“Materi UAS”

ABSTRACT DATA TYPE

* Dalam bahasa pemrograman abstraksi adalah sebuah tools untuk menghindari pemrograman yang kompleks dengan cara menyederhanakan proses pemrograman.
* Secara umum, ADT adalah sebuah tipe data yang memenuhi kondisi berikut :

1. Representasi, operasi, dan tupe objek didefinisikan dalam single syntatic unit.
2. Representasi dari objek disembunyikan dari unit program yang menggunakan objek tersebut, sehingga operasi yang memungkinkan adalah operasi yang disediakan didalam type’s definition.

(ADT tidak bisa terlepas dari attribute)

Keuntungan ADT :

1. Program terorganisasi, dapat dimodifikasi, dan kompilasi terpisah.
2. Dapat diandalkan dengan menyembunyikan representasi data, pengguna (user) tidak dapat mengakses objek secara langsung.

Fitur Utama ADT :

1. Packaging of data with their associated operation
2. Information hiding :

* Private ( - )
* Protected ( # )
* Pubic ( + )

OOP (Object-Oriented Program)

Mengapa OOP?

* OOP sangat mendukung abstraksi dari object (model) yang ada di dunia nyata dimana manusia lebih familiar dengan object nyata sehingga memudahkan dalam mengenali object tersebut. Contoh : manusia, mahasiswa, dll.

Pengenalan OOP

1. Istilah yang sering digunakan didalam OOP
2. Object merupakan sekumpulan (set) yang terdiri dari attribute dan method.

Attribute : struktur data object.

Method : fungsi yang bisa digunakan object.

1. Object itu sendiri direpresentasikan oleh sebuah nama yang mewakili object yang disebut sebagai class sehingga dijelaskan bahwa instance of class adalah object.
2. Class mempunyai turunan class yang disebut derived class (subclass), sementara induk dari class disebut superclass.
3. ADT biasanya disebut class.
4. Sub-program yang mendefinisikan operasi pada object disebut method.
5. Panggilan kepada method disebut message.
6. Seluruh koleksi dari method sebuah object disebut massage protocol dan method interface.
7. Ada 2 jenis variable didalam sebuah class : Class variable dan Instance variable.
8. Ada 2 jenis method pada class : Class method dan Instance method.
9. Karakteristik OOP :

* ADT
* Inheritance
* Dynamic binding (Polymorphism)

INHERITANCE

* Salah satu karakteristik class yang kegunaannya adalah subclass yang merupakan turunan dari superclass dapat mengambil pewarisan baik attribute maupun method dari superclass ditambah dengan apa yang dimiliki subclass itu sendiri.
* Dynamic binding berkaitan erat dengan polymorphism yang memungkinkan nama method yang sama dapat diimplementasikan dengan cara berbeda.
* Dalam C++, dynamic binding terjadi ketika sebuah virtual function dipanggil menggunakan sebuah reference atau sebuah pointer ke sebuah base class.

CONCURRENCY

Concurrent programming mendasari dua sudut pandang utama sistem operasi modern, yaitu multiprogramming dan pemrosesan terdistribusi, disamping merupakan dasar teknologi perancangan sistem operasi, konkurensi dibutuhkan dalam banyak konteks yang berbeda, antara lain aplikasi majemuk (multiple applications) pada multiprogramming, aplikasi terstuktur pada perancangan dan pemrograman terstuktur dan implementasi proses-proses pada stuktur sistem operasi.

Definisi Concurrency : Sebuah sistem komputer harus menangani beberapa program (task) uang harus dieksekusi bersama dalam sebuah lingkungan (baik mono/multi processor). Pada pemrograman konkruen, kita tidak lagi berpikir sekuensial, melainkan harus menangani komunikasi dan sinkronisasi antar program.

Concurrency dapat terjadi pada 4 level, yaitu :

1. Machine instruction level
2. High level language statement level
3. Unit level
4. Program level

Kategori Concurrency :

1. Physical Concurrency

Dikatakan physical concurrency apabila lebih dari 1 processor tersedia, beberapa unit program dari program yang sama dieksekusi secara simultan (multiple threads of control)

1. Logical Concurrency

Dikatakan logical concurrency apabila memiliki banyak processor menyediakan Actual concurrecy, dimana pada keadaan sebenarnya actual execution dari program berlangsung pada interleaved processor.

Introduction to sub-programming level concurrency :

* Task / proses adalah unit program yang dapat dieksekusi secara konkruen dengan unit program lain.
* Masing-masing task pada program dapat mendukung satu thread of control.

Karakteristik yang membedakan task dengan subprogram :

* Implisit = tidak terlihat jelas.
* Jika unit program mulai eksekusi task, tidak perlu menunggu eksekusi selesai baru melanjutkan.

Task dibagi menjadi 2 kategori umum :

1. Heavy weight task
2. Light weight task

Ada 2 jenis sinkronisasi sebagai fasilitas utama :

1. Coorperation synchronization
2. Competition synchronization

SEMAPHORES

* Semaphores adalah stuktur data yang terdiri dari counter dan antrian (queue) yang menyimpan task descriptor.
* Task descriptor adalah struktur data yang menyimpan semua informasi yang relevan terhadap state eksekusi dari sebuah task.

MONITOR

* Satu solusi untuk beberapa permasalahan semaphore pada concurrency adalah dengan mengekapsulasi struktur shared data dengan operasinya untuk membatasi akses.
* Monitor adalah tipe data abstrak untuk shared data.

MESSAGE PASSING

* Model umum untuk concurrency
* It can be model both semaphores and monitors
* It is not just for competition synchronization

EXECPTION HANDLING

* Merupakan proses untuk merespon suatu kejadian selama terjadi komputasi, pada saat terjadi kesalahan atau anomali yang membutuhkan suatu proses khusus dimana sering terjadi perubahan / kesalahan sdari alur pengeksekusian program.
* The event handler is a segment of code that’s called in response to an event.
* The special processing that may be required after detection of an exception is called exception handling.
* The exception handling code unit is called an axception handler.

FUNCTIONAL PROGRAMMING LANGUAGE

Comparing functional and Imperative language

1. Imperative language

* Efficient execution
* Complex semantics
* Complex syntax
* Concurrency is programmer designed

1. Functional language

* Simple semantics
* Simple syntax
* Inefficient execution
* Programs can automatically be made concurrent

Functional Programming Language

* Primitive function
* Lambda expression

Form is base on λ notation

e.g., ( LAMBDA ( x )( \* x x ) )

x is called variable

LOGIC PROGRAMMING LANGUAGE

* Berdasarkan pada deklarasi fakta atau aturan.
* Berdasarkan pada deklarasi fakta dan aturan yang disimpan dalam program, user dapat melakukan query untuk mendapatkan jawaban yang merupakan komputasi dari program logic tersebut