MODUL REKAYASA PERANGKAT LUNAK (RPL)

(CCC-110)

MODUL 05

SOFTWARE DESIGN & SOFTWARE DESIGN STRATEGIES

DISUSUN OLEH

MALABAY,S.KOM,M.KOM

UNIVERSITAS ESA UNGGUL

2020

SOFTWARE DESIGN & SOFTWARE DESIGN STRATEGIES

A. Kemampuan Akhir Yang Diharapkan

Setelah mempelajari modul ini, diharapkan mahasiswa mampu : Mahasiswa mampu memahami pengertian ***Software Design & Software Design Strategies.***

B. Uraian dan Contoh

Apa arti Desain Perangkat Lunak?

Persyaratan perangkat lunak pengguna diwujudkan dan diubah menjadi implementasi oleh proses desain perangkat lunak. Solusi terbaik untuk kebutuhan pengguna dikembangkan oleh desain perangkat lunak. Dalam proses mewujudkan kebutuhan, desain implementasi yang optimal diidentifikasi dengan menyusun rencananya.

Desain perangkat lunak memiliki varian yang berbeda. Beberapa variannya adalah sebagai berikut:

Desain Terstruktur

Masalah perangkat lunak dikategorikan ke dalam elemen solusi yang berbeda. Desain solusi dianggap sebagai desain terstruktur. Cara dan proses pemecahan masalah dibuat mudah dipahami dengan desain terstruktur. Desain terstruktur memungkinkan untuk menyederhanakan masalah oleh desainer.

Menurut prinsip 'bagi dan taklukkan', masalah tertentu dibagi menjadi masalah kecil yang terstruktur dengan baik dan masing-masing masalah kecil terselesaikan. Dengan menggunakan modul solusi, masalah yang terbagi diselesaikan. Modul-modul tersebut dirancang dengan cara yang terorganisir dengan baik dan bertujuan untuk mencapai solusi terbaik untuk masalah tersebut.

Modul diaktifkan untuk berinteraksi di antara lingkungannya sendiri. Beberapa aturan khusus diikuti oleh modul untuk interaksinya.

Kohesi - Semua elemen dari fungsi serupa dikelompokkan.
Kopel - Interaksi antar modul.
Desain terstruktur yang memiliki kohesi tinggi dan kopling rendah dianggap sebagai desain terstruktur yang baik.
Desain perangkat lunak adalah proses untuk mengubah kebutuhan pengguna menjadi beberapa bentuk yang sesuai, yang membantu programmer dalam pengkodean dan implementasi perangkat lunak**.**

Untuk menilai kebutuhan pengguna, dokumen SRS (Software Requirement Specification) dibuat sedangkan untuk pengkodean dan implementasi, dibutuhkan persyaratan yang lebih spesifik dan rinci dalam istilah software. Keluaran dari proses ini dapat langsung digunakan ke dalam implementasi dalam bahasa pemrograman.

Perancangan perangkat lunak adalah langkah pertama dalam SDLC (Software Design Life Cycle), yang memindahkan konsentrasi dari domain masalah ke domain solusi. Ini mencoba untuk menentukan bagaimana memenuhi persyaratan yang disebutkan dalam SRS.

Tingkat Desain Perangkat Lunak
Desain perangkat lunak menghasilkan tiga tingkat hasil:
Desain Arsitektur - Desain arsitektur adalah versi abstrak tertinggi dari sistem. Ini mengidentifikasi perangkat lunak sebagai sistem dengan banyak komponen yang saling berinteraksi. Pada level ini, para desainer mendapatkan ide dari domain solusi yang diusulkan.

Desain Tingkat Tinggi
Desain tingkat tinggi memecah konsep 'entitas tunggal-banyak komponen' dari desain arsitektur menjadi tampilan sub-sistem dan modul yang kurang abstrak dan menggambarkan interaksi satu sama lain. Desain tingkat tinggi berfokus pada bagaimana sistem beserta semua komponennya dapat diimplementasikan dalam

bentuk modul. Ia mengenali struktur modular dari setiap sub-sistem dan hubungan serta interaksinya satu sama lain.

Detailed Design

Detailed design berhubungan dengan implementasi bagian dari apa yang dilihat sebagai sistem dan sub-sistemnya pada dua desain sebelumnya. Ini lebih rinci tentang modul dan implementasinya. Ini mendefinisikan struktur logis dari setiap modul dan antarmuka
untuk berkomunikasi dengan modul lain.

Modularisasi

Modularisasi adalah teknik untuk membagi sistem perangkat lunak menjadi beberapa modul diskrit dan mandiri, yang diharapkan mampu melaksanakan tugas secara mandiri. Modul-modul ini dapat bekerja sebagai konstruksi dasar untuk keseluruhan perangkat lunak. Desainer cenderung merancang modul sedemikian rupa sehingga dapat dijalankan dan / atau dikompilasi secara terpisah dan independen.

Desain modular secara tidak sengaja mengikuti aturan strategi pemecahan masalah 'membagi dan menaklukkan' hal ini karena ada banyak manfaat lain yang melekat dengan desain modular suatu perangkat lunak.

Keuntungan dari modularisasi:

Komponen yang lebih kecil lebih mudah dirawat

Program dapat dibagi berdasarkan aspek fungsional

Tingkat abstraksi yang diinginkan dapat dibawa ke dalam program

Komponen dengan kohesi tinggi dapat digunakan kembali

Eksekusi serentak dapat dilakukan

Diinginkan dari aspek keamanan

Konkurensi

Dulu, semua perangkat lunak dimaksudkan untuk dijalankan secara berurutan. Yang dimaksud dengan eksekusi sekuensial adalah bahwa instruksi yang diberi kode akan dieksekusi satu demi satu yang menyiratkan hanya satu bagian dari

program yang diaktifkan pada waktu tertentu. Katakanlah, perangkat lunak memiliki banyak modul, maka hanya satu dari semua modul yang dapat ditemukan aktif pada saat eksekusi.

Dalam desain perangkat lunak, konkurensi diimplementasikan dengan membagi perangkat lunak menjadi beberapa unit eksekusi independen, seperti modul dan mengeksekusinya secara paralel. Dengan kata lain, konkurensi memberikan kemampuan pada perangkat lunak untuk mengeksekusi lebih dari satu bagian kode secara paralel satu sama lain.

Programer dan desainer harus mengenali modul-modul tersebut, yang dapat dibuat secara paralel.

Contoh

Fitur pemeriksa ejaan dalam pengolah kata adalah modul perangkat lunak, yang berjalan di sepanjang pengolah kata itu sendiri.

Kopling dan Kohesi

Ketika program perangkat lunak dimodularisasi, tugasnya dibagi menjadi beberapa modul berdasarkan beberapa karakteristik. Seperti yang diketahui, modul adalah sekumpulan instruksi yang disatukan untuk mencapai beberapa tugas. Meskipun demikian, dianggap sebagai entitas tunggal tetapi dapat merujuk satu sama lain untuk bekerja sama. Ada ukuran dimana kualitas desain modul dan interaksinya di antaranya dapat diukur. Ukuran ini disebut kopling dan kohesi.

Kohesi

Kohesi adalah ukuran yang menentukan tingkat ketergantungan intra di dalam elemen modul. Semakin besar kohesi, semakin baik desain programnya.

Ada tujuh jenis kohesi, yaitu -

Kohesi kebetulan - Ini adalah kohesi acak dan tidak terencana, yang mungkin merupakan hasil dari pemecahan program menjadi modul yang lebih kecil demi modularisasi. Karena tidak direncanakan, ini dapat membingungkan pemrogram dan umumnya tidak diterima.

Kohesi logis - Ketika elemen yang dikategorikan secara logis disatukan ke dalam modul, itu disebut kohesi logis.

Kohesi Temporal - Ketika elemen-elemen modul diatur sedemikian rupa sehingga diproses pada titik waktu yang sama, itu disebut kohesi temporal.

Kohesi prosedural - Ketika elemen-elemen modul dikelompokkan bersama, yang dijalankan secara berurutan untuk melakukan suatu tugas, itu disebut kohesi prosedural.

Kohesi komunikasional - Ketika elemen modul dikelompokkan bersama, yang dieksekusi secara berurutan

Desain Berorientasi Fungsi

Fungsi merupakan sub-sistem dari suatu sistem dan yang dimaksudkan untuk melaksanakan tugas-tugas sistem. Sistem merupakan tampilan atas untuk semua fungsi.

Beberapa sifat desain terstruktur diikuti oleh desain berorientasi fungsi. Bahkan Desain Berorientasi Fungsi mengikuti prinsip divide and conquer.

Informasi dan operasi transaksi disembunyikan dengan membagi sistem menjadi fungsi yang lebih kecil dengan mekanisme desain berorientasi fungsi. Informasi global dibagikan oleh modul fungsional ini dan informasi modul dibagikan di antaranya sendiri.

Status program diubah oleh fungsinya, ketika fungsi tertentu itu dipanggil oleh program. Modul lain meragukan perubahan tersebut. Desain berorientasi fungsi suite terbaik di mana status program tidak dipertimbangkan dan di mana input dipertimbangkan oleh fungsi.

Proses Desain

Diagram aliran data menggambarkan aliran data dalam sistem.

Proses perubahan negara dan data sistem digambarkan oleh DFD.

Sistem dibagi menjadi beberapa sub-sistem yang lebih kecil berdasarkan operasi yang disebut fungsi.

Masing-masing fungsi dijelaskan.

Desain Berorientasi Objek

Fokus utama dari desain berorientasi objek adalah pada entitas. Strategi yang dikembangkan oleh desain berorientasi objek ini juga menitikberatkan pada karakteristik entitas. Ada berbagai konsep yang digunakan dalam Desain Berorientasi Objek.

Objek adalah entitas sistem. Contoh objek adalah perusahaan, orang, pelanggan, dll. Setiap entitas memiliki atributnya sendiri yang dilakukan dengan mengadopsi beberapa metode.

Kelas - Objek dijelaskan dalam kelas. Instan kelas adalah objek. Fitur dari objek, fungsi, dan metode dijelaskan dan ditentukan oleh kelas.
Sumber yang berbeda dari metode atau tindakan, fungsi dijelaskan dan variabel merupakan atribut yang disimpan dari desain solusi.

Enkapsulasi - Enkapsulasi adalah kombinasi atribut dan metode. Data dan metode objek tidak diizinkan untuk diakses dengan enkapsulasi. Proses pembatasan ini dikenal sebagai penyembunyian informasi.
Warisan - Kelas serupa ditumpuk dalam hierarki melalui proses pewarisan. Variabel dan metode dapat diimpor oleh sub-kelas yang lebih rendah dari kelas yang lebih tinggi berikutnya. Inheritance memfasilitasi pembuatan kelas yang digeneralisasikan dengan mendefinisikan kelas tertentu.
Polimorfisme - Nama yang sama dapat diberikan ke metode berbeda yang melakukan tugas serupa tetapi berbeda dalam argumen. Proses ini dikenal sebagai polimorfisme. Berbagai tugas dilakukan oleh satu antarmuka. Kode dijalankan atas dasar fungsi yang dipanggil.
Proses Desain
Langkah-langkah yang terlibat dalam proses desain desain berorientasi objek adalah sebagai berikut:
Sistem sebelumnya, diagram urutan sistem, atau persyaratan dapat digunakan untuk mengembangkan desain solusi.
Objek dengan karakteristik atribut yang serupa diidentifikasi dan dikelompokkan ke dalam kelas.

Tentukan relasi dan hierarki kelas.

Tentukan kerangka aplikasi.

Apa sajakah pendekatan berbeda untuk Desain Perangkat Lunak?

Berikut ini dianggap sebagai pendekatan untuk desain perangkat lunak.

Desain Top Down

Setiap sistem dibagi menjadi beberapa sub-sistem dan komponen. Setiap sub-sistem dibagi lagi menjadi beberapa sub-sistem dan komponen. Proses pembagian ini memfasilitasi pembentukan struktur hierarki sistem.

Sistem perangkat lunak lengkap dianggap sebagai satu kesatuan

Seperti yang diketahui bahwa tahap perancangan mungkin merupakan tahap kedua dalam siklus pengembangan perangkat lunak, yang terjadi setelah tahap pengujian kelayakan dan analisis kebutuhan. Seperti namanya sendiri yang mendefinisikan bahwa dalam fase ini, perangkat lunak dirancang yang mencerminkan bagaimana perangkat lunak harus, fungsionalitas apa yang harus dikandungnya, bagaimana antarmuka pengguna seharusnya, dll. Tetapi, penting untuk mengetahui strategi yang dilakukan oleh perancang perangkat lunak. ikuti saat mengembangkan perangkat lunak apa pun. Pada artikel ini, akan membahas hal yang sama.

Secara umum, ada dua jenis strategi perancangan yang terutama diikuti dalam fase perancangan perangkat lunak apa pun selama pengembangannya:

Desain berorientasi fungsi

Desain berorientasi objek

Sekarang, mari tentukan masing-masing secara singkat dan pahami strategi di balik masing-masing:

1) Desain berorientasi fungsi

Dalam desain yang berorientasi fungsi, sistem dirancang sesuai dengan fungsionalitas yang ditawarkannya. Sistem pertama-tama diamati secara luas dan

kemudian masing-masing fungsinya diamati untuk mengidentifikasi subfungsi yang terdiri dari yang bertanggung jawab untuk melakukan fungsi tertentu itu.

Pada desain jenis ini terdapat fungsi terpusat yang terdiri dari berbagai fungsi. Perangkat lunak ini dirancang dengan cara yang sama.

Misalnya, pertimbangkan perangkat lunak untuk kalkulator. Fungsi utamanya adalah menghitung berbagai operasi dan mencetak hasilnya. Sekarang, sub fungsionalitas yang ditawarkannya adalah modulus operasi, logaritma, kuadrat, eksponen, dll. Sekarang semua fungsi ini selanjutnya terdiri dari operasi dasar seperti penjumlahan, pengurangan, dll. Seperti perkalian adalah hasil dari penjumlahan berulang, dll.

Dengan cara yang sama dalam perangkat lunak, elemen selanjutnya dapat menjadi sub-elemen untuk menjalankan fungsi dalamnya dan dapat juga bertindak sebagai sub-elemen untuk beberapa modul lainnya. Namun bersama-sama bertanggung jawab untuk menyediakan fungsionalitas yang diklaim ditawarkan oleh seluruh perangkat lunak.

2) Desain berorientasi objek

Dalam pendekatan desain berorientasi objek untuk merancang perangkat lunak apa pun, semuanya dianggap sebagai objek. Sekarang, setiap objek melakukan beberapa aktivitas dan memiliki beberapa perilaku. Ini didefinisikan melalui kelas karena semua objek yang termasuk dalam kelas yang sama akan menunjukkan jenis perilaku yang serupa dan juga akan melakukan fungsi serupa. Bagaimanapun, setiap objek akan berbeda dari objek lainnya dalam beberapa hal. Sekarang, ini ditentukan pada saat pembuatan objek ketika objek diinisialisasi dengan beberapa nilai yang memberikan objek perilaku uniknya.

Pendekatan desain berorientasi objek diistilahkan lebih baik daripada pendekatan desain berorientasi fungsi karena dalam pendekatan berorientasi objek, entitas dunia nyata dapat dengan mudah diimplementasikan di dunia komputer. Juga,

beberapa perilaku objek yang sangat dasar seperti polimorfisme, pewarisan, abstraksi, dan enkapsulasi dapat diimplementasikan melalui pendekatan ini.

Ini adalah fakta yang diketahui bahwa pengembangan perangkat lunak adalah tugas yang melibatkan beberapa aktivitas kompleks dan memakan waktu, yang digabungkan bersama untuk membuat dan merancang perangkat lunak yang unik dan berbeda dari mitranya, serta memiliki fitur dan kualitas yang luar biasa. Selain itu, sebelum memulai proses pengembangan perangkat lunak, tim pengembang, penguji, manajer, dan pemangku kepentingan lainnya yang terhubung ke proyek, melalui berbagai tahapan perencanaan, perancangan, pemrograman, dan lainnya. Hanya setelah mencapai kesuksesan di setiap tahapan ini proses pengembangan perangkat lunak dimulai. Berbagai tahapan Siklus Hidup Pengembangan Perangkat Lunak (SDLC) ini sangat signifikan dan membawa banyak kepentingan.

Insinyur perangkat lunak melakukan upaya besar untuk merencanakan seluruh proses pengembangan proyek dan memastikan bahwa tidak ada langkah yang terlewat atau dianggap tidak perlu. Demikian pula, perancangan perangkat lunak juga merupakan tahap yang sangat penting dari Siklus Hidup Pengembangan Perangkat Lunak (SDLC) dan diperlakukan dengan sangat hati-hati dan penting. Insinyur perangkat lunak, sebelum memulai proses pengembangan dan pengujian, merancang struktur dasar perangkat lunak, yang menentukan teknik, metodologi, alat, dan lainnya yang akan digunakan untuk menyelesaikan proses pengembangan. Selanjutnya juga menentukan jenis pengujian yang akan dijalankan pada perangkat lunak tersebut. Oleh karena itu, berikut adalah diskusi tentang desain perangkat lunak dan strateginya untuk membantu Anda memahami signifikansinya dalam Siklus Hidup Pengembangan Perangkat Lunak (SDLC).

Apa itu Desain Perangkat Lunak?

Langkah pertama dalam Siklus Hidup Pengembangan Perangkat Lunak (SDLC), Desain Perangkat Lunak, adalah proses penerapan solusi perangkat lunak untuk satu atau lebih rangkaian masalah. Ini mengubah persyaratan pengguna dan klien menjadi beberapa bentuk yang sesuai, yang membantu programmer dalam

pengkodean dan implementasi. Desain perangkat lunak biasanya melibatkan pemecahan masalah dan perencanaan solusi perangkat lunak. Ini mencakup desain komponen dan algoritme tingkat rendah, serta desain arsitektur tingkat tinggi. Selain itu, desain perangkat lunak memindahkan konsentrasi dari domain masalah ke drama solusi. Ini mencoba untuk menentukan bagaimana memenuhi persyaratan yang disebutkan dalam Spesifikasi Kebutuhan Perangkat Lunak (SRS).

Selain itu, untuk menilai kebutuhan pengguna, dokumen Spesifikasi Kebutuhan Perangkat Lunak (SRS) dibuat, sedangkan untuk pengkodean dan implementasi, diperlukan persyaratan yang lebih spesifik dan terperinci dalam istilah perangkat lunak. Keluaran dari proses ini dapat langsung digunakan ke dalam implementasi dalam bahasa pemrograman. Desain perangkat lunak adalah proses dan model, dimana yang pertama adalah urutan langkah-langkah yang memungkinkan perancang untuk mendeskripsikan semua aspek perangkat lunak untuk membangun.

Mendefinisikan Strategi Desain Perangkat Lunak:

Software Design Strategy adalah disiplin yang membantu perusahaan dalam memutuskan apa yang akan dibuat, mengapa membuatnya dan bagaimana berinovasi secara kontekstual, baik secara langsung maupun dalam jangka panjang. Proses ini melibatkan desain strategis dan interaksi antara desain dan strategi bisnis. Strategi desain perangkat lunak terutama tentang pengorganisasian aktivitas desain selama proses desain. Organisasi kegiatan desain, baik itu terencana atau ad-hoc mencerminkan pendekatan desainer untuk menciptakan desain. Selama proses pengembangan perangkat lunak, biasanya seorang perancang perangkat lunak memikirkan daftar masalah desain dan tugas yang perlu didiskusikan dan ditangani serta hanya akan dilakukan dengan cara ad-hoc. Lebih lanjut, strategi desain dapat membantu seorang desainer dalam menyelesaikan beberapa masalah umum seperti:

Mempromosikan adopsi teknologi.

Mengidentifikasi pertanyaan paling penting yang harus ditangani oleh produk dan layanan perusahaan.

Menerjemahkan wawasan menjadi solusi yang dapat ditindaklanjuti.

Memprioritaskan urutan peluncuran portofolio produk dan layanan.

Menghubungkan upaya desain ke strategi organisasi.

Mengintegrasikan desain sebagai aspek fundamental dari tujuan merek strategis.

Iklan ThinkSys

Strategi yang Digunakan untuk Desain Perangkat Lunak:

Untuk mengimplementasikan desain perangkat lunak, insinyur perangkat lunak menggunakan berbagai strategi yang membantu menentukan tugas masing-masing dan membantu dalam proses perancangan. Cara strategi desain ini digabungkan dan digunakan dapat mempengaruhi hasil dan keefektifan desain akhir. Selain itu, strategi ini mungkin tidak diperlukan setiap saat, tetapi sangat membantu dalam mendapatkan hasil yang diharapkan dan dalam mengurangi risiko dari tindakan apa pun. Dengan bantuan strategi ini, seseorang dapat merancang rencana pengembangan perangkat lunak yang tepat dan memastikan bahwa semua permintaan, persyaratan, dan permintaan klien diurus. Oleh karena itu, berikut adalah beberapa strategi yang digunakan oleh para insinyur perangkat lunak untuk merancang produk perangkat lunak bebas bug yang akurat dan juga bug.

Desain Terstruktur: Ini adalah konseptualisasi masalah menjadi beberapa elemen solusi yang terorganisir dengan baik. Ini terutama berkaitan dengan desain solusi. Keuntungan terbesar dari desain terstruktur adalah memberikan under yang lebih baik

Perancangan perangkat lunak adalah proses untuk mengkonseptualisasikan kebutuhan perangkat lunak ke dalam implementasi perangkat lunak. Desain perangkat lunak mengambil kebutuhan pengguna sebagai tantangan dan mencoba menemukan solusi yang optimal. Sementara perangkat lunak sedang dikonseptualisasikan, sebuah rencana dibuat untuk menemukan desain terbaik untuk menerapkan solusi yang diinginkan.

Ada beberapa varian desain perangkat lunak. Mari pelajari secara singkat:

Desain Terstruktur

Desain terstruktur merupakan konseptualisasi masalah menjadi beberapa elemen solusi yang tertata rapi. Ini pada dasarnya berkaitan dengan desain solusi. Manfaat desain terstruktur adalah memberikan pemahaman yang lebih baik tentang bagaimana masalah diselesaikan. Desain terstruktur juga memudahkan desainer untuk berkonsentrasi pada masalah dengan lebih akurat.

Desain terstruktur sebagian besar didasarkan pada strategi 'membagi dan menaklukkan' di mana masalah dipecah menjadi beberapa masalah kecil dan setiap masalah kecil diselesaikan secara individual sampai seluruh masalah terpecahkan.

Potongan-potongan kecil masalah diselesaikan dengan menggunakan modul solusi. Penekanan desain terstruktur bahwa modul-modul ini diatur dengan baik untuk mencapai solusi yang tepat.

Modul-modul ini disusun dalam hierarki, berkomunikasi satu sama lain. Desain terstruktur yang baik selalu mengikuti beberapa aturan komunikasi antar banyak modul, yaitu -

Kohesi - pengelompokan semua elemen yang terkait secara fungsional.

Kopling - komunikasi antara modul yang berbeda.

Desain terstruktur yang baik memiliki kohesi tinggi dan pengaturan kopling rendah.

Desain Berorientasi Fungsi

Dalam desain berorientasi fungsi, sistem terdiri dari banyak sub-sistem yang lebih kecil yang dikenal sebagai fungsi. Fungsi-fungsi ini mampu melakukan tugas penting dalam sistem. Sistem dianggap sebagai tampilan atas semua fungsi.

Desain berorientasi fungsi mewarisi beberapa properti desain terstruktur di mana metodologi divide and conquer digunakan.

Mekanisme desain ini membagi seluruh sistem menjadi fungsi-fungsi yang lebih kecil, yang menyediakan sarana abstraksi dengan menyembunyikan informasi dan operasinya. Modul fungsional ini dapat berbagi informasi sendiri melalui penyampaian informasi dan menggunakan informasi yang tersedia secara global.

Karakteristik lain dari fungsi adalah ketika suatu program memanggil suatu fungsi, fungsi tersebut mengubah status program, yang terkadang tidak dapat diterima oleh modul lain. Desain berorientasi fungsi bekerja dengan baik di mana status sistem tidak menjadi masalah dan program / fungsi bekerja pada input daripada pada status.

Proses Desain

Keseluruhan sistem dilihat sebagai bagaimana data mengalir dalam sistem melalui diagram arus data.

DFD menggambarkan bagaimana fungsi mengubah data dan keadaan seluruh sistem.

Seluruh sistem secara logis dipecah menjadi unit-unit yang lebih kecil yang dikenal sebagai fungsi berdasarkan operasinya dalam sistem.

Setiap fungsi kemudian dijelaskan secara luas.

Desain Berorientasi Objek

Desain berorientasi objek bekerja di sekitar entitas dan karakteristiknya alih-alih fungsi yang terlibat dalam sistem perangkat lunak. Strategi desain ini berfokus pada entitas dan karakteristiknya. Seluruh konsep solusi perangkat lunak berputar di sekitar entitas yang terlibat.

Mari lihat konsep penting dari Desain Berorientasi Objek:

Objek - Semua entitas yang terlibat dalam desain solusi dikenal sebagai objek. Misalnya orang, bank, perusahaan dan nasabah diperlakukan sebagai objek. Setiap entitas memiliki beberapa atribut yang terkait dengannya dan memiliki beberapa metode untuk dilakukan pada atribut.

Kelas - Kelas adalah deskripsi umum dari suatu objek. Objek adalah turunan dari kelas. Kelas mendefinisikan semua atribut, yang dapat dimiliki objek dan metode, yang mendefinisikan fungsionalitas objek.

Dalam desain solusi, atribut disimpan sebagai variabel dan fungsionalitas ditentukan melalui metode atau prosedur.

Enkapsulasi - Dalam OOD, atribut (variabel data) dan metode (operasi pada data) digabungkan bersama disebut enkapsulasi. Enkapsulasi tidak hanya menggabungkan informasi penting dari suatu objek, tetapi juga membatasi akses data dan metode dari dunia luar. Ini disebut penyembunyian informasi.

Warisan - OOD memungkinkan kelas serupa untuk ditumpuk secara hierarkis di mana kelas yang lebih rendah atau sub-kelas dapat mengimpor, mengimplementasikan dan menggunakan kembali variabel dan metode yang diizinkan dari kelas super langsung. Properti OOD ini dikenal sebagai warisan. Ini membuatnya lebih mudah untuk menentukan kelas tertentu dan untuk membuat kelas umum dari kelas tertentu.

Polimorfisme - Bahasa OOD menyediakan mekanisme di mana metode yang melakukan tugas serupa tetapi berbeda dalam argumen, dapat diberi nama yang sama. Ini disebut polimorfisme, yang memungkinkan satu antarmuka melakukan tugas untuk berbagai jenis. Bergantung pada bagaimana fungsi tersebut dipanggil, masing-masing bagian dari kode akan dieksekusi.

Proses Desain

Proses desain perangkat lunak dapat dianggap sebagai serangkaian langkah yang didefinisikan dengan baik. Meskipun itu bervariasi sesuai dengan pendekatan desain (berorientasi fungsi atau berorientasi objek).

C. Latihan

1. Proses untuk mengubah kebutuhan pengguna menjadi beberapa bentuk yang sesuai, yang membantu programmer dalam pengkodean dan implementasi perangkat lunak,disebut ?

2. Disiplin yang membantu perusahaan dalam memutuskan apa yang akan dibuat, mengapa membuatnya dan bagaimana berinovasi secara kontekstual, baik secara langsung maupun dalam jangka panjang, disebut ?

D. Kunci Jawaban

1. Desain perangkat lunak
2. Software Design Strategy

E. Daftar Pustaka

1. Roger S. Pressman, Software Engineering A Practioner's Apporach, 2014
2. Ian Sommerville, Software Engineering (10th Edition), 2015
3. https://www.wisdomjobs.com/e-university/software-engineering-tutorial-338/software-design-strategies-26000.html
4. https://www.includehelp.com/basics/different-types-of-design-strategies-in-software-engineering.aspx
5. https://www.professionalqa.com/strategies-for-software-design#:~:text=Software%20design%20is%20one%20such,takes%20place%20in%20this%20stage.
6. https://www.tutorialspoint.com/software_engineering/software_design_strategies.htm