digunakan untuk mensimulasikan operasi setoran (deposit) ke rekening bank dengan mengambil sejumlah amount dan menggambarkan prosesnya.

1. **deposit(amount) {…}**

Ini adalah deklarasi metode deposit dalam kelas BankAccount. Metode ini menerima satu parameter, yaitu amount, yang merupakan jumlah uang yang akan disetorkan ke rekening bank.

1. **console.log(...)**

Ini adalah pernyataan yang digunakan untuk mencetak pesan ke konsol. Dalam metode deposit, ada dua pesan yang dicetak:

* + Pesan pertama mencetak informasi tentang setoran yang akan dimulai. Pesan ini mencakup jumlah setoran (amount) dan nomor rekening (this.nomorRekening) yang menjadi tujuan setoran.
  + Pesan kedua mencetak informasi tentang berhasilnya setoran. Pesan ini mencakup jumlah setoran (amount) dan saldo terbaru (this.saldo) setelah setoran berhasil. Pesan ini dicetak setelah melalui sebuah proses simulasi asinkron menggunakan setTimeout.

1. **setTimeout(() => { ... }, 2000);**

Ini adalah metode JavaScript yang digunakan untuk menjalankan fungsi yang didefinisikan dalam lambda (fungsi anonim) setelah jeda waktu tertentu. Dalam hal ini, kita mensimulasikan operasi asinkron yang memakan waktu dengan menggunakan setTimeout. Fungsi lambda yang didefinisikan akan dijalankan setelah 2 detik (2000 milidetik) dari saat deposit dipanggil.

1. **this.saldo += amount;**

Dalam fungsi lambda yang dijalankan setelah jeda waktu, ini adalah pernyataan yang menambahkan jumlah amount ke properti saldo dari objek rekening bank. Ini menggambarkan penambahan saldo setelah setoran berhasil.

1. **console.log(...)**

Dalam fungsi lambda yang dijalankan setelah jeda waktu, kita mencetak pesan yang menggambarkan keberhasilan setoran. Pesan ini mencakup jumlah setoran (amount) dan saldo terbaru (this.saldo) setelah setoran berhasil.

withdraw(*amount*) {

    console.log(

      `Memulai penarikan sebesar ${*amount*} dari rekening ${*this*.nomorRekening}`

    );

    setTimeout(() => {

      if (*amount* <= *this*.saldo) {

*this*.saldo -= *amount*;

        console.log(

          `Penarikan sebesar ${*amount*} berhasil. Saldo saat ini: ${*this*.saldo}`

        );

      } else {

        console.log(

          `Penarikan sebesar ${*amount*} gagal. Saldo tidak mencukupi.`

        );

      }

    }, 2000); // Menggunakan setTimeout() untuk mensimulasikan operasi asynchronous

  }

}

Kode di atas adalah sebuah metode (fungsi) yang disebut withdraw(amount) yang juga ditambahkan ke dalam kelas BankAccount. Metode ini digunakan untuk mensimulasikan operasi penarikan (withdrawal) dari rekening bank dengan mengambil sejumlah amount dan menggambarkan prosesnya.

1. **withdraw(amount) {…}**

Ini adalah deklarasi metode withdraw dalam kelas BankAccount. Metode ini menerima satu parameter, yaitu amount, yang merupakan jumlah uang yang akan ditarik dari rekening bank.

1. **console.log(...)**

Ini adalah pernyataan yang digunakan untuk mencetak pesan ke konsol. Dalam metode withdraw, ada tiga pesan yang dicetak:

* + Pesan pertama mencetak informasi tentang penarikan yang akan dimulai. Pesan ini mencakup jumlah penarikan (amount) dan nomor rekening (this.nomorRekening) yang menjadi sumber penarikan.
  + Pesan kedua mencetak informasi tentang keberhasilan penarikan jika saldo mencukupi. Pesan ini mencakup jumlah penarikan (amount) dan saldo terbaru (this.saldo) setelah penarikan berhasil. Pesan ini dicetak setelah melalui sebuah proses simulasi asinkron menggunakan setTimeout.
  + Pesan ketiga mencetak pesan bahwa penarikan gagal jika saldo tidak mencukupi.

1. **setTimeout(() => { ... }, 2000);**

Seperti yang dijelaskan sebelumnya, ini adalah metode JavaScript yang digunakan untuk menjalankan fungsi yang didefinisikan dalam lambda (fungsi anonim) setelah jeda waktu tertentu. Dalam hal ini, kita mensimulasikan operasi asinkron yang memakan waktu dengan menggunakan setTimeout. Fungsi lambda yang didefinisikan akan dijalankan setelah 2 detik (2000 milidetik) dari saat withdraw dipanggil.

1. **if (amount <= this.saldo) { ... } else { ... }**

Dalam fungsi lambda yang dijalankan setelah jeda waktu, kita memeriksa apakah jumlah penarikan (amount) kurang dari atau sama dengan saldo (this.saldo) dalam rekening. Jika saldo mencukupi, maka penarikan berhasil dan jumlah penarikan dikurangkan dari saldo. Jika saldo tidak mencukupi, maka penarikan gagal dan pesan kesalahan dicetak.

// Membuat objek RekeningBank

const rekening1 = **new** BankAccount(12345, 1000);

// Melakukan setor dan tarik dengan operasi asynchronous

rekening1.deposit(500);

rekening1.withdraw(200);

rekening1.withdraw(1500);

Kode di atas adalah contoh penggunaan kelas BankAccount yang telah kita definisikan sebelumnya untuk membuat objek rekening1 dan melakukan operasi setoran (deposit) serta penarikan (withdraw) dengan operasi asynchronous.

1. **const rekening1 = new BankAccount(12345, 1000);**

Dalam baris ini, kita membuat sebuah objek rekening1 menggunakan kelas BankAccount. Objek ini dibuat dengan nomor rekening 12345 dan saldo awal sebesar 1000.

1. **rekening1.deposit(500);**

Ini adalah pemanggilan metode deposit pada objek rekening1 untuk melakukan setoran sebesar 500. Kode ini akan mencetak pesan bahwa setoran dimulai, menunggu 2 detik (karena menggunakan setTimeout untuk simulasi asinkron), dan kemudian mencetak pesan bahwa setoran berhasil, sehingga saldo akhirnya menjadi 1500.

1. **rekening1.withdraw(200);**

Ini adalah pemanggilan metode withdraw pada objek rekening1 untuk melakukan penarikan sebesar 200. Kode ini akan mencetak pesan bahwa penarikan dimulai, menunggu 2 detik (karena menggunakan setTimeout untuk simulasi asinkron), dan kemudian mencetak pesan bahwa penarikan berhasil, sehingga saldo akhirnya menjadi 1300.

1. **rekening1.withdraw(1500);**

Ini adalah pemanggilan metode withdraw pada objek rekening1 untuk melakukan penarikan sebesar 1500. Kode ini akan mencetak pesan bahwa penarikan dimulai, menunggu 2 detik (karena menggunakan setTimeout untuk simulasi asinkron), dan kemudian mencetak pesan bahwa penarikan gagal karena saldo tidak mencukupi. Setelah operasi ini, saldo tetap 1300.