|  |  |  |
| --- | --- | --- |
| **BAHAGIAN PENDIDIKAN DAN LATIHAN TEKNIK VOKASIONAL**  **KEMENTERIAN PENDIDIKAN MALAYSIA**  **ARAS 5 & 6, BLOK E14, KOMPLEKS E,**  **PUSAT PENTADBIRAN KERAJAAN PERSEKUTUAN**  **KERTAS PENERANGAN**  ***(INFORMATION SHEET)***  C:\Users\norsu\Desktop\jata negara.png | | |
| **KOD DAN NAMA PROGRAM NOSS** | IT-010-3:2016 PEMBANGUNAN APLIKASI | |
| **TAHAP NOSS** | 3 | |
| **KOD, NAMA CU DAN WA NOSS** | CU02/WA1 – INTERPRET APPLICATION MODULE DEVELOPMENT REQUIREMENT | |
| **NAMA PROGRAM KV** | SISTEM PENGURUSAN PANGKALAN DATA DAN APLIKASI WEB | |
| **KOD DAN NAMA KURSUS KSKV** | KPD2033 FUNDAMENTAL OF PROGRAMMING | |
| **NO DAN TAJUK STANDARD KANDUNGAN KSKV** | K1 INTERPRET APPLICATION MODULE DEVELOPMENT REQUIREMENT | |
| **NO. KOD NOSS** | IT-010-3:2016-C02/P(13/70) | Muka Surat : 1 Drp : 11 |
| **NO. KOD KSKV** | KPD2033/P(5/13) |

**TAJUK/***TITLE***:**

**KONSEP PENGATURCARAAN BERORIENTASIKAN OBJEK**

**TUJUAN/***PURPOSE* **:**

Kertas penerangan ini adalah bertujuan menerangkan mengenai :

* Konsep asas Pengaturcaraan Berorientasikan Objek.
* Ciri-ciri JAVA
* Translator

**PENERANGAN/***INFORMATION* **:**

**KONSEP ASAS BAGI PENGATURCARAAN BERORIENTASIKAN OBJEK.**

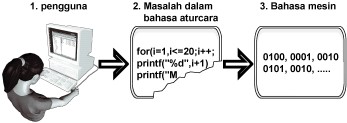
**1.BAHASA PENGATURCARAAN**

* **Ia juga boleh disebut sebagai bahasa buatan untuk pengaturcaraan komputer**. Secara amnya, komputer itu sendiri adalah dungu dan tidak boleh berfikir dengan sendiri. Komputer memerlukan arahan dari manusia untuk melakukan sesuatu tugas yang amat mudah (seperti melakukan operasi 2+2) hingga ke satu tugas yang paling sukar (seperti melancarkan kapal angkasa ke Marikh). Oleh itu, manusia menggunakan bahasa pengaturcaraan untuk berkomunikasi dan menyuruh komputer melakukan sesuatu tugasan. Arahan dalam aturcara disusun secara logik menerusi proses pengaturcaraan. Terdapat banyak jenis bahasa pengaturcaraan yang boleh digunakan oleh pengaturcara untuk berhubung dengan komputer.
* **Bahasa pengaturcaraan menyerupai bahasa tabii seperti Bahasa Inggeris dalam beberapa aspek**. Contohnya, ia mempunyai perkataan, simbol dan sintaks. Untuk menulis aturcara, pengaturcara perlu mengetahui perkataan, simbol dan sintaks bahasa pengaturcaraan yang akan digunakan. Terdapat banyak bahasa pengaturcaraan seperti APL, Pascal, Java, Perl, Eifel, C, C++, Basic, Fortran, Lisp, Scheme, BETA, Lua, Visual Basic, Prolog, Haskell, Hugs dan COBOL. Sebenarnya dianggarkan terdapat lebih 1000 bahasa pengaturcaraan di dunia.
* Setiap bahasa pengaturcaraan dicipta untuk tujuan yang berlainan seperti yang ditunjukkan di dalam Jadual 1.

|  |  |
| --- | --- |
| Dekad | Bahasa Pengaturcaraan |
| 50an | Lisp, FORTRAN |
| 60an - 70an | BASIC, C, SmallTalk, Pascal, Scheme |
| 90an | Objective-C, C++, BETA, Java dan Prolog |

Jadual 1: Evolusi Bahasa Pengaturcaraan

* Ada juga bahasa untuk tujuan khas seperti pengaturcaraan selari seperti ConcurrentC, Orca dan Leda.
* Rajah 1 menunjukkan model hubungan di antara pengguna, bahasa aturcara dan komputer.



Rajah 1: Model hubungan di antara pengguna, bahasa aturcara dan komputer

* **KENAPA KITA PERLU MEMAHAMI BAHASA PENGATURCARAAN**

Pengguna boleh menggunakan apa sahaja bahasa pengaturcaraan yang diketahui dalam menyelesaikan masalah. Namun demikian, jika pengguna hanya tahu satu bahasa pengaturcaraan sahaja, penyelesaian yang digunakan mungkin tidak optimal atau efisyen. Oleh itu, pengguna perlu memahami keupayaan sesuatu bahasa pengaturcaraan supaya pengguna dapat menilai kelebihan dan kelemahan bahasa pengaturcaraan tersebut dalam menyelesaikan masalah yang dihadapi. Setelah membuat penilaian, pengguna boleh menggunakan kaedah yang optimal dari beberapa kaedah yang difikirkan untuk menyelesaikan masalah. Di antara faedah memahami bahasa pengaturcaraan adalah;

* Mengetahui kelebihan bahasa pengaturcaraan dan di mana ia sesuai digunakan. Sebagai contoh, jika melibatkan pembangunan satu prototaip secara pantas, adalah lebih efisyen menggunakan bahasa skrip seperti Python dari menggunakan C atau C++.
* Memahami kelemahan bahasa pengaturcaraan. Setiap bahasa sesuai untuk satu bidang dan tidak efisyen untuk bidang lain. Sebagai contoh, untuk membangunkan GUI, adalah lebih efisyen menggunakan C++ atau Java berbanding dengan C. Untuk pengiraan yang banyak, FORTRAN adalah lebih sesuai dari JAVA atau C.
* Menghasilkan penyelesaian yang lebih efisien. Jika pengguna memahami bahasa pengaturcaraan, pengguna boleh menggunakan bahasa yang paling optimal untuk menyelesaikan masalah yang cuba diselesaikan. Misalnya, jika penyelesaian memerlukan capaian kepada pangkalan data, adalah lebih baik menggunakan bahasa skrip seperti Python sebagai penghubung berbanding dengan C atau C++.
* Bahasa pengaturcaraan akan mempengaruhi cara pengguna berfikir dalam menyelesaikan masalah. Misalnya, bahasa berasaskan objek akan membuat pengguna berfikir cara penyelesaian masalah berasaskan objek, dan bahasa berasaskan logik akan membuat pengguna berfikir cara penyelesaian berasaskan logik.
* Bahasa pengaturcaraan adalah satu alat untuk menyelesaikan masalah dengan menggunakan komputer. Kebanyakan masalah adalah lebih efisyen jika diselesaikan oleh mereka yang pakar dalam bidang yang tertentu. Oleh itu, mereka yang merupakan pakar dalam bidang itu boleh mempelajari bahasa pengaturcaraan untuk menyelesaikan masalah mereka.
* Adalah menjadi kebiasaan untuk seorang pengaturcara mengetahui lebih dari satu bahasa pengaturcaraan. Lazimnya, setiap bahasa pengaturcaraan mempunyai ciri-ciri yang hampir sama. Setelah mempelajari satu bahasa pengaturcaraan, adalah mudah untuk belajar bahasa pengaturcaraan lain kerana konsep pengaturcaraan adalah sama, hanya sintaks bahasa yang berubah.
* Sebuah atur cara komputer yang terdiri daripada sekumpulan unit-unit atau objek. Untuk membolehkan perkomputeran berlaku, setiap objek berupaya untuk menerima dan menghantar mesej (pesanan) kepada objek lain.
* Satu program yang menggabungkan data dan fungsi bagi menghasilkan objek.
* Objek pula merupakan item yang mengandungi data dan prosedur yang boleh bertindak ke atas data.
* Boleh digunakan untuk memanggil semula objek yang telah digunakan sebelumnya
* Contoh: C++ , java
* Pengaturcaraan berorientasi objek (OOP) adalah satu cara menyusun kod di dalam program dengan kumpulan ke dalam objek-individu unsur-unsur yang termasuk maklumat (nilai-nilai data) dan fungsi.

**PENGATURCARAAN BERORIENTASIKAN OBJEK**.

* **Pengaturcaraan berorientasi objek mempunyai dua bahagian iaitu  strategi dan teknik,** untuk  membentuk program (sering dipanggil reka bentuk berorientasikan objek).

|  |  |
| --- | --- |
| Pewarisan (*inheritance*) | Lazimnya sebuah kelas boleh memiliki "subkelas" yang mengkhususkan kelas tersebut. |
| Pengkapsulan (*encapsulation*) | Mengasingkan pelaksanaan (*implementasi*) daripada antaramuka |
| Pengabstrakan (*abstraction*)-- | Kemampuan sebuah program untuk melewati aspek informasi yang diproses olehnya, yaitu kemampuan untuk memfokus pada inti. |
| Polimorfisme (*polymorphism*) | Menggunakan nama yang sama untuk memulakan operasi yang berlainan pada objek yang menggunakan jenis data berbeza. |

Jadual1: Ciri-ciri Pengaturcaraan Berorientasikan Obje

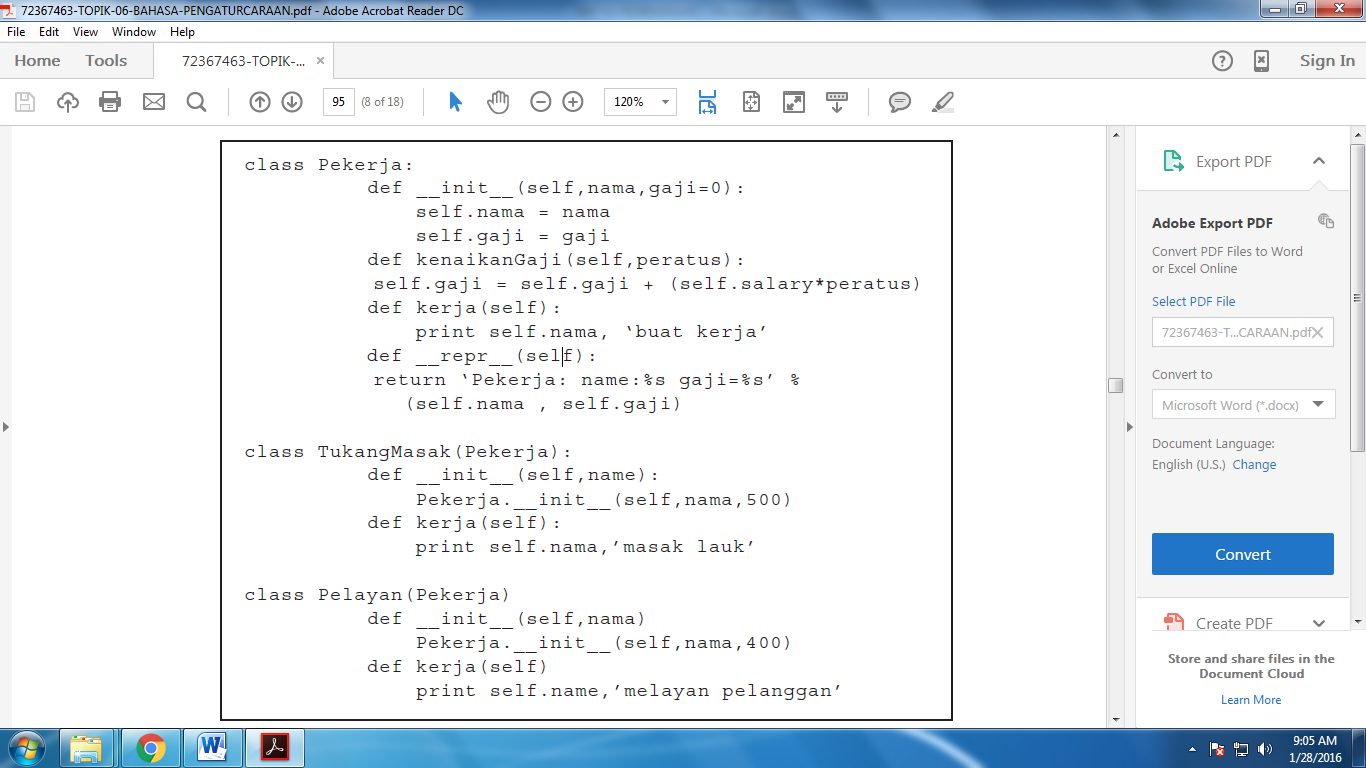
|  |  |
| --- | --- |
| **PENDEKATAN BERORIENTASIKAN OBJEK** | **PENDEKATAN BERSTRUKTUR** |
| Guna objek - objek | Guna rekabentuk atas bawah |
| gabungkan data manakala fungsinya adalah untuk merekacipta objek | kesemua program dipecahkan kepada bahagian - bahagian kecil dan dikenali sebagai modul |
| setiap objek mempunyai hubungan antara satu sama lain. | Selalu digunakan untuk mengurus dan meletakkan kod - kod program di mana bekerja sebagai hieraki kepada modul |
| contoh: smalltalk,java visual basic, c++ | e.contoh: ada, pascal, fotran |

Rajah 2 : Jadual perbandingan Pendekatan berasaskan objek dan pendekatan berstruktur.

**KELEBIHAN PENGATURCARAAN BERORIENTASIKAN OBJEK**

* Keperluan untuk bahasa berorientasikan objek timbul apabila bahasa yang sedia ada tidak lagi dapat memenuhi keperluan sistem yang semakin kompleks.
* Antara kelebihan pengaturcaraan berorientasikan objek adalah seperti berikut:
* Kemudahan dalam pengurusan sistem
* Kemampuan untuk mengubah sistem secara teratur
* Penggunaan semula komponen dalam sistem yang baru.
* Lebih mudah untuk direka bentuk dan difahami.
* Boleh mengendalikan imej, audio, video dan bunyi lebih mudah dari bahasa prosedur (generasi ketiga).
* Fokus kepada objek dalam sesuatu sistem, bukan kepada fungsi yang berlaku dalam satu sistem.
* Satu kelas menerangkan tentang satu objek, di mana objek ini mengandungi data tentang objek dan suruhan yang boleh diproses oleh objek.
* Contoh 🡪 Pengguna ingin memodelkan satu restoran seperti yang ditunjukkan dalam Rajah berikut :





Rajah 3 : Objek dalam pengaturcaraan berorientasikan objek

* Jika kita teliti, kod ini menerangkan tentang tiga objek; Pekerja, TukangMasak dan Pelayan. Objek Pekerja mempunyai data tentang nama, gaji dan suruhan bagaimana mengira kenaikan gaji. Objek TukangMasak dan Pelayan mempunyai data tentang kerja yang dilakukan seperti “masak lauk” dan “melayan pelanggan”.
* Sebagai contoh, objek TukangMasak adalah warisan dari objek Pekerja. Ini bermakna, objek TukangMasak mempunyai data dan suruhan yang sama dengan objek Pekerja. Perhatikan bahawa objek TukangMasak menakrifkan semula maklumat iaitu;

def kerja(self):

print self.nama, ‘buat kerja’

kepada

def kerja(self):

print self.nama,’masak lauk’

* Kaedah di mana satu objek mempunyai data dan suruhan disebut **enkapsulasi**.
* **Enkapsulasi / Encapsulation** bermaksud memasukkan ke dalam bekas.
* **Polimorphisma** merujuk kepada banyak bentuk iaitu data dalam objek boleh terdiri dari berbagai bentuk. Sebagai contoh, data self.gaji tidak dinyatakan bentuknya, sama ada integer, titik terapung atau aksara. Penghantaran mesej digunakan apabila satu objek ingin berhubung dengan satu objek yang lain. Interaksi antara objek adalah melalui mesej.

**2.CIRI-CIRI BAGI JAVA**

1. **Ringkas (Simple)**- merupakan ciri utama bahasa Java. Ia telah direkabentuk sehampir mungkin dengan C++ dari pelbagai aspek. Tujuannya supaya pengaturcara(programmer) C++ boleh berhijrah kepada Java dengan lebih mudah dan cepat. Ciri-ciri C++ yang kompleks dan sukar tidak dimasukkan.  
  
2.**Berorientasikan Objek**- Java telah direkabentuk sebagai suatu bahasa berorientasikan objek daripada peringkat awal lagi. Keperluan kepada sistem yang berasaskan client/server yang teragih(distributed) adalah selari dengan prinsip pengkapsulan(encapsuilation) dan penghantaran mesej dalam paradigma berorientasikan objek. Ia menyediakan satu platform pembangunan(development) berasaskan objek yang kemas dan cekap.  
  
3. **Teragih (distributed)** - Keupayaan rangkain yang terdapat pada Java adalah berkuasa dan baik serta mudah digunakan. Pustaka Java (Java Library) mempunyai kelas yang mampu bekerjasama dengan TCP/IP seperti HTTP dan FTP. Ia berupaya membuka dan mencapai objek merentasi rangkaian melalui URL (United Resource Locater) malah ia juga memudahkan penulisan skrip (code) antaramuak CGI (Common Gateway Interface) dan melalui mekanisme servlet.  
  
4. **Teguh (Robust)** - Java telah direkabentuk sebagai suatu bahasa pengaturcaraan yang mempunyai tahap kebolehpercayaan yang sangat tinggi. Ia menyediakan kemudahan semakan masa kompil dan semakan run-time yang baik. Bentuk ini menggalakkan programmer mengamalkan kaedah programming yang reliable.  
  
5. **Selamat (Secure)** - Java direkabentuk untuk persekitaran teragih (distributed environment). Dengan adanya ciri keselamatan dalam bahasa dan sistem run-timenya, ia boleh dihalang dari perkara-perkara berikut:  
# melampaui timbunan run-time  
# merosakkan ruang ingatan diluar ruang prosesnya  
# membaca atau menulis fail tempatan (local file) apabila diarah melalui suatu pemuat(loader) yang sedar akan ciri keselamatan.  
  
6. **Architechture-natural/ Tidak bergantung kepada senibina komputer/mesin** - Java tidak bergantung kepada mana-mana senibina komputer. Ini kerana compiler Java menjanakan fail objek Java yang formatnya tidak bergantung kepada komputer. Kod Java yang terkompil boleh dilaksanakan pada sebarang mesin yang mempunyai sistem run-time Java.  
  
7. **Mudah alih (portable)** - Ciri mudah alih Java dikuatkan lagi dengan mengetatkan takrifan bahasa primitif. Saiz jenis data primitif dan perlakuan operator aritmetik telah ditetapkan oleh Java.  
Contoh :  
"saiz jenis data int adalah sama sahaja pada mesin SPARC atau mesin Intel Pentium".  
  
8.**Ditafsirkan** - Fail objek yang dijanakan oleh compiler Java boleh dilaksanakan pada sebarang mesin asalkan mesin itu mempunyai pentafsir dan sistem run-time Java. Pengaturcara tidak perlu menghasilkan satu versi aturcara bagi setiap platform kerana ia boleh dilaksanakan pada mana-mana platform yang mempunyai sistem run-time.  
  
9. **Berprestasi tinggi** - Apabila bytecode Java ditafsirkan oleh pentafsir Java biasa, ia diterjemahkan (pada run-time) kepada kod mesin komputer yang melaksanakan aplikasi. Prestasinya tidak boleh dikategori sebagai terbaik. Untuk membaiki keadaan ini, pengkompil JIT(Just In Time compiler) diperkenalkan. Prestasi akan meningkat kerana kod natof boleh dilaksanakan terus pada perkakasan.  
  
10. **Multithreaded(multibebenang)** - Aplikasi seperti pelayar web "HotJava" selalunya melibatkan perlaksanaan pelbagai tugas/aktiviti secara serentak.  
  
11. **Dinamik** - Java adalah suatu bahasa yang dinamik. Sebarang kelas Java boleh dimuatkan ke dalam pentafsir Java bila-bila masa sewaktu perlaksanaannya. Pustaka kod natif juga boleh dimuatkan secara dinamik. Dengan adanya kelas Class dalam pustaka Java, maklumat mengenai suatu kelas boleh diperolehi secara dinamik pada masa larian.

**3. TRANSLATOR**

**TRANSLATOR / PENTERJEMAH**

* Penterjemah
* Dikenali sebagai Translator.
* Memeriksa sintaks sesuatu program bagi memastikan aturcara adalah betul
* Menterjemahkan aturcara ke bahasa yang boleh difahami oleh computer
* Memberikan mesej yang dipanggil ralat sintaks (syntax-error) kepada programmer sekiranya terdapat kesilapan pada aturcara.
* Sintaks – satu set peraturan yang mesti diikuti apabila menulis aturcara.
* Ralat Sintaks – kesilapan melanggar peraturan sesuatu bahasa pengaturcaraan
* 6 kriteria program (aturcara) yang baik
* Ketepatan (accuracy) 🡪 telah memenuhi semua spesifikasi, keperluan dan objektif
* Boleh dipercayai (Reliability) 🡪 tiada ralat. Menghasilkan output yang dikehendaki
* Kebolehgunaan (usability) 🡪 senang untuk digunakan dan dipelajari.
* Penyelenggaraan (Maintainability) 🡪 aturcara senang untuk difahami, diuji dan diubah tanpa memerlukan perubahan yang menyeluruh.
* Kecekapan (efficiency) 🡪 melakukan pemprosesan dengan cekap dan pantass dalam capaian dan menghasilkan output.
* Mudah dibaca (readability) 🡪 guna pemboleh ubah yang bermakna. Mempunyai komen untuk terang fungsi.aturcara jelas dan mudah difahami.
* Terdapat 3 jenis penterjemah:
* Penghimpun

Arahan untuk penghimpun ditulis menggunakan kod arahan berbentuk simbolik atau singkatan yang bermakna atau mnemonik.

Menukar aturcara kepada bahasa mesin.

Contoh : macro 80 assembler

* Pentafsir

Menterjemah aturcara baris demi baris kepada kod mesin dan melaksanakannya.

Setiap baris aturcara akan diterjemah satu persatu sehingga tamat.

Jika berlaku ralat sintaksis, aturcara tidak akan terlaksana dan mesej ralat akan dipaparkan di skrin.

Contoh : basic dan logo

* Pengkompil

Menukar kod sumber kepada kod mesin binari iaitu 1 dan 0 sehingga keputusan diperoleh.

Sewaktu pengkompilan, jika terdapat ralat sintaksis mesej akan dipaparkan di skrin.

Pembetulan boleh dilakukan, tetapi perlu dikompil semula bagi memastikan aturcara dapat dijalankan.

Contoh : c++

**SOALAN/***QUESTION* **:**

1. Berikan contoh Bahasa Pengaturcaraan..
2. Nyatakan ciri – ciri bahasa pengaturcaraan berorientasikan objek.
3. Nyatakan Kelebihan Object Oriented Programming.
4. Senaraikan jenis – jenis penterjemah.

**RUJUKAN/***REFERENCE* **:**

1. Keith Cooper, Linda Torczon(2011), Engineering a Compiler (2nd Edition), Morgan Kaufmann, ISBN: 978-0120884780
2. Todd Zaki Warfel (2009), Prototyping: A Practitioner's Guide, Louis Rosenfeld Media, LLC, ISBN I-933820-21-7
3. Steve McConnell (2004), Code Complete: A Practical Handbook of Software Construction, (2nd Edition),Microsoft Press, ISBN: 978-0-7356-1967-8
4. Michael L. Scot (2009) , Programming Language Pragmatics, Third Edition, Morgan Kaufmann, ISBN-13: 978-0123745149
5. Simon Marlow (2013), Parallel and Concurrent Programming in Haskell: Techniques for Multicore and Multithreaded Programming (1st Edition), O'Reilly Media, ISBN-13: 978-1449335946